capsule.js

Dave Herman

October 17, 2012

1   Abstract syntax

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

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

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

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

$$
\begin{array}{rcl}
c & ::= & \texttt{case } e{:}\, \textit{ss}
\end{array}
$$

$$\kappa_X \quad ::= \quad X \ \texttt{|}\ \texttt{0}$$
$$\quad | \quad X\ \texttt{\&}\ \texttt{0xff}$$
$$\quad | \quad X\ \texttt{\&}\ \texttt{0xffff}$$
$$\quad | \quad X\ \texttt{>>>}\ \texttt{0}$$
$$\quad | \quad \texttt{+}X$$
$$\quad | \quad X\ \texttt{+}\ \texttt{''}$$
$$\quad | \quad X\ \texttt{+}\ \texttt{""}$$
$$\quad | \quad \texttt{!!}X$$

$$v \quad ::= \quad \kappa_{num}$$
$$\quad | \quad bool$$
$$\quad | \quad str$$
$$\quad | \quad \texttt{null}$$

$$e \quad ::= \quad v$$
$$\quad | \quad lval$$
$$\quad | \quad lval\ \texttt{=}\ e$$
$$\quad | \quad f(\overline{e})$$
$$\quad | \quad unop\ e$$
$$\quad | \quad e\ binop\ e$$
$$\quad | \quad e\ \texttt{?}\ e\ \texttt{:}\ e$$
$$\quad | \quad (\overline{e})$$

$$unop \quad ::= \quad \texttt{void} \mid \texttt{\textasciitilde} \mid \texttt{-} \mid \texttt{!}$$

$$binop \quad ::= \quad \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{==}$$
$$\quad | \quad \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%}$$
$$\quad | \quad \texttt{|} \mid \texttt{\&} \mid \texttt{\textasciicircum}$$
$$\quad | \quad \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>}$$
$$\quad | \quad \texttt{||} \mid \texttt{\&\&}$$

$$lval \quad ::= \quad x$$
$$\quad | \quad x\texttt{[}e\texttt{]}$$

2 Type rules

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

$$\ell \quad ::= \quad lab \mid \epsilon$$
$$L \quad ::= \quad \{\overline{\ell}\}$$
$$\varepsilon \quad ::= \quad L \mid \mathsf{return}$$

$$L \; ; \; L' \quad = \quad L \cup L'$$
$$\emptyset \; ; \; \mathsf{return} \quad = \quad \mathsf{return}$$
$$\{\ell, \overline{\ell'}\} \; ; \; \mathsf{return} \quad = \quad \{\ell, \overline{\ell'}\}$$
$$\mathsf{return} \; ; \; L \quad = \quad \mathsf{return}$$

$$L \cup \mathsf{return} \quad = \quad L$$
$$\mathsf{return} \cup L \quad = \quad L$$
$$\mathsf{return} \cup \mathsf{return} \quad = \quad \mathsf{return}$$

$$type(bool) \quad = \quad \texttt{boolean}$$
$$type(str) \quad = \quad \texttt{string}$$
$$type(null) \quad = \quad \texttt{null}$$
$$type(X \;\&\; \texttt{0xff}) \quad = \quad \texttt{int8}$$
$$type(X \;\&\; \texttt{0xffff}) \quad = \quad \texttt{int16}$$
$$type(X \;\mid\; \texttt{0}) \quad = \quad \texttt{int32}$$
$$type(X \;\texttt{>>>}\; \texttt{0}) \quad = \quad \texttt{uint32}$$
$$type(\texttt{+}X) \quad = \quad \texttt{float64}$$
$$type(X \;\texttt{+}\; \texttt{''}) \quad = \quad \texttt{string}$$
$$type(X \;\texttt{+}\; \texttt{""}) \quad = \quad \texttt{string}$$
$$type(\texttt{!!}X) \quad = \quad \texttt{boolean}$$

$$\texttt{uint8}, \texttt{uint16}, \texttt{uint32} \quad <: \quad \texttt{int}$$
$$\texttt{int8}, \texttt{int16}, \texttt{int32} \quad <: \quad \texttt{int}$$

Function checking $\boxed{\rho; \Delta \vdash \textit{fn } \textbf{ok}}$

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

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

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

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

[T-If]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e : \texttt{boolean} \\ \rho; \Delta; \Gamma; \emptyset \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon \cup \emptyset\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{if } (e)\ s : \tau/\varepsilon'}$$

[T-IfElse]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e : \texttt{boolean} \\ \rho; \Delta; \Gamma; \emptyset \vdash s_1 : \tau/\varepsilon_1 \qquad \rho; \Delta; \Gamma; \emptyset \vdash s_2 : \tau/\varepsilon_2 \\ \varepsilon = \varepsilon_1 \cup \varepsilon_2\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{if } (e)\ s_1 \texttt{ else } s_2 : \tau/\varepsilon}$$

[T-Return]
$$\frac{}{\rho; \Delta; \Gamma; L \vdash \texttt{return } v; : type(v)/\texttt{return}}$$

[T-While]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e : \texttt{boolean} \\ \rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{while } (e)\ s : \tau/\varepsilon'}$$

[T-DoWhile]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \rho; \Delta; \Gamma \vdash e : \texttt{boolean} \\ \varepsilon' = \varepsilon - (L \cup \{\epsilon\})\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{do } s \texttt{ while } (e); : \tau/\varepsilon'}$$

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

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

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

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

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

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

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

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

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

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

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

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

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