asm.js

Dave Herman, Luke Wagner, and Alon Zakai

October 27, 2012

1 Abstract syntax

$$
\begin{array}{rcl}
b, e, f, g, x, y, z & \in & \mathit{Identifier} \\
\texttt{arguments} & \notin & \mathit{Identifier} \\
\texttt{eval} & \notin & \mathit{Identifier}
\end{array}
$$

$$
\begin{array}{rcl}
P & ::= & \texttt{function}(b, e)\ \{\ \overline{\mathit{imp}_x}\ \overline{\mathit{fn}_f}\ \mathit{exp}\ \} \\
\mathit{imp}_x & ::= & \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y\texttt{;} \\
 & \mid & \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y(b)\texttt{;} \\
 & \mid & \texttt{var}\ x\ \texttt{=}\ \texttt{new}\ e\texttt{.}y(b)\texttt{;} \\
\mathit{exp} & ::= & \texttt{return}\ f\texttt{;} \\
 & \mid & \texttt{return}\ \{\ \overline{x\texttt{:}f}\ \}\texttt{;} \\
\mathit{fn}_f & ::= & \texttt{function}\ f\texttt{(}\overline{x}\texttt{)}\ \{\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}\ \mathit{ss}\ \}
\end{array}
$$

$$
\begin{array}{rcl}
s & ::= & \{\ ss\ \} \\
& | & e\texttt{;} \\
& | & \texttt{if (}e\texttt{)}\ s \\
& | & \texttt{if (}e\texttt{)}\ s\ \texttt{else}\ s \\
& | & \texttt{return}\ v\texttt{;} \\
& | & \texttt{while (}e\texttt{)}\ s \\
& | & \texttt{do}\ s\ \texttt{while (}e\texttt{);} \\
& | & \texttt{for (}e\texttt{;}\ e\texttt{;}\ e\texttt{)}\ s \\
& | & \texttt{switch (}e\texttt{)}\ \{\ \overline{c}\ \} \\
& | & \texttt{switch (}e\texttt{)}\ \{\ \overline{c}\, d\ \} \\
& | & \texttt{break;} \\
& | & \texttt{break}\ lab\texttt{;} \\
& | & \texttt{continue;} \\
& | & \texttt{continue}\ lab\texttt{;} \\
& | & lab\texttt{:}\, s
\end{array}
$$

$$
\begin{array}{rcl}
ss & ::= & \overline{s}
\end{array}
$$

$$
\begin{array}{rcl}
c & ::= & \texttt{case}\ e\texttt{:}\, ss \\
d & ::= & \texttt{default:}\, ss \\
cd & ::= & c \mid d
\end{array}
$$

$$
\begin{array}{rcl}
\kappa_X & ::= & X\ \texttt{| 0} \\
& | & X\ \texttt{>>> 0} \\
& | & \texttt{+}X \\
& | & \texttt{[}X\texttt{[0] >>> 0}, X\texttt{[1] | 0]} \\
& | & \texttt{[}X\texttt{[0] >>> 0}, X\texttt{[1] >>> 0]}
\end{array}
$$

$$
\begin{array}{rcl}
v & ::= & \kappa_{num} \\
 & | & \texttt{[}\kappa_{num}, \kappa_{num}\texttt{]} \\
 & & \\
e & ::= & \kappa_{num} \\
 & | & \kappa_e \\
 & | & \mathit{lval} \\
 & | & \mathit{lval}\ \texttt{=}\ e \\
 & | & f\texttt{(}\overline{e}\texttt{)} \\
 & | & \mathit{unop}\ e \\
 & | & e\ \mathit{aop}\ e \\
 & | & e\ \texttt{/}\ e \\
 & | & e\ \texttt{\%}\ e \\
 & | & e\ \mathit{bop}\ e \\
 & | & e\ \mathit{relop}\ e \\
 & | & e\ \texttt{?}\ e\ \texttt{:}\ e \\
 & | & \texttt{(}\overline{e}\texttt{)} \\
 & | & \texttt{[}e, e\texttt{]} \\
 & | & e\texttt{[0]} \\
 & | & e\texttt{[1]} \\
 & & \\
\mathit{unop} & ::= & \texttt{\~{}} \mid \texttt{-} \mid \texttt{!} \\
 & & \\
\mathit{aop} & ::= & \texttt{+} \mid \texttt{-} \mid \texttt{*} \\
\mathit{bop} & ::= & \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \\
 & & \\
\mathit{relop} & ::= & \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{!=} \mid \texttt{==} \\
 & & \\
\mathit{lval} & ::= & x \\
 & | & x\texttt{[}e\texttt{]} \\
 & | & x\texttt{[0]} \\
 & | & x\texttt{[1]}
\end{array}
$$

2 Type rules

$$
\begin{array}{rcl}
\sigma, \tau & ::= & \texttt{bits1} \mid \texttt{bits32} \mid \texttt{bits64} \mid \texttt{boolish} \\
 & | & \texttt{int32} \mid \texttt{uint32} \\
 & | & \texttt{int64} \mid \texttt{uint64} \\
 & | & \texttt{float64} \\
 & | & \texttt{array}_\tau \mid \texttt{function} \mid \texttt{jsval} \\
 & | & \texttt{floor} \\
 & | & (\overline{\sigma}) \to \tau \\
 & | & \texttt{void}
\end{array}
$$

$$
\begin{array}{rcl}
\ell & ::= & lab \mid \epsilon \\
L & ::= & \{\overline{\ell}\} \\
\varepsilon & ::= & L \mid \textsf{return}
\end{array}
$$

$$
\begin{array}{rcl}
L \,;\, L' & = & L \cup L' \\
\emptyset \,;\, \textsf{return} & = & \textsf{return} \\
\{\ell, \overline{\ell'}\} \,;\, \textsf{return} & = & \{\ell, \overline{\ell'}\} \\
\textsf{return} \,;\, L & = & \textsf{return}
\end{array}
$$

$$
\begin{array}{rcl}
L \cup \textsf{return} & = & L \\
\textsf{return} \cup L & = & L \\
\textsf{return} \cup \textsf{return} & = & \textsf{return}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{type}(X \texttt{ | } \texttt{0}) & = & \texttt{int32} \\
\mathit{type}(X \texttt{ >>> } \texttt{0}) & = & \texttt{uint32} \\
\mathit{type}(\texttt{+}X) & = & \texttt{float64} \\
\mathit{type}(\texttt{[}X\texttt{[0] >>> 0}, X\texttt{[1] | 0]}) & = & \texttt{int64} \\
\mathit{type}(\texttt{[}X\texttt{[0] >>> 0}, X\texttt{[1] >>> 0]}) & = & \texttt{uint64}
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{int32}, \texttt{uint32} & <: & \texttt{bits32} \\
\texttt{int64}, \texttt{uint64} & <: & \texttt{bits64} \\
\texttt{bits1} & <: & \texttt{boolish} \\
\texttt{bits32} & <: & \texttt{boolish} \\
\texttt{bits32} & <: & \texttt{float64} \\
\texttt{float64} & <: & \texttt{jsval} \\
\texttt{function} & <: & \texttt{jsval} \\
\texttt{array}_\tau & <: & \texttt{jsval} \\
\texttt{void} & <: & \texttt{jsval} \\
\texttt{floor} & <: & (\texttt{float64}) \to \texttt{float64} \\
(\sigma) \to \tau & <: & \texttt{function}
\end{array}
$$

$$
\Gamma ::= \{\overline{x : \tau}\} \mid \Gamma, \{\overline{x : \tau}\}
$$

$$
\begin{array}{rcl}
M(\texttt{floor}) & = & \texttt{floor} \\
M(\texttt{ceil}) & = & (\texttt{float64}) \to \texttt{float64} \\
M(\texttt{sin}) & = & (\texttt{float64}) \to \texttt{float64} \\
M(\texttt{cos}) & = & (\texttt{float64}) \to \texttt{float64} \\
& \cdots &
\end{array}
$$

$$\begin{array}{rcl}
A(\texttt{Uint8Array}) & = & \texttt{uint32} \\
A(\texttt{Uint16Array}) & = & \texttt{uint32} \\
A(\texttt{Uint32Array}) & = & \texttt{uint32} \\
A(\texttt{Int8Array}) & = & \texttt{int32} \\
A(\texttt{Int16Array}) & = & \texttt{int32} \\
A(\texttt{Int32Array}) & = & \texttt{int32} \\
A(\texttt{Float32Array}) & = & \texttt{float64} \\
A(\texttt{Float64Array}) & = & \texttt{float64}
\end{array}$$

Program checking $\boxed{\vdash P \ \mathbf{ok}}$

[T-Program]
$$\frac{\begin{array}{c}\{\overline{x}\} \cap \{\overline{f}\} = \emptyset \qquad \{\overline{x}\} \cap \{b, e\} = \emptyset \qquad \{\overline{f}\} \cap \{b, e\} = \emptyset \\ \forall i.b; e; \Gamma_0 \vdash \mathit{imp}_x \ \mathbf{ok} \qquad \forall i.\Gamma_0, \Gamma_1 \vdash \mathit{fn}_f \ \mathbf{ok} \qquad \forall i.\Gamma_0, \Gamma_1 \vdash \mathit{exp} \ \mathbf{ok}\end{array}}{\vdash \texttt{function}(b, e) \ \{ \ \overline{\mathit{imp}_x} \ \overline{\mathit{fn}_f} \ \mathit{exp} \ \} \ \mathbf{ok}}$$

Import checking $\boxed{b; e; \Gamma \vdash \mathit{imp} \ \mathbf{ok}}$

[T-ImportStd]
$$\frac{\Gamma(x) = M(y)}{b; e; \Gamma \vdash \texttt{var} \ x \ \texttt{=} \ e\texttt{.}y\texttt{;} \ \mathbf{ok}}$$

[T-ImportFFI]
$$\frac{y \notin \mathit{dom}(M)}{b; e; \Gamma \vdash \texttt{var} \ x \ \texttt{=} \ e\texttt{.}y\texttt{;} \ \mathbf{ok}}$$

[T-View]
$$\frac{\Gamma(x) = \texttt{array}_{A(y)}}{b; e; \Gamma \vdash \texttt{var} \ x \ \texttt{=} \ e\texttt{.}y(b)\texttt{;} \ \mathbf{ok}}$$

[T-NewView]
$$\frac{\Gamma(x) = \texttt{array}_{A(y)}}{b; e; \Gamma \vdash \texttt{var} \ x \ \texttt{=} \ \texttt{new} \ e\texttt{.}y(b)\texttt{;} \ \mathbf{ok}}$$

Function checking $\boxed{\Gamma \vdash \mathit{fn} \ \mathbf{ok}}$

[T-Function]
$$\frac{\begin{array}{c}\{\overline{x}\} \cap \{\overline{y}\} = \emptyset \qquad \Gamma(f) = (\overline{\sigma}) \to \tau \qquad \overline{\sigma} = \overline{\mathit{type}(\kappa_x)} \qquad \tau \neq \texttt{void} \\ \Gamma, \{\overline{x : \sigma}, \overline{y : \mathit{type}(v)}\}; \emptyset \vdash \mathit{ss} : \tau/\mathsf{return}\end{array}}{\Gamma \vdash \texttt{function} \ f\texttt{(}\overline{x}\texttt{)} \ \{ \ \overline{x \ \texttt{=} \ \kappa_x\texttt{;}} \ \texttt{var} \ \overline{y \ \texttt{=} \ v}\texttt{;} \ \mathit{ss} \ \} \ \mathbf{ok}}$$

[T-VoidFunction]
$$\frac{\begin{array}{c}\{\overline{x}\} \cap \{\overline{y}\} = \emptyset \qquad \Gamma(f) = (\overline{\sigma}) \to \texttt{void} \qquad \overline{\sigma} = \overline{\mathit{type}(\kappa_x)} \\ \Gamma, \{\overline{x : \sigma}, \overline{y : \mathit{type}(v)}\}; \emptyset \vdash \mathit{ss} : \texttt{void}/\varepsilon\end{array}}{\Gamma \vdash \texttt{function} \ f\texttt{(}\overline{x}\texttt{)} \ \{ \ \overline{x \ \texttt{=} \ \kappa_x\texttt{;}} \ \texttt{var} \ \overline{y \ \texttt{=} \ v}\texttt{;} \ \mathit{ss} \ \} \ \mathbf{ok}}$$

Export checking $\boxed{\Gamma \vdash \mathit{exp} \ \mathbf{ok}}$

[T-Singleton]
$$\frac{\Gamma(f) = (\overline{\sigma}) \to \tau \qquad \tau <: \texttt{jsval}}{\Gamma \vdash \texttt{return} \ f\texttt{;} \ \mathbf{ok}}$$

[T-Module]
$$\frac{\forall f.\Gamma(f) = (\overline{\sigma}) \to \tau \wedge \tau <: \texttt{jsval}}{\Gamma \vdash \texttt{return} \ \{ \ \overline{x\texttt{:}f} \ \}\texttt{;} \ \mathbf{ok}}$$

Statement list checking $\boxed{\Gamma; L \vdash ss : \tau/\varepsilon}$

$$\text{[T-NoStatements]}\quad \frac{}{\Gamma; L \vdash \epsilon : \tau/\emptyset}$$

$$\text{[T-Statements]}\quad \frac{\forall i. \Gamma; L \vdash s_i : \tau/\varepsilon_i \qquad n > 0 \qquad \varepsilon = \varepsilon_1 \,;\, \ldots \,;\, \varepsilon_n}{\Gamma; L \vdash \overline{s} : \tau/\varepsilon}$$

Statement checking $\boxed{\Gamma; L \vdash s : \tau/\varepsilon}$

$$\text{[T-Block]}\quad \frac{\Gamma; \emptyset \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \{\ \ ss\ \ \} : \tau/\varepsilon}$$

$$\text{[T-ExprStmt]}\quad \frac{\Gamma \vdash e : \sigma}{\Gamma; L \vdash e\texttt{;} : \tau/\emptyset}$$

$$\text{[T-If]}\quad \frac{\Gamma \vdash e : \sigma <: \texttt{boolish} \qquad \Gamma; \emptyset \vdash s : \tau/\varepsilon \qquad \varepsilon' = \varepsilon \cup \emptyset}{\Gamma; L \vdash \texttt{if (}e\texttt{)}\ \ s : \tau/\varepsilon'}$$

$$\text{[T-IfElse]}\quad \frac{\Gamma \vdash e : \sigma <: \texttt{boolish} \qquad \Gamma; \emptyset \vdash s_1 : \tau/\varepsilon_1 \qquad \Gamma; \emptyset \vdash s_2 : \tau/\varepsilon_2 \qquad \varepsilon = \varepsilon_1 \cup \varepsilon_2}{\Gamma; L \vdash \texttt{if (}e\texttt{)}\ \ s_1\ \ \texttt{else}\ \ s_2 : \tau/\varepsilon}$$

$$\text{[T-ReturnExpr]}\quad \frac{\Gamma \vdash e : \tau}{\Gamma; L \vdash \texttt{return}\ \ e\texttt{;} : \tau/\mathsf{return}}$$

$$\text{[T-ReturnVoid]}\quad \frac{}{\Gamma; L \vdash \texttt{return;} : \texttt{void}/\mathsf{return}}$$

$$\text{[T-While]}\quad \frac{\Gamma \vdash e : \sigma <: \texttt{boolish} \qquad \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \qquad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})}{\Gamma; L \vdash \texttt{while (}e\texttt{)}\ \ s : \tau/\varepsilon'}$$

$$\text{[T-DoWhile]}\quad \frac{\Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \qquad \Gamma \vdash e : \sigma <: \texttt{boolish} \qquad \varepsilon' = \varepsilon - (L \cup \{\epsilon\})}{\Gamma; L \vdash \texttt{do}\ \ s\ \ \texttt{while (}e\texttt{);} : \tau/\varepsilon'}$$

$$\text{[T-For]}\quad \frac{\forall i. \Gamma \vdash e_i : \sigma_i \qquad \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \qquad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})}{\Gamma; L \vdash \texttt{for (}e_1\texttt{; }e_2\texttt{; }e_3\texttt{)}\ \ s : \tau/\varepsilon'}$$

Statement checking (cont'd) $\boxed{\Gamma; L \vdash s : \tau/\varepsilon}$

[T-Break]
$$\frac{}{\Gamma; L \vdash \texttt{break;} : \tau/\{\epsilon\}}$$

[T-BreakLabel]
$$\frac{}{\Gamma; L \vdash \texttt{break}\ \ lab\texttt{;} : \tau/\{lab\}}$$

[T-Continue]
$$\frac{}{\Gamma; L \vdash \texttt{continue;} : \tau/\emptyset}$$

[T-ContinueLabel]
$$\frac{}{\Gamma; L \vdash \texttt{continue}\ \ lab\texttt{;} : \tau/\emptyset}$$

[T-Label]
$$\frac{\begin{array}{c}\Gamma; L \cup \{lab\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon - (L \cup \{lab\})\end{array}}{\Gamma; L \vdash lab\texttt{:}\ s : \tau/\varepsilon'}$$

[T-Switch]
$$\frac{\begin{array}{c}\Gamma \vdash e : \sigma \\ \forall i.\Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\varepsilon \\ \varepsilon \neq \mathsf{return} \lor \exists i.\varepsilon_i \cup \emptyset \neq \emptyset \\ \varepsilon' = (\varepsilon \cup \bigcup_i \varepsilon_i) - (L \cup \{\epsilon\})\end{array}}{\Gamma; L \vdash \texttt{switch (}e\texttt{)}\ \{\ \bar{c}\ cd\ \ \} : \tau/\varepsilon'}$$

[T-SwitchReturn]
$$\frac{\begin{array}{c}\Gamma \vdash e : \sigma \\ \forall i.\Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \forall i.\varepsilon_i \cup \emptyset = \emptyset \\ \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\mathsf{return}\end{array}}{\Gamma; L \vdash \texttt{switch (}e\texttt{)}\ \{\ \bar{c}\ cd\ \ \} : \tau/\mathsf{return}}$$

Case checking $\boxed{\Gamma; L \vdash cd : \sigma, \tau/\varepsilon}$

[T-Case]
$$\frac{\begin{array}{c}\Gamma \vdash e : \sigma \\ \Gamma; L \vdash ss : \tau/\varepsilon\end{array}}{\Gamma; L \vdash \texttt{case}\ \ e\texttt{:}\ ss : \sigma, \tau/\varepsilon}$$

[T-Default]
$$\frac{\Gamma; L \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \texttt{default:}\ ss : \sigma, \tau/\varepsilon}$$

Expression checking $\boxed{\Gamma \vdash e : \tau}$

$$\frac{}{\Gamma \vdash \kappa_{num} : \mathit{type}(\kappa_{num})}\ \text{[T-Number]}$$

$$\frac{\begin{array}{c}\kappa_e = (e_1 \texttt{ / } e_2) \texttt{ | } \texttt{0} \Rightarrow \exists i.\Gamma \not\vdash e_i : \texttt{int32} \\ e = e_1 \ \mathit{aop}\ e_2 \Rightarrow \exists i.\Gamma \not\vdash e_i : \tau <: \texttt{bits32} \\ \Gamma \vdash e : \tau\end{array}}{\Gamma \vdash \kappa_e : \mathit{type}(\kappa_e)}\ \text{[T-Cast]}$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \text{[T-VarRef]} \qquad \frac{\Gamma(x) = \tau \qquad \Gamma \vdash e : \tau}{\Gamma \vdash x \texttt{ = } e : \tau}\ \text{[T-Assign]}$$

$$\frac{\Gamma(x) = \texttt{array}_\tau \qquad \Gamma \vdash e : \texttt{uint32}}{\Gamma \vdash x\texttt{[}e\texttt{]} : \tau}\ \text{[T-Load]} \qquad \frac{\Gamma \vdash e_1 : \texttt{uint32} \qquad \Gamma \vdash e_2 : \Gamma(x)}{\Gamma \vdash x\texttt{[}e_1\texttt{]} \texttt{ = } e_2 : \tau}\ \text{[T-Store]}$$

$$\frac{\Gamma \vdash e : \texttt{uint64}}{\Gamma \vdash e\texttt{[0]} : \texttt{uint32}}\ \text{[T-LowUint]} \qquad \frac{\Gamma \vdash e : \texttt{int64}}{\Gamma \vdash e\texttt{[0]} : \texttt{int32}}\ \text{[T-LowInt]} \qquad \frac{\Gamma \vdash e : \tau <: \texttt{bits64}}{\Gamma \vdash e\texttt{[1]} : \texttt{uint32}}\ \text{[T-HighUint]}$$

$$\frac{\begin{array}{c}\Gamma(f) = (\overline{\sigma}) \to \tau \\ \forall i.\Gamma \vdash e_i : \sigma_i\end{array}}{\Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \tau}\ \text{[T-FunCall]} \qquad \frac{\begin{array}{c}\Gamma(f) = \texttt{function} \\ \forall i.\Gamma \vdash e_i : \sigma_i <: \texttt{jsval}\end{array}}{\Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \texttt{jsval}}\ \text{[T-FFI]}$$

$$\frac{\Gamma \vdash e_1 : \sigma <: \texttt{boolish} \qquad \forall i.\Gamma \vdash e_i : \tau}{\Gamma \vdash e_1 \texttt{ ? } e_2 \texttt{ : } e_3 : \tau}\ \text{[T-Conditional]} \qquad \frac{\forall i \leq n.\Gamma \vdash e_i : \tau_i}{\Gamma \vdash \texttt{(}\overline{e}\texttt{)} : \tau_n}\ \text{[T-Paren]}$$

$$\frac{\forall i.\Gamma \vdash e_i : \tau <: \texttt{bits32}}{\Gamma \vdash (e_1 \ \mathit{aop}\ e_2) \texttt{ | } \texttt{0} : \texttt{int32}}\ \text{[T-IArith]} \qquad \frac{\forall i.\Gamma \vdash e_i : \tau <: \texttt{bits32}}{\Gamma \vdash (e_1 \ \mathit{aop}\ e_2) \texttt{ >>> } \texttt{0} : \texttt{uint32}}\ \text{[T-UArith]}$$

$$\frac{\forall i.\Gamma \vdash e_i : \tau_i <: \texttt{float64}}{\Gamma \vdash e_1 \ \mathit{aop}\ e_2 : \texttt{float64}}\ \text{[T-FArith]}$$

Expression checking (cont'd) $\boxed{\Gamma \vdash e : \tau}$

$$\frac{\forall i. \Gamma \vdash e_i : \texttt{int32}}{\Gamma \vdash (e_1 \texttt{ / } e_2) \texttt{ | } \texttt{0} : \texttt{int32}} \; [\text{T-IDiv}]$$

$$\frac{\Gamma(f) = \texttt{floor} \quad \forall i. \Gamma \vdash e_i : \texttt{uint32}}{\Gamma \vdash f(e_1 \texttt{ / } e_2) : \texttt{uint32}} \; [\text{T-UDiv}]$$

$$\frac{\forall i. \Gamma \vdash e_i : \tau_i <: \texttt{float64}}{\Gamma \vdash e_1 \texttt{ / } e_2 : \texttt{float64}} \; [\text{T-FDiv}]$$

$$\frac{\forall i. \Gamma \vdash e_i : \texttt{int32}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{int32}} \; [\text{T-IMod}]$$

$$\frac{\forall i. \Gamma \vdash e_i : \texttt{uint32}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{uint32}} \; [\text{T-UMod}]$$

$$\frac{\forall i. \Gamma \vdash e_i : \tau_i <: \texttt{float64}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{float64}} \; [\text{T-FMod}]$$

$$\frac{\forall i. \Gamma \vdash e_i : \tau <: \texttt{float64}}{\Gamma \vdash e_1 \; \textit{relop} \; e_2 : \texttt{bits1}} \; [\text{T-Rel}]$$

$$\frac{\forall i. \Gamma \vdash e_i : \tau_i <: \texttt{bits32}}{\Gamma \vdash e_1 \; \textit{bop} \; e_2 : \texttt{int32}} \; [\text{T-Bitwise}]$$

$$\frac{\Gamma \vdash e : \tau <: \texttt{bits32}}{\Gamma \vdash \texttt{\~{}}e : \texttt{int32}} \; [\text{T-BitwiseNot}]$$

$$\frac{\Gamma \vdash e : \tau <: \texttt{boolish}}{\Gamma \vdash \texttt{!}e : \texttt{bits1}} \; [\text{T-Not}]$$

$$\frac{\Gamma \vdash e : \tau <: \texttt{bits32}}{\Gamma \vdash \texttt{-}e : \texttt{int32}} \; [\text{T-Negate}]$$