`capsule.js`

Dave Herman

October 17, 2012

1 Abstract syntax

$$
\begin{array}{rcl}
\rho & ::= & \textit{runtime lexical environment} \\
\Delta & ::= & \{\overline{f : (\overline{\sigma}) \to \tau}\} \\
\Gamma & ::= & \{\overline{x : \tau}\}
\end{array}
$$

$$
\begin{array}{rcl}
\textit{capsule} & ::= & \texttt{function } f(\overline{x}) \texttt{ \{ } \overline{\textit{fn}} \texttt{ return } f\texttt{; \}} \\
& | & \texttt{function } f(\overline{x}) \texttt{ \{ } \overline{\textit{fn}} \texttt{ return \{ } \overline{x\texttt{:}f} \texttt{ \}; \}} \\
\textit{fn} & ::= & \texttt{function } f(\overline{x}) \texttt{ \{ } \overline{x \texttt{ = } \kappa_x\texttt{;}} \texttt{ var } \overline{y \texttt{ = } v}\texttt{; } \textit{ss} \texttt{ \}}
\end{array}
$$

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

$$
\begin{array}{rcl}
\kappa_X & ::= & X\ \texttt{| 0} \\
& | & X\ \texttt{\& 0xff} \\
& | & X\ \texttt{\& 0xffff} \\
& | & X\ \texttt{>>> 0} \\
& | & \texttt{+}X \\
& | & X\ \texttt{+ ''} \\
& | & X\ \texttt{+ ""} \\
& | & \texttt{!!}X
\end{array}
$$

$$
\begin{array}{rcl}
v & ::= & \kappa_{num} \\
& | & bool \\
& | & str \\
& | & \texttt{null}
\end{array}
$$

$$
\begin{array}{rcl}
e & ::= & v \\
& | & lval \\
& | & lval\ \texttt{=}\ e \\
& | & f\texttt{(}\overline{e}\texttt{)} \\
& | & unop\ e \\
& | & e\ binop\ e \\
& | & e\ \texttt{?}\ e\ \texttt{:}\ e \\
& | & \texttt{(}\overline{e}\texttt{)}
\end{array}
$$

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

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

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

2 Type rules

$$
\begin{array}{rcl}
\sigma, \tau & ::= & \texttt{boolean} \\
& | & \texttt{uint8} \mid \texttt{uint16} \mid \texttt{uint32} \\
& | & \texttt{int8} \mid \texttt{int16} \mid \texttt{int32} \\
& | & \texttt{float32} \mid \texttt{float64} \\
& | & \texttt{array}_\tau \mid \texttt{Function} \\
& | & \texttt{any} \mid \texttt{undefined} \mid \texttt{null} \mid \texttt{string} \mid \texttt{number} \mid \texttt{object}
\end{array}
$$

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

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

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

$$
\begin{array}{rcl}
type(bool) & = & \texttt{boolean} \\
type(str) & = & \texttt{string} \\
type(null) & = & \texttt{null} \\
type(X\ \texttt{|}\ \texttt{0}) & = & \texttt{int32} \\
type(X\ \texttt{\&}\ \texttt{0xff}) & = & \texttt{uint8} \\
type(X\ \texttt{\&}\ \texttt{0xffff}) & = & \texttt{uint16} \\
type(X\ \texttt{>>>}\ \texttt{0}) & = & \texttt{uint32} \\
type(\texttt{+}X) & = & \texttt{float64} \\
type(X\ \texttt{+}\ \texttt{''}) & = & \texttt{string} \\
type(X\ \texttt{+}\ \texttt{""}) & = & \texttt{string} \\
type(\texttt{!!}X) & = & \texttt{boolean}
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{uint8}, \texttt{uint16}, \texttt{uint32} & <: & \texttt{int} \\
\texttt{int8}, \texttt{int16}, \texttt{int32} & <: & \texttt{int}
\end{array}
$$

Function checking $\boxed{\rho;\Delta \vdash \mathit{fn}\ \mathbf{ok}}$

[T-FUNCTION]

$$\frac{\overline{\forall i.\mathit{type}(\kappa_{x_i}) = \sigma_i} \qquad \rho;\Delta;\{\overline{x:\sigma},\overline{y:\mathit{type}(v)}\};\emptyset \vdash \mathit{ss} : \Delta(f)/\mathsf{return}}{\rho;\Delta \vdash \texttt{function}\ f\texttt{(}\overline{x}\texttt{)}\ \texttt{\{}\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}\ \mathit{ss}\ \texttt{\}}\ \mathbf{ok}}$$

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

[T-STATEMENTS]

$$\frac{\forall i.\rho;\Delta;\Gamma;L \vdash s_i : \tau/\varepsilon_i \qquad \varepsilon = \varepsilon_1\ ;\ \ldots\ ;\ \varepsilon_n}{\rho;\Delta;\Gamma;L \vdash \overline{s} : \tau/\varepsilon}$$

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

[T-BLOCK]

$$\frac{\rho;\Delta;\Gamma;\emptyset \vdash \mathit{ss} : \tau/\varepsilon}{\rho;\Delta;\Gamma;L \vdash \texttt{\{}\ \mathit{ss}\ \texttt{\}} : \tau/\varepsilon}$$

[T-EXPRSTMT]

$$\frac{\rho;\Delta;\Gamma \vdash e : \sigma}{\rho;\Delta;\Gamma;L \vdash e\texttt{;} : \tau/\emptyset}$$

[T-IF]

$$\frac{\rho;\Delta;\Gamma \vdash e : \texttt{boolean} \qquad \rho;\Delta;\Gamma;\emptyset \vdash s : \tau/\varepsilon \qquad \varepsilon' = \varepsilon \cup \emptyset}{\rho;\Delta;\Gamma;L \vdash \texttt{if (}e\texttt{)}\ s : \tau/\varepsilon'}$$

[T-IFELSE]

$$\frac{\rho;\Delta;\Gamma \vdash e : \texttt{boolean} \qquad \rho;\Delta;\Gamma;\emptyset \vdash s_1 : \tau/\varepsilon_1 \qquad \rho;\Delta;\Gamma;\emptyset \vdash s_2 : \tau/\varepsilon_2 \qquad \varepsilon = \varepsilon_1 \cup \varepsilon_2}{\rho;\Delta;\Gamma;L \vdash \texttt{if (}e\texttt{)}\ s_1\ \texttt{else}\ s_2 : \tau/\varepsilon}$$

[T-RETURN]

$$\frac{}{\rho;\Delta;\Gamma;L \vdash \texttt{return}\ v\texttt{;} : \mathit{type}(v)/\mathsf{return}}$$

[T-WHILE]

$$\frac{\rho;\Delta;\Gamma \vdash e : \texttt{boolean} \qquad \rho;\Delta;\Gamma;L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \qquad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})}{\rho;\Delta;\Gamma;L \vdash \texttt{while (}e\texttt{)}\ s : \tau/\varepsilon'}$$

[T-DOWHILE]

$$\frac{\rho;\Delta;\Gamma;L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \qquad \rho;\Delta;\Gamma \vdash e : \texttt{boolean} \qquad \varepsilon' = \varepsilon - (L \cup \{\epsilon\})}{\rho;\Delta;\Gamma;L \vdash \texttt{do}\ s\ \texttt{while (}e\texttt{);} : \tau/\varepsilon'}$$

[T-FOR]

$$\frac{\forall i \in \{1,2,3\}.\rho;\Delta;\Gamma \vdash e_i : \sigma_i \qquad \rho;\Delta;\Gamma;L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \qquad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})}{\rho;\Delta;\Gamma;L \vdash \texttt{for (}e_1\texttt{;}\ e_2\texttt{;}\ e_3\texttt{)}\ s : \tau/\varepsilon'}$$

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

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

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

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

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

[T-Label]
$$\frac{\rho; \Delta; \Gamma; L \cup \{lab\} \vdash s : \tau / \varepsilon}{\rho; \Delta; \Gamma; L \vdash lab\texttt{:}\ s : \tau / \varepsilon}$$

[T-Switch]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e : \sigma \\ \forall i. \rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau / \varepsilon_i \\ \varepsilon = \bigcup_i \varepsilon_i\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{switch (}e\texttt{) \{ }\overline{c}\texttt{ \}} : \tau / \varepsilon}$$

[T-SwitchDefault]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e : \sigma \\ \forall i. \rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau / \varepsilon_i \\ \rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash ss : \tau / \varepsilon \\ \varepsilon \neq \mathsf{return} \vee \exists i. \varepsilon_i \cup \emptyset \neq \emptyset \\ \varepsilon' = \varepsilon \cup \bigcup_i \varepsilon_i\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{switch (}e\texttt{) \{ }\overline{c}\texttt{ default:}\ ss\texttt{ \}} : \tau / \varepsilon'}$$

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

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

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

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

$$\frac{}{\rho; \Delta; \Gamma \vdash v : \mathit{type}(v)}\ \text{[T-Value]}$$

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

$$\frac{\mathit{type}(\rho(x)) = \texttt{array}_\tau \quad \rho; \Delta; \Gamma \vdash e : \sigma \qquad \sigma <: \texttt{int}}{\rho; \Delta; \Gamma \vdash x\texttt{[}e\texttt{]} : \tau}\ \text{[T-Load]} \qquad \frac{\mathit{type}(\rho(x)) = \texttt{array}_\tau \quad \rho; \Delta; \Gamma \vdash e_1 : \sigma \qquad \sigma <: \texttt{int} \quad \rho; \Delta; \Gamma \vdash e_2 : \tau}{\rho; \Delta; \Gamma \vdash x\texttt{[}e_1\texttt{]} \texttt{ = } e_2 : \tau}\ \text{[T-Store]}$$

$$\frac{\Delta(f) = (\overline{\sigma}) \to \tau \quad \forall i.\rho; \Delta; \Gamma \vdash e_i : \sigma_i}{\rho; \Delta; \Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \tau}\ \text{[T-FunCall]} \qquad \frac{f \notin \mathit{dom}(\Delta) \qquad f \notin \mathit{dom}(\Gamma) \quad \mathit{type}(\rho(f)) = \texttt{Function} \quad \forall i.\rho; \Delta; \Gamma \vdash e_i : \sigma_i}{\rho; \Delta; \Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \texttt{any}}\ \text{[T-FFI]}$$

$$\frac{\mathit{type}(\mathit{unop}) = \sigma \to \tau \qquad \rho; \Delta; \Gamma \vdash e : \sigma}{\rho; \Delta; \Gamma \vdash \mathit{unop}\ e : \tau}\ \text{[T-UnaryOp]}$$

$$\frac{\forall i \in \{1, 2\}.\rho; \Delta; \Gamma \vdash e_i : \sigma_i \qquad \mathit{type}(\mathit{binop}) = \sigma_1 \times \sigma_2 \to \tau}{\rho; \Delta; \Gamma \vdash e_1\ \mathit{binop}\ e_2 : \tau}\ \text{[T-BinaryOp]}$$

$$\frac{\rho; \Delta; \Gamma \vdash e_1 : \texttt{boolean} \quad \forall i \in \{2, 3\}.\rho; \Delta; \Gamma \vdash e_i : \tau}{\rho; \Delta; \Gamma \vdash e_1 \texttt{ ? } e_2 \texttt{ : } e_3 : \tau}\ \text{[T-Conditional]}$$

$$\frac{\forall i \leq n.\rho; \Delta; \Gamma \vdash e_i : \tau_i}{\rho; \Delta; \Gamma \vdash \texttt{(}\overline{e}\texttt{)} : \tau_n}\ \text{[T-Paren]}$$