

asm.js

Dave Herman, Luke Wagner, and Alon Zakai

October 26, 2012

1 Abstract syntax

$a, b, c, d, e, f, g, x, y, z \in Identifier$

```
 $P ::= \text{function}(b, e) \{ \overline{imp_x} \overline{fn_f} \exp \}$   
 $imp_x ::= \text{var } x = e.y;$   
          |  $\text{var } x = e.y(b);$   
          |  $\text{var } x = \text{new } e.y(b);$   
 $exp ::= \text{return } f;$   
        |  $\text{return } \{ \overline{x:f} \};$   
 $fn_f ::= \text{function } f(\overline{x}) \{ \overline{x = \kappa_x}; \text{var } \overline{y = v}; ss \}$ 
```

```
 $s ::= \{ ss \}$   
      |  $e;$   
      |  $\text{if } (e) \ s$   
      |  $\text{if } (e) \ s \ \text{else } s$   
      |  $\text{return } v;$   
      |  $\text{while } (e) \ s$   
      |  $\text{do } s \ \text{while } (e);$   
      |  $\text{for } (e; e; e) \ s$   
      |  $\text{switch } (e) \{ \overline{c} \}$   
      |  $\text{switch } (e) \{ \overline{c} \ d \}$   
      |  $\text{break};$   
      |  $\text{break } lab;$   
      |  $\text{continue};$   
      |  $\text{continue } lab;$   
      |  $lab: s$ 
```

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

```
 $c ::= \text{case } e: ss$   
 $d ::= \text{default}: ss$   
 $cd ::= c \mid d$ 
```

$$\begin{array}{lcl} \kappa_X & ::= & X \mid 0 \\ & | & X \ggg 0 \\ & | & +X \\ & | & [X[0] \ggg 0, X[1] \mid 0] \\ & | & [X[0] \ggg 0, X[1] \ggg 0] \end{array}$$

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

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

$$unop ::= \sim \mid - \mid !$$

$$aop ::= + \mid - \mid *$$

$$bop ::= \mid \mid \& \mid \sim \mid << \mid >> \mid \ggg$$

$$relop ::= < \mid <= \mid > \mid >= \mid != \mid ==$$

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

2 Type rules

$$\begin{array}{lcl} \sigma, \tau & ::= & \text{bits1} \mid \text{bits32} \mid \text{boolish} \\ & & \mid \text{int32} \mid \text{uint32} \\ & & \mid \text{int64} \mid \text{uint64} \\ & & \mid \text{float64} \\ & & \mid \text{array}_\tau \mid \text{function} \mid \text{jsval} \\ & & \mid \text{floor} \\ & & \mid (\bar{\sigma}) \rightarrow \tau \end{array}$$

$$\begin{array}{lcl} \ell & ::= & \text{lab} \mid \epsilon \\ L & ::= & \{\ell\} \\ \varepsilon & ::= & L \mid \text{return} \end{array}$$

$$\begin{array}{lcl} L ; L' & = & L \cup L' \\ \emptyset ; \text{return} & = & \text{return} \\ \{\ell, \bar{\ell}'\} ; \text{return} & = & \{\ell, \bar{\ell}'\} \\ \text{return} ; L & = & \text{return} \end{array}$$

$$\begin{array}{lcl} L \cup \text{return} & = & L \\ \text{return} \cup L & = & L \\ \text{return} \cup \text{return} & = & \text{return} \end{array}$$

$$\begin{array}{lcl} \text{type}(X \mid 0) & = & \text{int32} \\ \text{type}(X \ggg 0) & = & \text{uint32} \\ \text{type}(+X) & = & \text{float64} \\ \text{type}([X[0] \ggg 0, X[1] \mid 0]) & = & \text{int64} \\ \text{type}([X[0] \ggg 0, X[1] \ggg 0]) & = & \text{uint64} \end{array}$$

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

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

```

M(floor)  = floor
M(ceil)   = (float64) → float64
M(sin)    = (float64) → float64
M(cos)    = (float64) → float64
...

```

```

A(UInt8Array) = uint32
A(UInt16Array) = uint32
A(UInt32Array) = uint32
A(Int8Array)   = int32
A(Int16Array)  = int32
A(Int32Array)  = int32
A(Float32Array) = float64
A(Float64Array) = float64

```

Program checking

$\boxed{\vdash P \text{ ok}}$

$$\frac{\begin{array}{c} \text{[T-PROGRAM]} \\ \{\bar{x}\} \cap \{\bar{f}\} = \emptyset \quad \{\bar{x}\} \cap \{b, e\} = \emptyset \quad \{\bar{f}\} \cap \{b, e\} = \emptyset \\ \forall i. b; e; \Gamma_0 \vdash \text{imp}_x \text{ ok} \\ \forall i. \Gamma_0, \Gamma_1 \vdash fn_f \text{ ok} \\ \forall i. \Gamma_0, \Gamma_1 \vdash r \text{ ok} \end{array}}{\vdash \text{function}(b, e) \{ \overline{\text{imp}_x} \overline{fn_f} \exp \} \text{ ok}}$$

Import checking

$\boxed{b; e; \Gamma \vdash \text{imp} \text{ ok}}$

$$\begin{array}{c} \text{[T-IMPORTSTD]} \\ \frac{\Gamma(x) = M(y)}{b; e; \Gamma \vdash \text{var } x = e.y; \text{ ok}} \end{array} \quad \begin{array}{c} \text{[T-IMPORTFFI]} \\ \frac{y \notin \text{dom}(M)}{b; e; \Gamma \vdash \text{var } x = e.y; \text{ ok}} \end{array}$$

$$\begin{array}{c} \text{[T-VIEW]} \\ \frac{\Gamma(x) = \text{array}_{A(y)}}{b; e; \Gamma \vdash \text{var } x = e.y(b); \text{ ok}} \end{array} \quad \begin{array}{c} \text{[T-NEWVIEW]} \\ \frac{\Gamma(x) = \text{array}_{A(y)}}{b; e; \Gamma \vdash \text{var } x = \text{new } e.y(b); \text{ ok}} \end{array}$$

Function checking

$\boxed{\Gamma \vdash fn \text{ ok}}$

$$\frac{\begin{array}{c} \text{[T-FUNCTION]} \\ \{\bar{x}\} \cap \{\bar{y}\} = \emptyset \quad \Gamma(f) = (\bar{\sigma}) \rightarrow \tau \quad \bar{\sigma} = \overline{\text{type}(\kappa_x)} \\ \Gamma, \{\bar{x} : \bar{\sigma}, \bar{y} : \text{type}(v)\}; \emptyset \vdash ss : \tau / \text{return} \end{array}}{\Gamma \vdash \text{function } f(\bar{x}) \{ \overline{x = \kappa_x}; \text{var } \overline{y = v}; \overline{ss} \} \text{ ok}}$$

Export checking

$\boxed{\Gamma \vdash \text{exp} \text{ ok}}$

$$\begin{array}{c} \text{[T-SINGLETON]} \\ \frac{\Gamma(f) = (\bar{\sigma}) \rightarrow \tau \quad \tau <: \text{jval}}{\Gamma \vdash \text{return } f; \text{ ok}} \end{array} \quad \begin{array}{c} \text{[T-MODULE]} \\ \frac{\forall f. \Gamma(f) = (\bar{\sigma}) \rightarrow \tau \wedge \tau <: \text{jval}}{\Gamma \vdash \text{return } \{ \overline{x : f} \}; \text{ ok}} \end{array}$$

Statement list checking

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

$$\frac{[\text{T-NoSTATEMENTS}]}{\Gamma; L \vdash \epsilon : \tau/\emptyset} \quad \frac{[\text{T-STATEMENTS}] \quad \forall i. \Gamma; L \vdash s_i : \tau/\varepsilon_i \quad n > 0 \quad \varepsilon = \varepsilon_1 ; \dots ; \varepsilon_n}{\Gamma; L \vdash \bar{s} : \tau/\varepsilon}$$

Statement checking

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

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

$$\frac{[\text{T-IF}] \quad \begin{array}{l} \Gamma \vdash e : \sigma \\ \sigma <: \text{boolish} \\ \Gamma; \emptyset \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon \cup \emptyset \end{array}}{\Gamma; L \vdash \text{if } (e) \ s : \tau/\varepsilon'} \quad \frac{[\text{T-IFELSE}] \quad \begin{array}{l} \Gamma \vdash e : \sigma \\ \sigma <: \text{boolish} \\ \Gamma; \emptyset \vdash s_1 : \tau/\varepsilon_1 \quad \Gamma; \emptyset \vdash s_2 : \tau/\varepsilon_2 \\ \varepsilon = \varepsilon_1 \cup \varepsilon_2 \end{array}}{\Gamma; L \vdash \text{if } (e) \ s_1 \ \text{else} \ s_2 : \tau/\varepsilon}$$

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

$$\frac{[\text{T-WHILE}] \quad \begin{array}{l} \Gamma \vdash e : \sigma \\ \sigma <: \text{boolish} \\ \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \text{while } (e) \ s : \tau/\varepsilon'} \quad \frac{[\text{T-DOWHILE}] \quad \begin{array}{l} \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \Gamma \vdash e : \sigma \\ \sigma <: \text{boolish} \\ \varepsilon' = \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \text{do } s \ \text{while } (e) ; : \tau/\varepsilon'}$$

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

Statement checking (cont'd)

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

[T-BREAK]

$$\frac{\varepsilon = \{\epsilon\}}{\Gamma; L \vdash \mathbf{break}; : \tau/\varepsilon}$$

[T-BREAKLABEL]

$$\frac{\varepsilon = \{lab\}}{\Gamma; L \vdash \mathbf{break} \quad lab; : \tau/\varepsilon}$$

[T-CONTINUE]

$$\overline{\Gamma; L \vdash \mathbf{continue}; : \tau/\emptyset}$$

[T-CONTINUELABEL]

$$\overline{\Gamma; L \vdash \mathbf{continue} \quad lab; : \tau/\emptyset}$$

[T-LABEL]

$$\frac{\begin{array}{l} \Gamma; L \cup \{lab\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon - (L \cup \{lab\}) \end{array}}{\Gamma; L \vdash lab : s : \tau/\varepsilon'}$$

[T-SWITCH]

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

[T-SWITCHRETURN]

$$\frac{\begin{array}{l} \Gamma \vdash e : \sigma \\ \forall i. \Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \forall i. \varepsilon_i \cup \emptyset = \emptyset \\ \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\mathbf{return} \end{array}}{\Gamma; L \vdash \mathbf{switch} \ (e) \ \{ \ \bar{c} \ cd \ } : \tau/\mathbf{return}}$$

Case checking

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

[T-CASE]

$$\frac{\begin{array}{l} \Gamma \vdash e : \sigma \\ \Gamma; L \vdash ss : \tau/\varepsilon \end{array}}{\Gamma; L \vdash \mathbf{case} \ e : ss : \sigma, \tau/\varepsilon}$$

[T-DEFAULT]

$$\frac{\Gamma; L \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \mathbf{default} : ss : \sigma, \tau/\varepsilon}$$

Expression checking

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

$$\begin{array}{c}
\text{[T-NUMBER]} \quad \frac{}{\Gamma \vdash \kappa_{num} : type(\kappa_{num})} \qquad \text{[T-CAST]} \quad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash \kappa_e : type(\kappa_e)} \\
\\
\text{[T-VARREF]} \quad \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \text{[T-ASSIGN]} \quad \frac{\Gamma(x) = \tau \quad \Gamma \vdash e : \tau}{\Gamma \vdash x = e : \tau} \\
\\
\text{[T-LOAD]} \quad \frac{\Gamma(x) = \mathbf{array}_\tau \quad \Gamma \vdash e : \mathbf{uint32}}{\Gamma \vdash x[e] : \tau} \qquad \text{[T-STORE]} \quad \frac{\Gamma \vdash e_1 : \mathbf{uint32} \quad \Gamma \vdash e_2 : \tau \quad \tau <: \Gamma(x)}{\Gamma \vdash x[e_1] = e_2 : \tau} \\
\\
\text{[T-FUNCALL]} \quad \frac{\Gamma(f) = (\bar{\sigma}) \rightarrow \tau \quad \forall i. \Gamma \vdash e_i : \sigma_i}{\Gamma \vdash f(\bar{e}) : \tau} \qquad \text{[T-FFI]} \quad \frac{\Gamma(f) = \mathbf{function} \quad \forall i. \Gamma \vdash e_i : \sigma_i \wedge \sigma_i <: \mathbf{jsval}}{\Gamma \vdash f(\bar{e}) : \mathbf{jsval}} \\
\\
\text{[T-CONDITIONAL]} \quad \frac{\Gamma \vdash e_1 : \sigma \quad \sigma <: \mathbf{boolish} \quad \forall i \in \{2, 3\}. \Gamma \vdash e_i : \tau}{\Gamma \vdash e_1 ? e_2 : e_3 : \tau} \qquad \text{[T-PAREN]} \quad \frac{\forall i \leq n. \Gamma \vdash e_i : \tau_i}{\Gamma \vdash (\bar{e}) : \tau_n} \\
\\
\text{[T-IARITH]} \quad \frac{\forall i \in \{1, 2\}. \Gamma \vdash e_i : \tau \quad \tau <: \mathbf{bits32}}{\Gamma \vdash (e_1 \mathit{ aop } e_2) \mid 0 : \mathbf{int32}} \qquad \text{[T-UARITH]} \quad \frac{\forall i \in \{1, 2\}. \Gamma \vdash e_i : \tau \quad \tau <: \mathbf{bits32}}{\Gamma \vdash (e_1 \mathit{ aop } e_2) >>> 0 : \mathbf{uint32}} \\
\\
\text{[T-FARITH]} \quad \frac{\forall i \in \{1, 2\}. \Gamma \vdash e_i : \tau_i \quad \forall i. \tau_i <: \mathbf{float64}}{\Gamma \vdash e_1 \mathit{ aop } e_2 : \mathbf{float64}}
\end{array}$$

Expression checking (cont'd)

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

$$\begin{array}{c}
 \text{[T-IDiv]} \\
 \frac{\forall i \in \{1, 2\}. \Gamma \vdash e_i : \text{int32}}{\Gamma \vdash (e_1 / e_2) \mid 0 : \text{int32}}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[T-UDiv]} \\
 \frac{\Gamma(f) = \text{floor} \quad \forall i \in \{1, 2\}. \Gamma \vdash e_i : \text{uint32}}{\Gamma \vdash f(e_1 / e_2) : \text{uint32}}
 \end{array}$$

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

$$\begin{array}{c}
 \text{[T-IMod]} \\
 \frac{\forall i \in \{1, 2\}. \Gamma \vdash e_i : \text{int32}}{\Gamma \vdash e_1 \% e_2 : \text{int32}}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[T-UMod]} \\
 \frac{\forall i \in \{1, 2\}. \Gamma \vdash e_i : \text{uint32}}{\Gamma \vdash e_1 \% e_2 : \text{uint32}}
 \end{array}$$

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

$$\begin{array}{c}
 \text{[T-REL]} \\
 \frac{\forall i \in \{1, 2\}. \Gamma \vdash e_i : \tau \quad \tau <: \text{float64}}{\Gamma \vdash e_1 \text{ relop } e_2 : \text{bits1}}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[T-BITWISE]} \\
 \frac{\forall i \in \{1, 2\}. \Gamma \vdash e_i : \tau_i \quad \forall i. \tau_i <: \text{bits32}}{\Gamma \vdash e_1 \text{ bop } e_2 : \text{int32}}
 \end{array}$$

$$\begin{array}{c}
 \text{[T-BITWISENOT]} \\
 \frac{\Gamma \vdash e : \tau \quad \tau <: \text{bits32}}{\Gamma \vdash \sim e : \text{int32}}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[T-NOT]} \\
 \frac{\Gamma \vdash e : \tau \quad \tau <: \text{boolish}}{\Gamma \vdash !e : \text{bits1}}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{[T-NEGATE]} \\
 \frac{\Gamma \vdash e : \tau \quad \tau <: \text{bits32}}{\Gamma \vdash -e : \text{int32}}
 \end{array}$$