Concurrency: Theory, Languages and Programming

– Pi Calculus Examples –

Session 7 – December 4, 2002

Uwe Nestmann

EPFL-LAMP

Replication via Recursion

In the presence of process identifiers, recursion means that a process identifier A defined by

$$A(\,\vec{x}\,) \stackrel{\text{def}}{=} Q_A$$

can be used in any process term P by means of instantiation. Note that A could also be used like this within Q_A itself ...

Using recursion, how can we model/simulate replication? Define a process identifier that, when triggered, "behaves roughly like" $\boxed{!\,P}$

Recursion via Replication

Using replication, recursion can be modeled through:

1. invent name a to stand for identifier A
2. for any R,
   let $\widehat{R}$ denote the result of replacing any call $A\langle\,\vec{w}\,\rangle$ by $\overline{a}\langle\vec{w}\rangle.\mathbf{0}$
3. replace P by

$$(\boldsymbol{\nu} a)\,(\;\widehat{P}\;\mid !\,a(\vec{x}).\widehat{Q_A}\;)$$

Example:

$$\begin{array}{rcl} A(\,x_1,x_2\,) & \stackrel{\text{def}}{=} & \overline{x_1}\langle x_2\rangle.B\langle\,x_1,x_2\,\rangle \\ B(\,x_1,x_2\,) & \stackrel{\text{def}}{=} & \overline{x_2}\langle x_1\rangle.A\langle\,x_1,x_2\,\rangle \\ C(\,x_1,x_2\,) & \stackrel{\text{def}}{=} & \overline{x_1}\langle x_2\rangle.C\langle\,x_2,x_1\,\rangle \end{array}$$

Unbounded Buffers

$$\begin{array}{rcl} B(\,i,o\,) & \stackrel{\text{def}}{=} & i(x).C\langle\,x,i,o\,\rangle \\ C(\,x,i,o\,) & \stackrel{\text{def}}{=} & \overline{o}\langle x\rangle.B\langle\,i,o\,\rangle \\ & & +\;\; i(y).(\;C\langle\,y,i,o\,\rangle \frown C\langle\,x,i,o\,\rangle\;) \end{array}$$

where

$$X\langle\,i,o\,\rangle \frown Y\langle\,i,o\,\rangle \stackrel{\text{def}}{=} (\boldsymbol{\nu} m)\,(\;X\langle\,i,o\,\rangle[m/o] \mid Y\langle\,i,o\,\rangle[m/i]\;)$$

- ☐ Follow the sequence $\stackrel{i1}{\longrightarrow}\stackrel{i2}{\longrightarrow}\stackrel{\bar{o}2}{\longrightarrow}\stackrel{\cdots}{\longrightarrow}$ to convince yourself that the buffer process is indeed a buffer (FIFO) and that i can grow unboundedly.
- ☐ Note the "type" of the stored values ...
- ☐ Note the behavior of empty cells inside a buffer "chain".

Elastic Buffers

lake the buffer elastic,
e., make empty cells disappear!

everal design decisions to be taken concern the question
hen an empty cell should cut itself out of a chain and die.

- if empty cell is next to a full/empty cell?
- if empty cell is left/right to a cell?
- should it be *allowed* (suicide)
  or *forced* (murder) to die?

ne goal of this exercise is to make you think about how to
rgue for or against that the various design decisions above
ead to equivalent solutions.

Elastic Buffers: Setup

$$
\begin{aligned}
B(\,i,l,o,r\,) &\stackrel{\text{def}}{=} i().C\langle\, i,l,o,r\,\rangle \\
&\quad + \ \ldots \\
C(\,i,l,o,r\,) &\stackrel{\text{def}}{=} \overline{o}\langle\rangle.B\langle\, i,l,o,r\,\rangle \\
&\quad + \ i().(\ C\langle\, i,l,o,r\,\rangle \frown C\langle\, i,l,o,r\,\rangle\ ) \\
&\quad + \ \ldots
\end{aligned}
$$

where

$$
X\langle\, i,l,o,r\,\rangle \frown Y\langle\, i,l,o,r\,\rangle \stackrel{\text{def}}{=} \ldots
$$

Elastic Buffers: cut-when-left

$$
\begin{aligned}
B &\stackrel{\text{def}}{=} (i,l,o,r).\\
&\qquad i().C\langle\, i,l,o,r\,\rangle \\
&\quad + \ \ldots \\
C &\stackrel{\text{def}}{=} (i,l,o,r).\\
&\qquad \overline{o}\langle\rangle.B\langle\, i,l,o,r\,\rangle \\
&\quad + \ i().(\ (C \frown C)\langle\, i,l,o,r\,\rangle\ ) \\
&\quad + \ \ldots
\end{aligned}
$$

Elastic Buffers: cut-when-right

$$
\begin{aligned}
B &\stackrel{\text{def}}{=} (i,l,o,r).\\
&\qquad i().C\langle\, i,l,o,r\,\rangle \\
&\quad + \ \ldots \\
C &\stackrel{\text{def}}{=} (i,l,o,r).\\
&\qquad \overline{o}\langle\rangle.B\langle\, i,l,o,r\,\rangle \\
&\quad + \ i().(\ (C \frown C)\langle\, i,l,o,r\,\rangle\ ) \\
&\quad + \ \ldots
\end{aligned}
$$