May 11, 2016

1 Notation

- The store Σ is a function from variable names to memory addresses.
- A heap μ is a function from memory addresses to objects or resources.
- If f is a function, $f\{a \mapsto b\}$ refers to a new function f' defined as:

$$f'(x) = \left\{ \begin{array}{ll} f(x), & x \neq a \\ b, & x = a \end{array} \right\}$$

- For an expression e, $e[e_1/x_1, ..., e_n/x_n]$ is a new expression with the structure of e, where every free occurrence of x_i is replaced with e_i.
- If f is a function, $f(x) \uparrow$ means that $f(x)$ is undefined. $\uparrow$ on its own represents 'undefined'.
- $\varnothing$ refers to the empty set. The single instance of the empty type is denoted `Unit`.
- A configuration is a triple $\langle \mu, \Sigma, e \rangle$.
- $\langle \mu, \Sigma, e \rangle \longrightarrow \langle \mu', \Sigma', e' \rangle$ if, after one reduction step on e in heap μ and store Σ, the program ends in heap μ' and store Σ', ready to execute e'.
- A program with body e begins execution in the configuration $\langle \lambda x. \uparrow, \lambda x. \uparrow, e \rangle$.
- Terms which are values are members of the set $V = \{l\}$
- A valid program terminates in a finite number of reductions in the configuration $\langle \mu, v \rangle$ for some $v \in V$.
- $\langle \mu_1, e_1 \rangle \longrightarrow_* \langle \mu_2, v \rangle$ if $\langle \mu_2, v \rangle$ is the terminating configuration after repeatedly applying reduction rules (assuming that reduction rules are congruent and that $\langle \mu_1, e_1 \rangle$ terminates).
- $l \mapsto x \Rightarrow \{\overline{\sigma = e}\}$ means that memory address l contains the object $x \Rightarrow \{\overline{\sigma = e}\}$.
- $l \mapsto x \Rightarrow \{\bar{r}\}$ means that memory address l contains the resource $\bar{r}$.

2 Grammar

$$\begin{array}{rll} c ::= & \langle \mu, \varepsilon \rangle & \textit{configurations} \\ \\ e ::= & x & \textit{expressions} \\ | & r & \\ | & \texttt{new}\ x \Rightarrow \overline{\sigma = e} & \\ | & e.m(e) & \\ | & e.\pi(e) & \\ | & l & (\textit{memory address}) \end{array}$$

3 Dynamic Semantics

This first section introduces a basic dynamic semantics that has no notion of an effect.

$$\boxed{\langle \mu, \Sigma, e \rangle \longrightarrow \langle \mu, \Sigma, e \rangle}$$

$$\frac{\Sigma(x) = l}{\langle \mu, \Sigma, x \rangle \longrightarrow \langle \mu, \Sigma, l \rangle} \text{ (E-VAR)} \qquad \frac{\mu(l) \uparrow}{\langle \mu, \texttt{new x} \Rightarrow \overline{\sigma = e}, \Sigma \rangle \longrightarrow \langle \mu\{l \mapsto \texttt{new x} \Rightarrow \overline{\sigma = e}\}, \Sigma, l \rangle} \text{ (E-NEW}_\sigma\text{)}$$

$$\frac{\begin{array}{c} \langle \mu_1, e_1 \rangle \longrightarrow_* \langle \mu_2, l \rangle \qquad \langle \mu_2, e_2 \rangle \longrightarrow_* \langle \mu_3, v \rangle \\ \mu_3(l) = x \Rightarrow \overline{\sigma = e} \qquad \texttt{def m}(y : \tau_1) : \tau_2 \ \texttt{with}\ \varepsilon \in \overline{\sigma = e} \end{array}}{\langle \mu_1, \Sigma, e_1.m(e_2) \rangle \longrightarrow \langle \mu_3, \Sigma\{y \mapsto v\}, e[l/x, v/y] \rangle} \text{ (E-METHCALL)}$$

$$\frac{\langle \mu_1, e_1 \rangle \longrightarrow_* \langle \mu_2, r \rangle \quad \langle \mu_2, e_2 \rangle \longrightarrow_* \langle \mu_3, v \rangle}{\langle \mu, \Sigma, e_1.\pi(e_2) \rangle \longrightarrow \langle \mu, \Sigma, \texttt{Unit} \rangle} \text{ (E-OPERCALL)}$$

4 Dynamic Semantics With Effects

We amend the definition of configuration. A configuration is a quadruple $\langle \mu, \Sigma, e, \varepsilon \rangle$, where ε is an accumulated set of effects (i.e. pairs from $R \times \Pi$) from the computation so far. The code e has the effect (r, π) if, when $\langle \lambda x. \uparrow, \lambda x. \uparrow, e, \varnothing \rangle \longrightarrow_* \langle \mu, \Sigma, e', \varepsilon \rangle$, we have $(r, m) \in \varepsilon$.

This definition could be overstrict. A non-terminating program (or a program which does not terminate in a value) still has effects during execution. However, it might be a useful simplification to narrow our focus to only consider those programs which terminate.

$$\boxed{\langle \mu, \Sigma, e, \varepsilon \rangle \simeq \langle \mu, \Sigma, e, \varepsilon \rangle}$$

$$\frac{\Sigma(x) = l}{\langle \mu, \Sigma, x, \varepsilon \rangle \longrightarrow \langle \mu, \Sigma, l, \varepsilon \rangle} \text{ (E-VAR)} \qquad \frac{\mu(l) \uparrow}{\langle \mu, \texttt{new x} \Rightarrow \overline{\sigma = e}, \Sigma, \varepsilon \rangle \longrightarrow \langle \mu\{l \mapsto \texttt{new x} \Rightarrow \overline{\sigma = e}\}, \Sigma, l, \varepsilon \rangle} \text{ (E-NEW}_\sigma\text{)}$$

$$\frac{\begin{array}{c}\langle \mu_1, \Sigma, e_1, \varepsilon_1 \rangle \longrightarrow_* \langle \mu_2, \Sigma_2, r, \varepsilon_2 \rangle \qquad \langle \mu_2, \Sigma, e_2, \varepsilon_2 \rangle \longrightarrow_* \langle \mu_3, \Sigma_3, v, \varepsilon_3 \rangle \\ \mu_3(l) = x \Rightarrow \overline{\sigma = e} \qquad \texttt{def m}(y : \tau_1) : \tau_2 \texttt{ with } \varepsilon \in \overline{\sigma = e}\end{array}}{\langle \mu_1, \Sigma, e_1.m(e_2), \varepsilon_1 \rangle \longrightarrow \langle \mu_3, \Sigma\{y \mapsto v\}, e[l/x, v/y], \varepsilon_3 \rangle} \text{ (E-METHCALL)}$$

$$\frac{\langle \mu_1, \Sigma, e_1, \varepsilon_1 \rangle \longrightarrow_* \langle \mu_2, \Sigma_2, r, \varepsilon_2 \rangle \quad \langle \mu_2, \Sigma, e_2, \varepsilon_2 \rangle \longrightarrow_* \langle \mu_3, \Sigma_3, v, \varepsilon_3 \rangle}{\langle \mu, \Sigma, e_1.\pi(e_2), \varepsilon_1 \rangle \longrightarrow \langle \mu, \Sigma_3, \texttt{Unit}, \varepsilon_3 \rangle} \text{ (E-OPERCALL)}$$