c	numeric value
a	array
x	term variable
f	function name
κ	program counter
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

l	$::=$		literals
	$\vert$	TRUE	bitmask true (`0b1111...`)
	$\vert$	FALSE	bitmask false (`0b0000...`)
v	$::=$		values
	$\vert$	l	numeric literal
	$\vert$	c	numeric value
	$\vert$	a	bytearray
$\ominus$	$::=$		unary operations
	$\vert$	`~`	bitwise not
$\oplus$	$::=$		binary operations
	$\vert$	`+`	
	$\vert$	`-`	
	$\vert$	`*`	
	$\vert$	`<<`	
	$\vert$	`>>`	
	$\vert$	`&`	bitwise and
	$\vert$	$\mathtt{\vert}$	bitwise or
	$\vert$	$\mathtt{==}_{\mathtt{s}}$	equals (sign extended)
	$\vert$	$\mathtt{!=}_{\mathtt{s}}$	
	$\vert$	$\mathtt{>}_{\mathtt{s}}$	
	$\vert$	$\mathtt{<}_{\mathtt{s}}$	
	$\vert$	$\mathtt{>=}_{\mathtt{s}}$	
	$\vert$	$\mathtt{<=}_{\mathtt{s}}$	
$\{\sigma\}$	$::=$		variable substitution
	$\vert$	$\{x_1/v_1, .., x_k/v_k\}$	
	$\vert$	$\{\sigma_1\} \cup \{\sigma_2\}$	
	$\vert$	$\{\sigma_1\} \cap \{\sigma_2\}$	
fval	$::=$		function spec
	$\vert$	$(x_1, .., x_n) : s$ `@`e	
fndef	$::=$		function definition
	$\vert$	**fdef** f *fval*	
program	$::=$		program
	$\vert$	$fndef_1; ..; fndef_n;$ **expose** *fndef*	list of fdefs
Λ	$::=$		function store
	$\vert$	$\emptyset_\Lambda$	empty function store
	$\vert$	$\Lambda[f \mapsto fval]$	define function
Γ	$::=$		global memory
	$\vert$	$\emptyset_\Gamma$	

	$\mid$	$\Gamma[a \mapsto \texttt{[]}]$	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
e	$::=$		expressions
	$\mid$	l	numeric literal
	$\mid$	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
	$\mid$	$\{\sigma\}\ e$	variable substitution
	$\mid$	$\{\sigma\}\ s\ \texttt{@}e$	function body
s	$::=$		statements
	$\mid$	$\textbf{skip}$	skip
	$\mid$	$s_1; s_2$	sequence
	$\mid$	$\textbf{def}\ x := e$	variable declaration
	$\mid$	$\textbf{adef}\ x := a$	array declaration
	$\mid$	$x := e$	variable assignment
	$\mid$	$a[e_1] := e_2$	array assignment
	$\mid$	$\textbf{for}\ x\ \textbf{from}\ v_1\ \textbf{to}\ v_2 : s$	for loop
	$\mid$	$(\{\sigma\}\ s)$	additional variable substitution
v_h	$::=$		values
	$\mid$	b_h	boolean value
	$\mid$	c_h	numeric value
$\ominus_h$	$::=$		unary operations
	$\mid$	$\texttt{!}$	logical not
	$\mid$	$\texttt{\~{}}$	bitwise not
$\oplus_h$	$::=$		binary operations
	$\mid$	$\texttt{+}$	
	$\mid$	$\texttt{-}$	
	$\mid$	$\texttt{*}$	
	$\mid$	$\texttt{<<}$	
	$\mid$	$\texttt{>>}$	
	$\mid$	$\texttt{\&}$	
	$\mid$	$\texttt{|}$	
	$\mid$	$\texttt{\&\&}$	

	$\vert$	`||`	
	$\vert$	`==`	
	$\vert$	`!=`	
	$\vert$	`>`	
	$\vert$	`<`	
	$\vert$	`>=`	
	$\vert$	`<=`	
e_h	$::=$		expressions
	$\vert$	b_h	boolean value
	$\vert$	c_h	numeric value
	$\vert$	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
	$\vert$	**hskip**	skip
	$\vert$	$s_{h1}; s_{h2}$	sequence
	$\vert$	**hdef** $x := e_h$	variable declaration
	$\vert$	**hadef** $x := a$	array declaration
	$\vert$	$x := e_h$	variable assignment
	$\vert$	$a[e_{h1}] := e_{h2}$	array assignment
	$\vert$	**hfor** x **from** v_{h1} **to** v_{h2} : s_h	for loop
	$\vert$	**hif** e_h **then** s_{h1} **else** s_{h2}	conditional branch
	$\vert$	**hreturn** e_h	return
hfval	$::=$		function spec
	$\vert$	$(x_1, .., x_n) : s_h$	
hfndef	$::=$		function definition
	$\vert$	**fdef** f_h *hfval*	
hprogram	$::=$		program
	$\vert$	$\mathit{hfndef}_1; ..; \mathit{hfndef}_n;$ **expose** *hfndef*	list of fdefs
ctx	$::=$		branch context
	$\vert$	l	numeric literal
	$\vert$	x	variable
	$\vert$	$\ominus \mathit{ctx}$	unary operation
	$\vert$	$\mathit{ctx}_1 \oplus \mathit{ctx}_2$	binary operation

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

$$\frac{\begin{array}{c}\mu = \mu'[x \mapsto v] \\ \kappa' = \kappa + 1\end{array}}{\{\Lambda, \Gamma, \mu, \kappa\}\ x \longrightarrow \{\Lambda, \Gamma, \mu, \kappa'\}\ v} \quad \textsc{Exr\_var}$$

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

$$\frac{\begin{array}{l} v' = \Gamma(a)[v] \\ \kappa' = \kappa + 1 \end{array}}{\{\Lambda, \Gamma, \mu, \kappa\}\ a[v] \longrightarrow \{\Lambda, \Gamma, \mu, \kappa'\}\ v'}\ \textsc{Exr\_arr\_get\_val}$$

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

$$\frac{\begin{array}{l} v' \equiv [\![\ominus v]\!] \\ \kappa' = \kappa + 1 \end{array}}{\{\Lambda, \Gamma, \mu, \kappa\}\ \ominus v \longrightarrow \{\Lambda, \Gamma, \mu, \kappa'\}\ v'}\ \textsc{Exr\_unop\_val}$$

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

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

$$\frac{\begin{array}{l} v_3 \equiv [\![v_1 \oplus v_2]\!] \\ \kappa' = \kappa + 1 \end{array}}{\{\Lambda, \Gamma, \mu, \kappa\}\ v_1 \oplus v_2 \longrightarrow \{\Lambda, \Gamma, \mu, \kappa'\}\ v_3}\ \textsc{Exr\_binop\_val}$$

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

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

$$\frac{\kappa' = \kappa + 1}{\{\Lambda, \Gamma, \mu, \kappa\}\ \{x_1/v_1, \ldots, x_k/v_k\}\ x_i \longrightarrow \{\Lambda, \Gamma, \mu, \kappa'\}\ v_i}\ \textsc{Exr\_subst\_var}$$

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

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

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

$$\frac{\begin{array}{l} \Lambda = \Lambda'[f \mapsto (x_1, .., x_k) : s\ \texttt{@}e] \\ \mu' = \mu \triangleright \emptyset_\mu \\ \kappa' = \kappa + 1 \end{array}}{\{\Lambda, \Gamma, \mu, \kappa\}\, f(v_1, .., v_k) \longrightarrow \{\Lambda, \Gamma, \mu', \kappa'\}\ \{x_1/v_1, .., x_k/v_k\}\ s\ \texttt{@}e}\ \textsc{Exr\_fn\_call}$$

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

$$\frac{\mu = \mu_1 \triangleright \mu_2}{\{\Lambda, \Gamma, \mu, \kappa\}\ \{\sigma\}\ \mathbf{skip}\ \texttt{@}v \longrightarrow \{\Lambda, \Gamma, \mu_1, \kappa\}\ v}\ \textsc{Exr\_skip\_val}$$

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

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

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

$$\frac{\begin{array}{l} \mu' = \mu[x \mapsto v] \\ \kappa' = \kappa + 1 \end{array}}{\{\Lambda, \Gamma, \mu, \kappa\}\ \{\sigma\}\ \mathbf{def}\ x := v\ \texttt{@}e_0 \longrightarrow \{\Lambda, \Gamma, \mu', \kappa'\}\ \{\sigma\}\ \mathbf{skip}\ \texttt{@}e_0}\ \textsc{Exr\_def\_val}$$

$$\frac{\Gamma' = \Gamma[a \mapsto []] \quad \mu' = \mu[x \mapsto a] \quad \kappa' = \kappa + 1}{\{\Lambda, \Gamma, \mu, \kappa\}\, \{\sigma\}\, \textbf{adef}\ x := a \ \texttt{@}e_0 \longrightarrow \{\Lambda, \Gamma', \mu', \kappa'\}\, \{\sigma\}\, \textbf{skip} \ \texttt{@}e_0} \quad \textsc{Exr\_def\_arr}$$

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

$$\frac{\mu' = \mu[x \mapsto v] \quad \kappa' = \kappa + 1}{\{\Lambda, \Gamma, \mu, \kappa\}\, \{\sigma\}\, x := v \ \texttt{@}e_0 \longrightarrow \{\Lambda, \Gamma, \mu', \kappa'\}\, \{\sigma\}\, \textbf{skip} \ \texttt{@}e_0} \quad \textsc{Exr\_assign\_val}$$

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

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

$$\frac{\Gamma' = \Gamma[a \mapsto \Gamma(a)[v_1 \mapsto v_2]] \quad \kappa' = \kappa + 1}{\{\Lambda, \Gamma, \mu, \kappa\}\, \{\sigma\}\, a[v_1] := v_2 \ \texttt{@}e_0 \longrightarrow \{\Lambda, \Gamma', \mu, \kappa'\}\, \{\sigma\}\, \textbf{skip} \ \texttt{@}e_0} \quad \textsc{Exr\_arr\_assign\_val}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{l} [\![e_h]\!]_t = e \\ e' = ctx\;\texttt{\&}\;rnset \\ e'' = e\;\texttt{\&}\;e' \\ e''' = x\;\texttt{\&}\;(\sim e') \end{array}}{[\![x := e_h]\!]_{ctx} = x := (e''\;\texttt{|}\;e''')}\quad \textsc{Stt\_var\_assign}$$

$$\frac{\begin{array}{l} [\![e_{h1}]\!]_t = e_1 \\ [\![e_{h2}]\!]_t = e_2 \\ e' = ctx\;\texttt{\&}\;rnset \\ e'' = e_2\;\texttt{\&}\;e' \\ e''' = a[e_1]\;\texttt{\&}\;(\sim e') \end{array}}{[\![a[e_{h1}] := e_{h2}]\!]_{ctx} = a[e_1] := (e''\;\texttt{|}\;e''')}\quad \textsc{Stt\_arr\_assign}$$

$$\frac{\begin{array}{l} [\![v_{h1}]\!]_t = v_1 \\ [\![v_{h2}]\!]_t = v_2 \\ [\![s_h]\!]_{ctx} = s \end{array}}{[\![\mathbf{hfor}\; x\;\mathbf{from}\; v_{h1}\;\mathbf{to}\; v_{h2} : s_h]\!]_{ctx} = \mathbf{for}\; x\;\mathbf{from}\; v_1\;\mathbf{to}\; v_2 : s}\quad \textsc{Stt\_for}$$

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

$$\frac{\begin{array}{l} [\![e_h]\!]_t = e \\ e' = ctx\;\texttt{\&}\;rnset \\ e'' = e\;\texttt{\&}\;e' \end{array}}{[\![\mathbf{hreturn}\; e_h]\!]_{ctx} = rval := (e''\;\texttt{|}\;rval); rnset := (rnset\;\texttt{\&}\;(\sim ctx))}\quad \textsc{Stt\_ret}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
Definition rules:        55 good    0 bad
Definition rule clauses: 121 good    0 bad
```