1 Arrays

Arrays are added at the expression level by means of the following extension:

$$\begin{array}{lll} \mathcal{E} & += & [\,\mathcal{E}^*\,] \\ & & \mathcal{E}\,[\,\mathcal{E}] \\ & & \mathcal{E}\ .\ \text{length} \\ & & \text{elemRef}\ \mathcal{E}\,[\,\mathcal{E}] \end{array}$$

Additional well-formedness rules for the new constructs:

$$\frac{\textbf{Val} \vdash e_1 \ldots \textbf{Val} \vdash e_k}{\textbf{Val} \vdash [\,e_1,\ldots,e_k\,]} \quad \frac{\textbf{Val} \vdash e_1 \ldots \textbf{Val} \vdash e_k}{\textbf{Weak} \vdash [\,e_1,\ldots,e_k\,]} \quad \frac{\textbf{Val} \vdash e_1 \ldots \textbf{Val} \vdash e_k}{\textbf{Void} \vdash \text{ignore}\ \ [\,e_1,\ldots,e_k\,]}$$

$$\frac{\textbf{Val} \vdash e \quad \textbf{Val} \vdash i}{\textbf{Val} \vdash e\,[\,i\,]} \quad \frac{\textbf{Val} \vdash e \quad \textbf{Val} \vdash i}{\textbf{Weak} \vdash e\,[\,i\,]} \quad \frac{\textbf{Val} \vdash e \quad \textbf{Val} \vdash i}{\textbf{Void} \vdash \text{ignore}\ e\,[\,i\,]}$$

$$\frac{\textbf{Val} \vdash e \quad \textbf{Val} \vdash i}{\textbf{Ref} \vdash \text{elemRef}\, e\,[\,i\,]}$$

2 Operational Semantics

An array can be represented as a pair: the length of the array and a mapping from indices to elements. If we denote X the set of elements then the set of all arrays $\mathcal{A}(X)$ can be defined as follows:

$$\mathcal{A}(X) = \mathbb{N} \times (\mathbb{N} \to X)$$

For an array $\langle n, f\rangle$ we assume $\texttt{dom}\, f = [0\,..\,n-1]$. An element selection function:

$$\bullet[\bullet] : \mathcal{A}(X) \to \mathbb{N} \to X$$

$$\langle n, f\rangle\,[i] = \left\{ \begin{array}{ccl} f(i) & , & i < n \\ \perp & , & \text{otherwise} \end{array} \right.$$

We represent arrays by references. Thus, we introduce a (linearly) ordered set of locations

$$\mathcal{L} = \{l_0, l_1, \ldots\}$$

Now, the set of all values the programs operate on can be described as follows:

$$\mathcal{V} = \mathbb{Z} \mid \mathcal{L}$$

To access arrays, we introduce an abstraction of memory:

$$\mathcal{M} = \mathcal{L} \to \mathcal{A}\,(\mathcal{V})$$

We now add two more components to the configurations: a memory function μ and the first free memory location l_m, and define the following primitive

$$\mathbf{mem}\, \langle \sigma, \omega, \mu, l_m \rangle = \mu$$

which gives a memory function from a configuration.

The definition of state does not change, hence all existing rules are preserved (modulo adding additinal components to configurations) The rules for the new kinds of expressions are as follows:

$$\frac{c \xRightarrow[\mathcal{E}^*]{e_0, \dots, e_k} \langle \langle \sigma, \omega, \mu, l \rangle, v_1, \dots, v_k \rangle}{c \xRightarrow[\mathcal{E}]{[e_0, \dots, e_k]} \langle \langle \sigma, \omega, \mu[l \leftarrow \langle k+1, i \mapsto v_i \rangle], l+1 \rangle, l \rangle} \quad [\text{Array}]$$

$$\frac{c \xRightarrow[\mathcal{E}^*]{ei} \langle c', lv \rangle \quad l \in \mathcal{L} \quad v \in \mathbb{Z}}{c \xRightarrow[\mathcal{E}]{e\,[\,i\,]} \langle c', ((\mathbf{mem}\, c')(l))[i] \rangle} \quad [\text{ArrayElem}]$$

$$\frac{c \xRightarrow[\mathcal{E}^*]{ei} \langle c', lv \rangle \quad l \in \mathcal{L} \quad v \in \mathbb{Z}}{c \xRightarrow[\mathcal{E}]{\text{elemRef}\, e\,[\,i\,]} \langle c', \mathbf{elemRef}\, l\, v \rangle} \quad [\text{ArrayElemRef}]$$

$$\frac{c \xRightarrow[\mathcal{E}]{e} \langle c', l \rangle \quad l \in \mathcal{L}}{c \xRightarrow[\mathcal{E}]{e\,.\,\text{length}} \langle c', \mathbf{fst}\, (\mathbf{mem}\, c')(l) \rangle} \quad [\text{ArrayLength}]$$

We also need one additional rule for assignment:

$$\frac{c \xRightarrow[\mathcal{E}^*]{lr} \langle \langle \sigma, \omega, \mu, l \rangle, [\mathbf{elemRef}\, a\, i] v \rangle \quad a \in \mathcal{L}}{c \xRightarrow[\mathcal{E}]{l := r} \langle \langle \sigma, \omega, \mu[a \leftarrow \langle \mathbf{fst}\, \mu(a), (\mathbf{snd}\, \mu(a))[i \leftarrow v] \rangle], l \rangle, v \rangle} \quad [\text{AssignArray}]$$

3 Stack Machine

In stack machine we add the following new instructions:

$$\begin{array}{rcl} I & += & \text{ARRAY}\ \mathbb{N} \\ & & \text{ELEM} \\ & & \text{STA} \end{array}$$

We also add memory function and current location components to the configuration; as state components are preserved, all rules are preserved as well. The new rules are:

$$\frac{P \vdash \langle s[n,..], s_c, \sigma, \omega, \mu[l \leftarrow \langle n, i \mapsto s[n-i-1] \rangle], l+1 \rangle \xrightarrow{p}_{\mathcal{SM}} c}{P \vdash \langle s, s_c, \sigma, \omega, \mu, l \rangle \xrightarrow{[\text{ARRAY } n]p}_{\mathcal{SM}} c} \quad [\text{ARRAY}_{\mathcal{SM}}]$$

$$\frac{P \vdash \langle [(\mu(a))[i]]s, s_c, \sigma, \omega, \mu, l \rangle \xrightarrow{p}_{\mathcal{SM}} c}{P \vdash \langle ias, s_c, \sigma, \omega, \mu, l \rangle \xrightarrow{[\text{ELEM}]p}_{\mathcal{SM}} c} \quad [\text{ELEM}_{\mathcal{SM}}]$$

$$\frac{P \vdash \langle vs, s_c, \sigma, \omega, \mu[a \leftarrow \langle \mathbf{fst}\ (\mu(a)), (\mathbf{snd}\ (\mu(a)))[i \leftarrow v] \rangle], l \rangle \xrightarrow{p}_{\mathcal{SM}} c}{P \vdash \langle vias, s_c, \sigma, \omega, \mu, l \rangle \xrightarrow{[\text{STA}]p}_{\mathcal{SM}} c} \quad [\text{STA}_{\mathcal{SM}}]$$