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

v	$::=$		values
	$\|$	$\textsc{True}$	bitmask true (`0b1111...`)
	$\|$	$\textsc{False}$	bitmask false (`0b0000...`)
	$\|$	c	numeric value
	$\|$	a	bytearray
$\ominus$	$::=$		unary operations
	$\|$	`~`	bitwise not
$\oplus$	$::=$		binary operations
	$\|$	`+`	
	$\|$	`-`	
	$\|$	`*`	
	$\|$	`<<`	
	$\|$	`>>`	
	$\|$	`&`	bitwise and
	$\|$	`|`	bitwise or
	$\|$	$\texttt{==}_{\texttt{s}}$	equals (sign extended)
	$\|$	$\texttt{!=}_{\texttt{s}}$	
	$\|$	$\texttt{>}_{\texttt{s}}$	
	$\|$	$\texttt{<}_{\texttt{s}}$	
	$\|$	$\texttt{>=}_{\texttt{s}}$	
	$\|$	$\texttt{<=}_{\texttt{s}}$	
$\{\sigma\}$	$::=$		variable substitution
	$\|$	$\{x_1/v_1, .., x_k/v_k\}$	
	$\|$	$\{\sigma_1\} \cup \{\sigma_2\}$	
	$\|$	$\{\sigma_1\} \cap \{\sigma_2\}$	
fval	$::=$		function spec
	$\|$	$(x_1, .., x_n) : s \ \texttt{@}e$	
fndef	$::=$		function definition
	$\|$	$\mathbf{fdef}\, f\ \mathit{fval}$	
$\mathit{program}$	$::=$		program
	$\|$	$\mathit{fndef}_1; ..; \mathit{fndef}_n;\ \mathbf{expose}\, \mathit{fndef}$	list of fdefs
Λ	$::=$		function store
	$\|$	$\emptyset_\Lambda$	empty function store
	$\|$	$\Lambda[f \mapsto \mathit{fval}]$	define function
Γ	$::=$		global memory
	$\|$	$\emptyset_\Gamma$	
	$\|$	$\Gamma[a \mapsto \texttt{[]}]$	new array
	$\|$	$\Gamma[a \mapsto \Gamma(a)[v_1 \mapsto v_2]]$	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
	$\mid$	$\textsc{True}$	bitmask true (`0b1111...`)
	$\mid$	$\textsc{False}$	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
	$\mid$	$\{\sigma\}\ e$	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
	$\mid$	a	bytearray
$\ominus_h$	::=		unary operations
	$\mid$	`!`	logical not
	$\mid$	`~`	bitwise not
$\oplus_h$	::=		binary operations
	$\mid$	`+`	
	$\mid$	`-`	
	$\mid$	`*`	
	$\mid$	`<<`	
	$\mid$	`>>`	
	$\mid$	`&`	
	$\mid$	`|`	
	$\mid$	`&&`	
	$\mid$	`||`	

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\boxed{[\![\ominus_h]\!]_t = \ominus}$ $\ominus_h$ 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 $\oplus$

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

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

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

$$\frac{}{[\![\,!=]\!]_t = \,!=_{\mathtt{s}}}\ \textsc{Binopt\_neq}$$

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

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

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

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

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

```
Definition rules:        56 good    0 bad
Definition rule clauses: 122 good    0 bad
```