JOIN CONTINUATIONS

Consider:

\[
\text{tree-prod} = \text{rec}(\lambda p. \lambda \text{tree}. \\
    \text{if (isnat (tree),} \\
    \text{tree,} \\
    f(\text{left(tree)}) * f(\text{right(tree)}) \\
    ))
\]

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

You can do the "left" and "right" computations concurrently.
TREC PRODUCT BEHAVIOR

\[ B_{\text{treeprod}} = \]
rec (Ab, λself, Am.
  seg (become (b(self)),
    if (isnat (tree(m)),
      send (cust(m), tree(m)),
      refactor {newcust := B_{\text{joinout}}
        (cust(m), ∅, nil)}
    )
  )

)}

)},

)}
$B_{\text{joincont}} =$
\[ \text{rec} \ (ab. \ \lambda \text{cust}, \lambda \text{nargs}, \lambda \text{firstnum}, \lambda \text{num} \]
\[ \text{if } (\text{eg } (\text{nargs}, 0), \]
\[ \text{become } (b \ (\text{cust}, 1, \text{num})), \]
\[ \text{seg } (\text{become } (\text{sink}), \]
\[ \text{send } (\text{cust}, \text{firstnum } \# \text{ num})))) \]
SAMPLE EXECUTION

(a) f(tree, cust)

(b) f(left(tree, JC), f(right(tree, JC))}

Cust

Cust

JC
\[ \kappa = \ll \alpha | \mu \gg_{\chi}^{\varrho} \]

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

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

\( \varrho \) is a set of receptionists.

\( \chi \) is a set of external actors.

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

- \( \varrho \subseteq A \), \( A \cap \chi = \emptyset \)
- if \( \alpha(a) = (?a') \), then \( a' \in A \)
- if \( a \in A \), then \( \text{FV}(\alpha(a)) \subseteq \text{AUX} \)
- if \( < v_0 \leftarrow v_1 > \in \mu \), \( \text{FV}(v_0, v_1) \subseteq \text{AUX} \).
\[ x \in x \vdash A \]

\[ A_s = (\forall x) \cup (\forall v) \cup [E] \]

\[ M \in M_w[M] \]

\[ M = \langle v \equiv v \rangle \]

\[ p, x \in P_w[x] \]

\[ \langle a \leftarrow s \rangle \]

\[ V = \text{At} \cup \times \cup \lambda x. E \cup \text{pr}(v, v) \]

\[ E = V \cup \text{app}(E, E) \cup F_n(E^0) \]
**Labeled Transition Relation (\(\rightarrow\))**

\(<\text{fun}: a>\)

\[ e \xrightarrow{\lambda \text{Dom}(d)\cup\{a\}} e' \quad \Rightarrow \\
\langle \alpha, [e]_a \mid M \rangle_x^g \quad \Rightarrow \quad \langle \alpha, [e']_a \mid M \rangle_x^g \]

\(<\text{newactor}: a, a'>\)

\[ \langle \alpha, [R[\text{newactor}(e)]]_a \mid M \rangle_x^g \quad \Rightarrow \\
\langle \alpha, [R[a']]_a, [e]_a' \mid M \rangle_x^g \quad a' \text{ fresh} \]

\(<\text{send}: a, v_0, v_i>\)

\[ \langle \alpha, [R[\text{send}(v_0, v_i)]]_a \mid M \rangle_x^g \quad \Rightarrow \\
\langle \alpha, [R[nil]]_a \mid M, <v_0 \leftarrow v_i> \rangle_x^g \]
<receive: \( V_0, V_i \)>

\[
\ll \alpha, [\text{ready}(V)]_{V_0} \mid V_0 \in V_i, \mu \rr_x^g \to \\
\ll \alpha, [\text{app}(V, V_i)]_{V_0} \mid \mu \rr_x^g
\]

<out: \( V_0, V_i \)>

\[
\ll \alpha | \mu, V_0 \in V_i \rr_x^g \to \ll \alpha | \mu \rr_x^g
\]

if \( V_0 \in X \) and \( g' = g \cup (\text{FV}(V_i) \cap \text{Dom}(\alpha)) \)

<in: \( V_0, V_i \)>

\[
\ll \alpha | \mu \rr_x^g \to \ll \alpha | \mu, V_0 \in V_i \rr_x^{g'}
\]

if \( V_0 \in S \) and \( \text{FV}(V_i) \cap \text{Dom}(\alpha) \subseteq S \)

and \( X' = X \cup (\text{FV}(V_i) - \text{Dom}(\alpha)) \)
Computation Sequences and Paths

If \( k \) is a configuration, then the computation tree \( \Gamma(k) \) is the set of all finite sequences of labelled transitions \([k_i \xrightarrow{li} k_{in} \mid i < n]\) for some \( n \in \mathbb{N} \), with \( k = k_0 \). Such sequences are called computation sequences.

A computation path from \( k \) is a maximal linearly ordered set of computation sequences in the computation tree, \( \Gamma(k) \).

\( \Gamma^\infty(k) \) denotes the set of all paths from \( k \).
FAIRNESS

A path $\pi = [k_i \xrightarrow{e_i} k_{i+1} \mid i < \infty]$ in the computation tree $T^\infty (k)$ is fair if each enabled transition eventually happens or becomes permanently disabled.

For a configuration $k$ we define $F (k)$ to be the subset of $T^\infty (k)$ that are fair.
Composition of Actor Configurations

\[ k_0 = \ll x_0 \mid M_0 \rr_{x_0} \]
\[ k_1 = \ll x_1 \mid M_1 \rr_{x_1} \]

\[ k_0 \parallel k_1 = \ll x_0 \cup x_1 \mid M_0 \cup M_1 \rr_{(x_0 \cup x_1) \cup (g_0 \cup g_1)} \]

Configurations are "composable" if

\[ \text{Dom}(x_0) \cap \text{Dom}(x_1) = \emptyset \]
\[ x_0 \cup \text{Dom}(x_1) \subseteq g_1 \]
\[ x_1 \cup \text{Dom}(x_0) \subseteq g_0 \]

\[ k_\emptyset = \ll \emptyset \mid \emptyset \rr_{\emptyset} \]
\( \underbrace{AC} \)

\[
K_0 \parallel k_1 = k_1 \parallel K_0
\]
\[
K_0 \parallel K_0 = K_0
\]
\[
(K_0 \parallel k_1) \parallel k_2 = K_0 \parallel (k_1 \parallel k_2)
\]

**Closed Configurations**

A configuration in which both the receptionist and external actors sets are empty is said to be closed.

**Composition Preserves Computation Paths**

\[
\mathcal{Z}(K_0 \parallel k_1) = \mathcal{Z}(K_0) \parallel \mathcal{Z}(k_1)
\]
\[
\mathcal{F}(K_0 \parallel k_1) = \mathcal{F}(K_0) \parallel \mathcal{F}(k_1)
\]