ISWIM-like languages

ISWIM (If you See What I Mean) [Peter Landin, "The Next 700 Prog. Languages"]

— an eager functional language extended with references,
as a solution to the aliasing problem of Algol 60.

If imperative features are simply merged from the SIL
into the eager functional language, meaning is lost:

$$(\lambda \mathsf{x}.\, \mathsf{x} := 1)\, 2 \quad \mapsto \quad 2 := 1$$

Even if parameter types are introduced to distinguish
between variable and value parameters,
the meaning of programs is far from obvious — one might expect that

$$\mathsf{mul\_and\_inc} \;\equiv\; \lambda \mathsf{x}.\, \lambda \mathsf{y}.\, (\mathsf{y} := \mathsf{y} * \mathsf{x}\, ;\, \mathsf{x} := \mathsf{x}+1\, ;\, \langle\rangle)$$

$$\mathsf{mul\_and\_inc}' \;\equiv\; \lambda \mathsf{x}.\, \lambda \mathsf{y}.\, (\mathsf{x} := \mathsf{x}+1\, ;\, \mathsf{y} := \mathsf{y} * (\mathsf{x}-1)\, ;\, \langle\rangle)$$

are equivalent, but

$$\langle \mathsf{mul\_and\_inc}\, \mathsf{z}\, \mathsf{z},\ [\mathsf{z} : 0] \rangle \;\Rightarrow\; \langle \langle\rangle,\ [\mathsf{z} : 1] \rangle$$

$$\langle \mathsf{mul\_and\_inc}'\, \mathsf{z}\, \mathsf{z},\ [\mathsf{z} : 0] \rangle \;\Rightarrow\; \langle \langle\rangle,\ [\mathsf{z} : 0] \rangle$$

References

Mathematical abstraction of memory address with the following signature:

component	specification	(an implementation)
$\mathbf{Rf}$	a countably infinite set of references	$\mathbf{N}$
$\mathcal{R}$	a set of subsets of $\mathbf{Rf}$	$\{0\ \mathsf{to}\ n-1 \mid n \in \mathbf{N}\}$
newref	$\in \mathcal{R} \to \mathbf{Rf}$ such that	newref $(0\ \mathsf{to}\ n-1) = n$
	$\forall R \in \mathcal{R}.$ newref $R \notin \mathcal{R}$ and $R \cup \{\text{newref}\, R\} \in \mathcal{R}$	
Σ	$\bigcup_{R \in \mathcal{R}} (R \to V)$ the set of states	V^*

Extending the Eager Functional Language with References

Syntax:

$$
\begin{array}{lcll}
\text{exp} & ::= & \ldots & \\
& | & \textbf{mkref}\ \mathit{exp} & \text{create an initialized reference} \\
& | & \textbf{val}\ \mathit{exp} & \text{derefence (obtain the current value of a reference)} \\
& | & \mathit{exp}\ \mathbf{:=}\ \mathit{exp} & \text{assign a new value to an existing reference} \\
& | & \mathit{exp} =_R \mathit{exp} & \text{compare references for equality}
\end{array}
$$

Semantics:

- extend the set of values z to include the references r

  (in addition to the canonical forms)

- define an evaluation semantics on configurations of a state and an expression:

$$\langle \sigma,\, e \rangle \Rightarrow \langle z,\, \sigma' \rangle$$

Evaluation Semantics of an EFL with References

$$\frac{\langle \sigma_0, e\rangle \Rightarrow \langle \lambda v.\, \widehat{e},\, \sigma_1\rangle \quad \langle \sigma_1, e'\rangle \Rightarrow \langle z', \sigma_2\rangle \quad \langle \sigma_2,\, (\widehat{e}/v \rightarrow z')\rangle \Rightarrow \langle z, \sigma_3\rangle}{\langle \sigma_0,\, e\, e'\rangle \Rightarrow \langle z,\, \sigma_3\rangle}$$

$$\frac{\langle \sigma, e\rangle \Rightarrow \langle z, \sigma'\rangle}{\langle \sigma,\, \mathbf{mkref}\, e\rangle \Rightarrow \langle r,\, [\sigma' | r : z]\rangle} \text{ where } r = \text{newref}\,(\text{dom}\, \sigma')$$

$$\frac{\langle \sigma, e\rangle \Rightarrow \langle r, \sigma'\rangle}{\langle \sigma,\, \mathbf{val}\, e\rangle \Rightarrow \langle \sigma'\, r,\, \sigma'\rangle}$$

$$\frac{\langle \sigma, e\rangle \Rightarrow \langle r, \sigma'\rangle \quad \langle \sigma', e'\rangle \Rightarrow \langle z', \sigma''\rangle}{\langle \sigma,\, e := e'\rangle \Rightarrow \langle z',\, [\sigma'' | r : z']\rangle}$$

$$\frac{\langle \sigma, e\rangle \Rightarrow \langle r, \sigma'\rangle \quad \langle \sigma', e'\rangle \Rightarrow \langle r', \sigma''\rangle}{\langle \sigma,\, e =_R e'\rangle \Rightarrow \langle \lfloor r = r' \rfloor,\, \sigma''\rangle}$$

Note: It is harder to define small-step semantics

since references are not part of the language,

so e.g. the result of $\mathbf{mkref}\; e$ cannot be expressed as a term.

Continuation Semantics of References

Changes to the continuation semantics of the eager functional language:

- the semantic function has a new argument for the state $\sigma \in \Sigma$
- the continuations also take the state as an argument
- the predomain of values is extended with references

$$
\begin{aligned}
[\![-]\!] &\in exp \to E \to V_{cont} \to \Sigma \to V_* \\
V_{cont} &= V \to \Sigma \to V_* \\
V_* &= (V + \{\mathbf{error}, \mathbf{typeerror}\})_\perp
\end{aligned}
$$

$$
\begin{aligned}
\iota_{norm} &= \lambda z \in V.\, \lambda\sigma \in \Sigma.\, \iota_\uparrow(\iota_0\, z) \\
\mathrm{err} &= \lambda\sigma \in \Sigma.\, \iota_\uparrow(\iota_1\, \mathbf{error}) \\
\mathrm{tyerr} &= \lambda\sigma \in \Sigma.\, \iota_\uparrow(\iota_2\, \mathbf{typeerror})
\end{aligned}
$$

$$
\begin{aligned}
V &\underset{\psi}{\overset{\phi}{\rightleftarrows}} \ldots + V_{fun} + \ldots + V_{ref} \\
V_{fun} &= V \to V_{cont} \to \Sigma \to V_* \\
V_{ref} &= \mathbf{Rf}
\end{aligned}
$$

$$
\begin{aligned}
\iota_{fun} &= \psi \cdot \iota_2 \in V_{fun} \to V \\
\iota_{fun} &= \psi \cdot \iota_6 \in V_{ref} \to V
\end{aligned}
$$

Continuation Semantic Equations: The Pure Segment

A number of language constructs have no effect on the state,
it is passed to their continuation directly:

$$\text{e.g.}\ \ [\![\lfloor n \rfloor]\!]\eta\,\kappa\,\sigma \;=\; \kappa\,(\iota_{int}\,n)\,\sigma$$

$$\text{or equivalently}\ \ [\![\lfloor n \rfloor]\!]\eta\,\kappa \;=\; \kappa\,(\iota_{int}\,n)$$

$$[\![\text{-}e]\!]\eta\,\kappa \;=\; [\![e]\!]\eta\,(\lambda n \in \mathbf{Z}.\,\kappa\,(\iota_{int}\,(-n)))_{int}$$

$$\text{where}\ \ (-)_{int} \in (V_{int} \to \Sigma \to V_*) \to V \to \Sigma \to V_*$$

$$[\![v]\!]\eta\,\kappa \;=\; \kappa\,(\eta\,v)$$

$$[\![\lambda v.\,e]\!]\eta\,\kappa \;=\; \kappa\,(\iota_{fun}\,(\lambda z \in V.\,\lambda\kappa' \in V_{cont}.\,[\![e]\!][\eta\,|\,v : z]\,\kappa'))$$

$$\ldots$$

Continuation Semantic Equations: References

Semantics of the constructs for operations on references:

$$
\begin{aligned}
[\![\mathbf{val}\, e]\!]\eta\,\kappa\,\sigma &= [\![e]\!]\eta\,(\lambda r \in V_{ref}.\, \lambda\sigma' \in \Sigma.\, \kappa\,(\sigma'\, r)\,\sigma')_{ref}\,\sigma \\
\text{i.e. } [\![\mathbf{val}\, e]\!]\eta\,\kappa &= [\![e]\!]\eta\,(\lambda r \in V_{ref}.\, \lambda\sigma' \in \Sigma.\, \kappa\,(\sigma'\, r)\,\sigma')_{ref}
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathbf{mkref}\, e]\!]\eta\,\kappa &= [\![e]\!]\eta\,(\lambda z \in V.\, \lambda\sigma \in \Sigma. \\
&\qquad (\lambda r \in \mathbf{Rf}.\, \kappa\,(\iota_{ref}\, r)\,[\sigma \mid r : z])\,(\mathrm{newref}\,(\mathrm{dom}\,\sigma)))
\end{aligned}
$$

$$
[\![\mathbf{val}\, e]\!]\eta\,\kappa = [\![e]\!]\eta\,(\lambda r \in V_{ref}.\, \lambda\sigma \in \Sigma.\, \kappa\,(\sigma\, r)\,\sigma)_{ref}
$$

$$
[\![e := e']\!]\eta\,\kappa = [\![e]\!]\eta\,(\lambda r \in V_{ref}.\, [\![e']\!]\eta\,(\lambda z \in V.\, \lambda\sigma \in \Sigma.\, \kappa\, z\,[\sigma \mid r : z]))_{ref}
$$

$$
[\![e =_R e']\!]\eta\,\kappa = [\![e]\!]\eta\,(\lambda r \in V_{ref}.\, [\![e']\!]\eta\,(\lambda r' \in V_{ref}.\, \kappa\,(\iota_{bool}\,(r = r')))_{ref})_{ref}
$$

Note: It is not obvious that the references bound to r in the equations for **val** and $:=$ are in the domain of the state σ.