asm.js

Dave Herman, Luke Wagner, and Alon Zakai

October 26, 2012

1 Abstract syntax

$$b, e, f, g, x, y, z \in \mathit{Identifier}$$

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

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

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

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

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

$$
\begin{array}{rcl}
v & ::= & \kappa_{num} \\
& | & \texttt{[}\kappa_{num}\texttt{,}\kappa_{num}\texttt{]}
\end{array}
$$

$$
\begin{array}{rcl}
e & ::= & \kappa_{num} \\
& | & \kappa_e \\
& | & lval \\
& | & lval \texttt{ = } e \\
& | & f\texttt{(}\overline{e}\texttt{)} \\
& | & unop\ e \\
& | & e\ aop\ e \\
& | & e \texttt{ / } e \\
& | & e \texttt{ \% } e \\
& | & e\ bop\ e \\
& | & e\ relop\ e \\
& | & e \texttt{ ? } e \texttt{ : } e \\
& | & \texttt{(}\overline{e}\texttt{)} \\
& | & \texttt{[}e\texttt{,}e\texttt{]} \\
& | & e\texttt{[0]} \\
& | & e\texttt{[1]}
\end{array}
$$

$$
\begin{array}{rcl}
unop & ::= & \texttt{~} \mid \texttt{-} \mid \texttt{!}
\end{array}
$$

$$
\begin{array}{rcl}
aop & ::= & \texttt{+} \mid \texttt{-} \mid \texttt{*} \\
bop & ::= & \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>}
\end{array}
$$

$$
\begin{array}{rcl}
relop & ::= & \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{!=} \mid \texttt{==}
\end{array}
$$

$$
\begin{array}{rcl}
lval & ::= & x \\
& | & x\texttt{[}e\texttt{]}
\end{array}
$$

2 Type rules

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

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

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

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

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

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

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

$$
\begin{aligned}
M(\texttt{floor}) &= \texttt{floor} \\
M(\texttt{ceil}) &= (\texttt{float64}) \to \texttt{float64} \\
M(\texttt{sin}) &= (\texttt{float64}) \to \texttt{float64} \\
M(\texttt{cos}) &= (\texttt{float64}) \to \texttt{float64} \\
&\ldots
\end{aligned}
$$

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

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

[T-Program]

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

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]

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

[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]

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

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

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

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

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

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

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

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

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

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

$$\frac{\forall i \in \{1,2,3\}.\Gamma \vdash e_i : \sigma_i \quad \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \quad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})}{\Gamma; L \vdash \texttt{for (}e_1\texttt{;}\ e_2\texttt{;}\ e_3\texttt{)}\ s : \tau/\varepsilon'}\ [\text{T-For}]$$

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

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

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

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

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

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

[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]
$$\frac{\Gamma \vdash e : \sigma \quad \Gamma; L \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \texttt{case}\ e\texttt{:}\ ss : \sigma, \tau/\varepsilon}$$

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

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

$$\frac{}{\Gamma \vdash \kappa_{num} : type(\kappa_{num})}\ [\text{T-Number}] \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \kappa_e : type(\kappa_e)}\ [\text{T-Cast}]$$

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

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

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

$$\frac{\Gamma \vdash e_1 : \sigma \qquad \sigma <: \texttt{boolish} \qquad \forall i \in \{2,3\}.\Gamma \vdash e_i : \tau}{\Gamma \vdash e_1 \texttt{ ? } e_2 \texttt{ : } e_3 : \tau}\ [\text{T-Conditional}] \qquad \frac{\forall i \leq n.\Gamma \vdash e_i : \tau_i}{\Gamma \vdash \texttt{(}\overline{e}\texttt{)} : \tau_n}\ [\text{T-Paren}]$$

$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau \qquad \tau <: \texttt{bits32}}{\Gamma \vdash (e_1\ aop\ e_2)\ \texttt{|}\ \texttt{0} : \texttt{int32}}\ [\text{T-IArith}] \qquad \frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau \qquad \tau <: \texttt{bits32}}{\Gamma \vdash (e_1\ aop\ e_2)\ \texttt{>>>}\ \texttt{0} : \texttt{uint32}}\ [\text{T-UArith}]$$

$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau_i \qquad \forall i.\tau_i <: \texttt{float64}}{\Gamma \vdash e_1\ aop\ e_2 : \texttt{float64}}\ [\text{T-FArith}]$$

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

[T-IDiv]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \texttt{int32}}{\Gamma \vdash (e_1 \texttt{ / } e_2) \texttt{ | } 0 : \texttt{int32}}$$

[T-UDiv]
$$\frac{\Gamma(f) = \texttt{floor} \quad \forall i \in \{1,2\}.\Gamma \vdash e_i : \texttt{uint32}}{\Gamma \vdash f\texttt{(}e_1 \texttt{ / } e_2\texttt{)} : \texttt{uint32}}$$

[T-FDiv]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau_i \quad \forall i.\tau_i <: \texttt{float64}}{\Gamma \vdash e_1 \texttt{ / } e_2 : \texttt{float64}}$$

[T-IMod]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \texttt{int32}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{int32}}$$

[T-UMod]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \texttt{uint32}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{uint32}}$$

[T-FMod]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau_i \quad \forall i.\tau_i <: \texttt{float64}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{float64}}$$

[T-Rel]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau \quad \tau <: \texttt{float64}}{\Gamma \vdash e_1 \ \textit{relop}\ e_2 : \texttt{bits1}}$$

[T-Bitwise]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau_i \quad \forall i.\tau_i <: \texttt{bits32}}{\Gamma \vdash e_1 \ \textit{bop}\ e_2 : \texttt{int32}}$$

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

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

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