1 Introduction

A fundamental limitation of CCS is that the communication structure of a process is fixed. For example, it is easy to show that the set $\{\alpha \mid P \xrightarrow{\alpha} P'\}$ is finite. The π-calculus is a similar, but more expressive calculus that addresses this deficiency.

2 Syntax

$$
\begin{array}{rcll}
P & ::= & 0 & \text{Inert} \\
& \mid & x(y).P & \text{Receive} \\
& \mid & \overline{x}\langle y \rangle.P & \text{Send} \\
& \mid & P_1 \mid P_2 & \text{Parallel composition} \\
& \mid & \nu x.\ P & \text{Restriction} \\
& \mid & !P & \text{Replication}
\end{array}
$$

Compared to CCS, note that instead of simply interacting on a named channel, we can now communicate channel names! In addition, π-calculus does not have summation or top-level definitions. Again, we work up to α equivalence for restrictions.

3 Labeled Transition System

Structural congruence is defined as follows:

$$
\begin{array}{l}
P|Q \equiv Q|P \\
(P|Q)|R \equiv P|(Q|R)P|0 \equiv P \\
\nu x.\ 0 \equiv 0 \\
\nu x.\ (P|Q) \equiv (\nu x.P)|Q \text{ if } x \notin fv(Q) \\
!P \equiv !P|P
\end{array}
$$

Reduction is defined as follows:

$$\frac{}{\overline{x}\langle y \rangle.\ P|x(z).Q \to P|Q\{y/z\}}$$

$$\frac{P \to P'}{P|Q \to P'|Q}$$

$$\frac{P \to P'}{\nu x.\ P \to \nu x.\ P'}$$

$$\frac{P \equiv Q \qquad Q \to Q' \qquad Q' \equiv P'}{P' \to P'}$$

The definition of the labeled transition system is left as an exercise.

4 Programming in the π-calculus

The rest of this lecture will explore how we can implement various programming constructs in π-calculus.

4.1 Polyadic Communication

Although π-calculus send/receive are unary, we can encode polyadic communication as follows:

$$\begin{aligned} \bar{l}\langle x_1, \ldots, x_n\rangle &\triangleq \nu p.\ \bar{l}\langle p\rangle.\ \bar{p}\langle x_1\rangle \ldots \bar{p}\langle x_n\rangle \\ l(y_1, \ldots, y_n) &\triangleq l(p).\ p(y_1) \ldots p(y_n) \end{aligned}$$

Intuitively, this encoding works by first creating a fresh channel name p, sending p along l, and then sending the actual names $x_1, \ldots, x_n$ along p. The use of a fresh channel ensures that multiple senders and receivers will not interfere with each other.

4.2 Booleans

We can encode booleans as processes that receive names of t and f channels, and then send on the corresponding channel.

$$\begin{aligned} \mathit{True}(b) &\triangleq\ !b(t, f).\bar{t} \\ \mathit{False}(b) &\triangleq\ !b(t, f).\bar{f} \\ \mathit{Cond}(P, Q)(b) &\triangleq \nu t, f(\bar{b}\langle t, f\rangle.(t().P + f().Q)) \end{aligned}$$

Note that we put a ! in front of processes to turn them into servers create arbitrary numbers of the original process. This prevents their destruction after sending or receiving a message.

4.3 Internal Choice

Although π-calculus does not have summation, we can encode a limited form of "internal" choice:

$$P \oplus Q \triangleq \nu c.\ (\bar{c}\langle\rangle \mid c().P \mid c().Q)$$

4.4 References

We can also encode mutable references as processes.

$$\begin{aligned} \mathit{Ref}(r, w, i) &\triangleq \nu l.\ \bar{l}\langle i\rangle \mid \mathit{Read}(l, r) \mid \mathit{Write}(l, r) \\ \mathit{Read}(l, r) &\triangleq \nu l.\ !r(c).\ l(v).(\bar{c}\langle v\rangle \mid \bar{l}\langle v\rangle) \\ \mathit{Write}(l, w) &\triangleq \nu l.\ !w(c, v').\ l(v).(\bar{c}\langle\rangle \mid \bar{l}\langle v'\rangle) \end{aligned}$$

4.5 λ-calculus

Finally, we can encode the λ-calculus into the π-calculus as follows:

$$
\begin{array}{rcl}
[\![x]\!](p) & \triangleq & \overline{x}\langle p\rangle \\
[\![\lambda x.\, e]\!](p) & \triangleq & p(x,q).[\![e]\!](q) \\
[\![e_1, e_2]\!](p) & \triangleq & \nu q([\![e_1]\!](q) \mid \nu y(\overline{q}\langle y, p\rangle \mid !y(r).[\![e_2]\!](r))
\end{array}
$$

Note the similarity to continuation-passing style.