CS611 Lecture 19: State: variables, pass by reference 10/13/99
David Welte

Denotational semantics for F!

$$\begin{aligned}
\mathcal{C}[e] : (Env \times Store) &\rightarrow (Result \times Store) \\
Result &= (Value + Error)_\bot \\
Store &= Loc \rightarrow Storable \\
Storable &= Value + Unassigned \text{ (If we store Result, it would be a lazy store)} \\
Function &= Value \times Store \rightarrow Result \times Store
\end{aligned}$$

$alloc(\sigma)$ returns some $Loc(l)$ such that $\sigma(l)$ is unassigned.

$$\begin{aligned}
\mathcal{C}[n]\rho\sigma &= \langle n, \sigma\rangle \\
\mathcal{C}[x]\rho\sigma &= \langle \rho(x), \sigma\rangle \\
\mathcal{C}[\triangle e]\rho\sigma &= \text{ case } \mathcal{C}[e]\rho\sigma \text{ of } \langle Loc(l), \sigma'\rangle \Rightarrow \langle \sigma'(l), \sigma'\rangle \\
\mathcal{C}[e_0 := e_1]\rho\sigma &= \text{ case } \mathcal{C}[e_0]\rho\sigma \text{ of } \langle Loc(l), \sigma'\rangle \Rightarrow \text{ case } \mathcal{C}[e_1]\rho\sigma' \text{ of } \langle v, \sigma''\rangle \Rightarrow \langle v, \sigma''[l \mapsto v] \\
\mathcal{C}[\text{ref } e]\rho\sigma &= \text{ case } \mathcal{C}[e]\rho\sigma \text{ of } \langle v, \sigma'\rangle \Rightarrow \text{ let } l = alloc(\sigma') \text{ in } \langle l, \sigma'[l \mapsto v]\rangle \\
\mathcal{C}[\text{call } e_0\ e_1]\rho\sigma &= \text{ case } \mathcal{C}[e_0]\rho\sigma \text{ of } \langle Function(f), \sigma'\rangle \Rightarrow \text{ case } \mathcal{C}[e_1]\rho\sigma' \text{ of } \langle v, \sigma''\rangle \Rightarrow f(v, \sigma'') \\
\mathcal{C}[\text{fn } x\ e]\rho\sigma &= \lambda v\sigma'.\mathcal{C}[e]\rho[x \mapsto v]\sigma' \\
\mathcal{C}[\text{seq } e_0\ e_1]\rho\sigma &= \text{ case } \mathcal{C}[e_0]\rho\sigma \text{ of } \langle v, \sigma'\rangle \Rightarrow \mathcal{C}[e_1]\rho\sigma' \\
\mathcal{C}[\text{if } e_0\ e_1\ e_2]\rho\sigma &= \text{ case } \mathcal{C}[e_0]\rho\sigma \text{ of } \langle true, \sigma'\rangle \Rightarrow \mathcal{C}[e_1]\rho\sigma' | \langle false, \sigma'\rangle \Rightarrow \mathcal{C}[e_2]\rho\sigma' \\
\mathcal{C}[\text{rec } f\ (\text{fn } y\ e)]\rho\sigma &= \text{fix } \lambda f'.\lambda v\sigma'.\mathcal{C}[e]\rho[y \mapsto v, f \mapsto f']\sigma'
\end{aligned}$$

Here is another way to define F! using l-values (assignable) and r-values (unassignable), like in C. It also uses cute C-like notation. We will start with the added syntax.

$$\begin{aligned}
&(\text{malloc } e) \\
&(e_0 = e_1) \\
&(\&e) \\
&(*e)
\end{aligned}$$

Here are the changes to the syntax:

$$\begin{aligned}
\mathcal{L}[e] &: Env \times Store \rightarrow Loc \times Store(*e) \\
\mathcal{R}[e] &: Env \times Store \rightarrow Result \times Store
\end{aligned}$$

$$\begin{aligned}
\mathcal{R}[x]\rho\sigma &= \langle \sigma(\rho(x)), \sigma\rangle \\
\mathcal{L}[x]\rho\sigma &= \langle \rho(x), \sigma\rangle \\
\mathcal{R}[\text{malloc } e]\rho\sigma &= \mathcal{C}[\text{ref } e]\rho\sigma \\
\mathcal{R}[e_0 = e_1]\rho\sigma &= \text{ case } \mathcal{R}[e_1]\rho\sigma \text{ of } \langle v, \sigma'\rangle \Rightarrow \text{ case } \mathcal{L}[e_0]\rho\sigma' \text{ of } \langle Loc(l), \sigma''\rangle \Rightarrow \langle v, \sigma''[l \mapsto v]\rangle
\end{aligned}$$

$$\mathcal{R}[\&e]\rho\sigma = \mathcal{L}[e]\rho\sigma$$
$$\mathcal{L}[*e]\rho\sigma = \mathcal{R}[e]\rho\sigma$$
$$\mathcal{R}[*e]\rho\sigma = \langle \sigma(\mathcal{R}[e]\rho\sigma), \sigma\rangle$$
$$\mathcal{R}[\text{fn } x \ e] = \lambda v\sigma'. \text{ let } l = alloc(\sigma') \text{ in } \mathcal{R}[e]\rho[x \mapsto l]\sigma[l \mapsto v]$$

Next we can impliment call by reference with this syntax:

$$(\text{rfn } x \ e)$$
$$(\text{rcall } e_0 \ e_1)$$

Here are the additions:

$$RFunction = Loc \times Store \to Result \times Store$$

$$\mathcal{R}[\text{rfn } x \ e]\langle \lambda l\sigma'.\mathcal{R}[e]\rho[x \mapsto l]\sigma', \sigma\rangle$$
$$\mathcal{R}[\text{rcall } e_0 \ e_1]\rho\sigma = \ \text{case } \mathcal{R}[e_0]\rho\sigma \text{ of } \langle RFunction(f), \sigma'\rangle \Rightarrow \ \text{case } \mathcal{L}[e_1]\rho\sigma' \text{ of } \langle l, \sigma''\rangle = f(l, \sigma'')$$