asm.js

Dave Herman, Luke Wagner, and Alon Zakai

November 4, 2012

1 Abstract syntax

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

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

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

$$
\mathit{ss}\ ::=\ \overline{s}
$$

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

$$\kappa_X ::= \texttt{~~}X \mid \texttt{+}X$$

$$
\begin{array}{rcl}
v & ::= & r \mid n \\
\\
e & ::= & v \\
& \mid & \mathit{lval} \\
& \mid & \mathit{lval}\ \texttt{=}\ e \\
& \mid & f\texttt{(}\overline{e}\texttt{)} \\
& \mid & \mathit{unop}\ e \\
& \mid & e\ \mathit{binop}\ e \\
& \mid & e\ \texttt{?}\ e\ \texttt{:}\ e \\
& \mid & \texttt{(}\overline{e}\texttt{)} \\
\\
\mathit{unop} & ::= & \texttt{+} \mid \texttt{~} \mid \texttt{!} \\
\\
\mathit{binop} & ::= & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%} \\
& \mid & \texttt{|} \mid \texttt{\&} \mid \texttt{^} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \\
& \mid & \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{!=} \mid \texttt{==} \\
\mathit{lval} & ::= & x \mid x\texttt{[(}e\ \texttt{\&}\ m\texttt{)}\ \texttt{>>}\ n\texttt{]}
\end{array}
$$

2 Type rules

$$
\begin{array}{rcl}
\sigma, \tau & ::= & \texttt{bit} \mid \texttt{int} \mid \texttt{boolish} \\
& \mid & \texttt{signed} \mid \texttt{unsigned} \\
& \mid & \texttt{double} \\
& \mid & \texttt{array}^n_\tau \mid \texttt{function} \mid \texttt{unknown} \mid \texttt{jsval} \\
& \mid & \texttt{coercible} \\
& \mid & \texttt{imul} \\
& \mid & ((\overline{\sigma}) \to \tau) \wedge \ldots \wedge ((\overline{\sigma'}) \to \tau') \\
& \mid & \texttt{void}
\end{array}
$$

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

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

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

$$
\begin{aligned}
\mathit{type}(\texttt{\char`~\char`~}X) &= \texttt{int} \\
\mathit{type}(\texttt{+}X) &= \texttt{double} \\
\mathit{type}(n) &= \texttt{int} \\
\mathit{type}(r) &= \texttt{double}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{signed}, \texttt{unsigned} &<: \texttt{int}, \texttt{jsval} \\
\texttt{bit}, \texttt{int} &<: \texttt{boolish} \\
\texttt{void}, \texttt{double}, \texttt{array}^{n}_{\tau}, \texttt{function}, \texttt{unknown} &<: \texttt{jsval} \\
\texttt{unknown}, \texttt{int} &<: \texttt{coercible} \\
((\overline{\sigma}) \to \tau) \wedge \ldots \wedge ((\overline{\sigma'}) \to \tau') &<: \texttt{function} \\
((\overline{\sigma}_1) \to \tau_1) \wedge \ldots \wedge ((\overline{\sigma}_n) \to \tau_n) &<: ((\overline{\sigma}_1) \to \tau_1) \wedge \ldots \wedge ((\overline{\sigma}_{n-1}) \to \tau_{n-1}) \\
\texttt{imul} &<: (\texttt{coercible}, \texttt{coercible}) \to \texttt{signed}
\end{aligned}
$$

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

$$
\begin{aligned}
M(\texttt{imul}) &: \texttt{imul} \\
M(\texttt{ceil}), M(\texttt{sin}), M(\texttt{cos}) &: (\texttt{double}) \to \texttt{double}
\end{aligned}
$$

$$
\begin{aligned}
A(\texttt{Uint8Array}), A(\texttt{Int8Array}) &= \texttt{array}^{8}_{\texttt{int}} \\
A(\texttt{Uint16Array}), A(\texttt{Int16Array}) &= \texttt{array}^{16}_{\texttt{int}} \\
A(\texttt{Uint32Array}), A(\texttt{Int32Array}) &= \texttt{array}^{32}_{\texttt{int}} \\
A(\texttt{Float32Array}) &= \texttt{array}^{32}_{\texttt{double}} \\
A(\texttt{Float64Array}) &= \texttt{array}^{64}_{\texttt{double}}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{+}, \texttt{-} &: \quad (\texttt{double}, \texttt{double}) \to \texttt{double} \\
&\quad \wedge\ (\texttt{int}, \texttt{int}) \to \texttt{coercible} \\
\texttt{*} &: \quad (\texttt{double}, \texttt{double}) \to \texttt{double} \\
\texttt{/}, \texttt{\%} &: \quad (\texttt{double}, \texttt{double}) \to \texttt{double} \\
&\quad \wedge\ (\texttt{signed}, \texttt{signed}) \to \texttt{coercible} \\
&\quad \wedge\ (\texttt{unsigned}, \texttt{unsigned}) \to \texttt{coercible}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{|}, \texttt{\&}, \texttt{\char`^}, \texttt{<<}, \texttt{>>} &: \quad (\texttt{coercible}, \texttt{coercible}) \to \texttt{signed} \\
\texttt{>>>} &: \quad (\texttt{coercible}, \texttt{coercible}) \to \texttt{unsigned}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{<}, \texttt{<=}, \texttt{>}, \texttt{>=}, \texttt{==}, \texttt{!=} &: \quad (\texttt{signed}, \texttt{signed}) \to \texttt{bit} \\
&\quad \wedge\ (\texttt{unsigned}, \texttt{unsigned}) \to \texttt{bit} \\
&\quad \wedge\ (\texttt{double}, \texttt{double}) \to \texttt{bit}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{+} &: \quad (\texttt{coercible}) \to \texttt{double} \\
\texttt{\char`~} &: \quad (\texttt{coercible}) \to \texttt{signed} \\
\texttt{!} &: \quad (\texttt{boolish}) \to \texttt{bit}
\end{aligned}
$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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]
$$\frac{\begin{array}{c}\Gamma \vdash e : \sigma \\ \forall i. \Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\varepsilon \\ \varepsilon \neq \mathsf{return} \vee \exists i. \varepsilon_i \cup \emptyset \neq \emptyset \\ \varepsilon' = (\varepsilon \cup \bigcup_i \varepsilon_i) - (L \cup \{\epsilon\})\end{array}}{\Gamma; L \vdash \texttt{switch (}e\texttt{) \{}\ \ \overline{c}\ cd\ \ \texttt{\}} : \tau/\varepsilon'}$$

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

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

$$\frac{}{\Gamma \vdash n : \texttt{int}}\ [\textsc{T-Int}] \qquad \frac{}{\Gamma \vdash r : \texttt{double}}\ [\textsc{T-Double}]$$

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

$$\frac{m = 2^k - 1 \qquad \Gamma(x) = \texttt{array}_\tau^n \qquad \Gamma \vdash e : \texttt{int}}{\Gamma \vdash x\texttt{[(}e \texttt{ \& } m\texttt{) >> } n/8\texttt{]} : \tau}\ [\textsc{T-Load}]$$

$$\frac{m = 2^k - 1 \qquad \Gamma(x) = \texttt{array}_\tau^n \qquad \Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash x\texttt{[(}e_1 \texttt{ \& } m\texttt{) >> } n/8\texttt{] = } e_2 : \tau}\ [\textsc{T-Store}]$$

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

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

$$\frac{\mathit{unop} : \_ \wedge (\sigma) \to \tau \wedge \_ \qquad \Gamma \vdash e : \sigma}{\Gamma \vdash \mathit{unop}\ e : \tau}\ [\textsc{T-Unop}] \qquad \frac{\mathit{binop} : \_ \wedge (\sigma_1, \sigma_2) \to \tau \wedge \_ \qquad \Gamma \vdash e_1 : \sigma_1 \qquad \Gamma \vdash e_2 : \sigma_2}{\Gamma \vdash e_1\ \mathit{binop}\ e_2 : \tau}\ [\textsc{T-Binop}]$$