Join Continuations

Consider:

\[
\text{treeprod} = \text{rec} (\lambda f. \lambda \text{tree}.
\]

\[
\text{if (isnat (tree),}
\]

\[
\text{tree,}
\]

\[
f (\text{left (tree)})*f (\text{right (tree)})\)
\]

which multiplies all leaves of a tree, which are numbers.

You can do the “left” and “right” computations concurrently.

C. Varela
Tree Product Behavior

\[ B_{\text{treeprod}} = \]
\[ \text{rec}(\lambda b. \lambda m. \]
\[ \text{seq}(\text{if}(\text{isnat}(\text{tree}(m))), \]
\[ \text{send}(\text{cust}(m), \text{tree}(m)), \]
\[ \text{let newcust} = \text{new}(B_{\text{joincont}}(\text{cust}(m))), \]
\[ \text{lp} = \text{new}(B_{\text{treeprod}}), \]
\[ \text{rp} = \text{new}(B_{\text{treeprod}}) \text{ in} \]
\[ \text{seq}(\text{send}(\text{lp}, \]
\[ \text{pr}(\text{left}(\text{tree}(m)), \text{newcust})), \]
\[ \text{send}(\text{rp}, \]
\[ \text{pr}(\text{right}(\text{tree}(m)), \text{newcust}))), \]
\[ \text{ready}(b))) \]

C. Varela
Tree Product (continued)

\[ B_{\text{joincont}} = \lambda \text{cust.} \lambda \text{firstnum.} \text{ready}(\lambda \text{num.}
\text{seq}(\text{send(cust, firstnum*num)},
\text{ready(sink)))) \]

C. Varela
Sample Execution

f(tree, cust)

f(left(tree), JC)

f(right(tree), JC)

C. Varela
Sample Execution

f(left(tree),JC)

(c)

(d)

C. Varela
Sample Execution

(e)

(f)

C. Varela
Operational Semantics for AMST Actor Language

- Operational semantics of actor model as a labeled transition relationship between actor configurations.

- Actor configurations model open system components:
  - Set of individually named actors
  - Messages “en-route”

C. Varela
Actor Configurations

\[ k = \alpha \parallel \mu \]

\(\alpha\) is a function mapping actor names (represented as free variables) to actor states.

\(\mu\) is a multi-set of messages “en-route.”

C. Varela
Syntactic restrictions on configurations

Given \( A = \text{Dom}(\alpha) \):

- If \( a \) in \( A \), then \( \text{fv}(\alpha(a)) \) is a subset of \( A \).

- If \( <a \leq v> \) in \( \mu \), then \( \{a\} \cup \text{fv}(v) \) is a subset of \( A \).
Reduction contexts and redexes

Consider the expression:

\[ e = \text{send}(\text{new}(b5), a) \]

- The redex \( r \) represents the next sub-expression to evaluate in a left-first call-by-value evaluation strategy.
- The reduction context \( R \) (or \textit{continuation}) is represented as the surrounding expression with a \textit{hole} replacing the redex.

\[
\text{send}(\text{new}(b5), a) = \text{send}(\square, a) \hole \text{new}(b5) \hole \\
e = R \hole r \hole \quad \text{where} \\
R = \text{send}(\square, a) \\
r = \text{new}(b5)
\]

C. Varela
Labeled Transition Relation

\[
\frac{e \rightarrow^{\lambda} e'}{\alpha, [R \upharpoonright e \downharpoonleft]_a \parallel \mu \quad \xrightarrow{\text{[fun:a]}} \quad \alpha, [R \upharpoonright e' \downharpoonleft]_a \parallel \mu}
\]

\[
\alpha, [R \upharpoonright \text{new}(b) \downharpoonleft]_a \parallel \mu \quad \xrightarrow{\text{[new:a,a']}\quad a' \text{ fresh}} \quad \alpha, [R \upharpoonright a' \downharpoonleft]_a, [\text{ready}(b)]_{a'} \parallel \mu
\]

\[
\alpha, [R \upharpoonright \text{send}(a', v) \downharpoonleft]_a \parallel \mu \quad \xrightarrow{\text{[snd:a]}} \quad \alpha, [R \upharpoonright \text{nil} \downharpoonleft]_a \parallel \mu \uplus \{\langle a' \leftarrow v \rangle\}
\]

\[
\alpha, [R \upharpoonright \text{ready}(b) \downharpoonleft]_a \parallel \{\langle a \leftarrow v \rangle\} \uplus \mu \quad \xrightarrow{\text{[rcv:a,v]}} \quad \alpha, [b(v)]_a \parallel \mu
\]

C. Varela
Exercises

• PDCS Exercise 4.6.6 (page 77).

• Using the AMST actor language operational semantics, illustrate the transitions from the following actor configuration:

\[
\kappa_0 = a, [send(new(B_{\text{treeprod}}), pr(pr(5,7),c))]_a \parallel \emptyset
\]

• PDCS Exercise 4.6.7 (page 78).