CONCURRENCY ISSUES

ADVANTAGES

- Reactive Programming
- Availability of Services
- Controllability.
- Active Objects (Actors)
- Asynchronous Messages
- Parallelism

CONCURRENCY ISSUES

LIMITATIONS

- Safety "Nothing bad ever happens"
- Liveness "Anything ever happens at all"
- Nondeterminism
- Inherent sequencing in algorithms
- Resource consumption
  - Threads are expensive
  - Scheduling overhead
  - Synchronization overhead

∴ Concurrent programs can run more slowly than sequential ones even if you have multiple CPUs.

THEORY AND PRACTICE

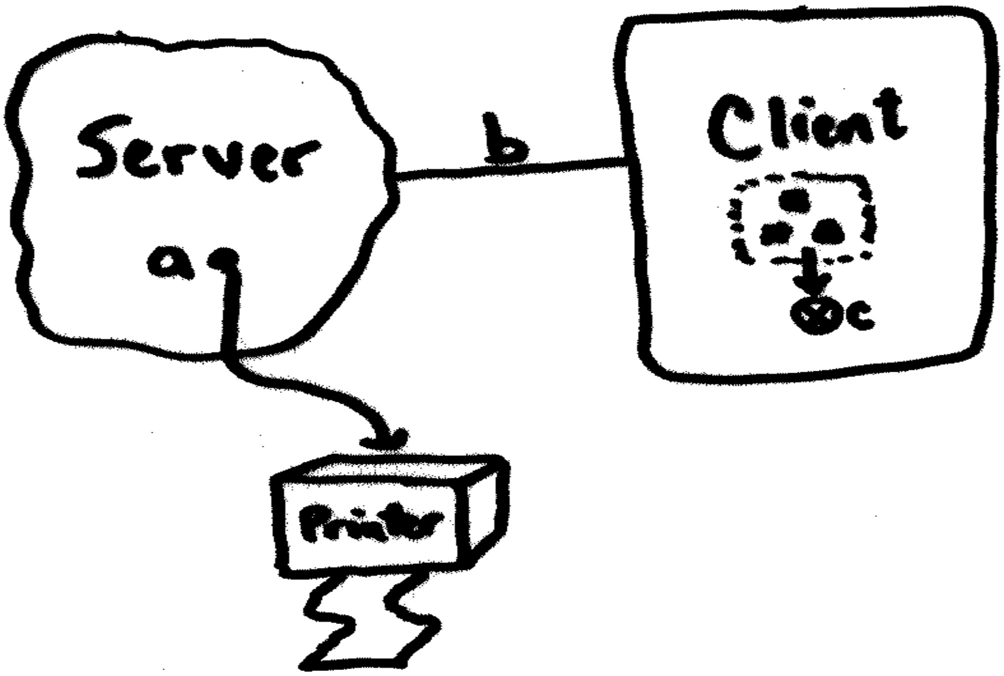
"In theory, theory and practice are very related. In practice, ..."

λ-Calculus	Functional & Imperative Programming
Predicate Calculus	Logic Programming
π-Calculus & Actor Model	Concurrent & Distributed Programming

Those theoretically inclined, check out:

A Theory of Objects, by Abadi & Cardelli.

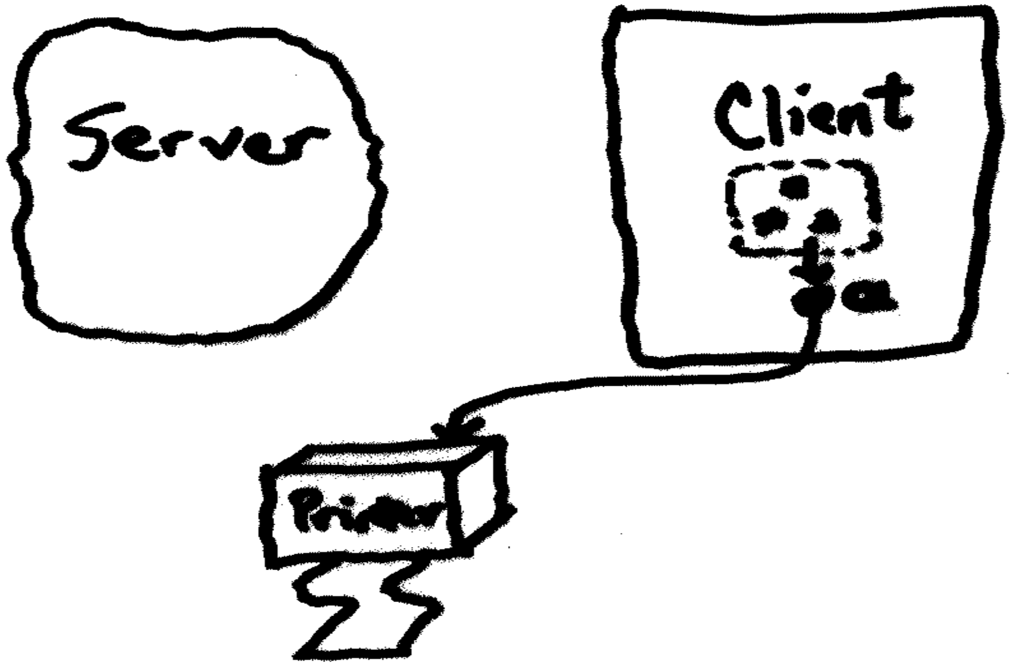
π-CALCULUS



$$\bar{b}a.S \mid b(c).\bar{c}d.P$$

$$\xrightarrow{\tau} \quad S \mid \bar{c}d.P\{a/c\}$$

$$S \mid \bar{a}d.P$$



S | ad. P

π-CALCULUS NOTATION

CHANNELS / NAMES

$a, b, c, \ldots$

PROCESSES / AGENTS

$P, Q, R, \ldots$

e.g.:

$$\bar{b}a.S \mid b(c).\bar{c}d.P$$

$$\xrightarrow{\tau} S \mid \bar{c}d.P\{a/c\}$$

$$S \mid \bar{a}d.P\{a/c\}$$

π-CALCULUS SYNTAX

Prefixes	$\alpha ::=$	$\bar{a}x$	Output
		$a(x)$	Input
		τ	Silent
Agents	$P ::=$	0	Nil
		$\alpha.P$	Prefix
		$P+P$	Sum
		$P \mid P$	Parallel
		if $x=y$ then P	Match
		if $x \neq y$ then P	Mismatch
		$(\nu x)P$	Restriction
		$A(y_1, \ldots, y_n)$	Identifier

Definitions $\quad A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P \quad$ (where $i \neq j \Rightarrow x_i \neq x_j$)

FREE AND BOUND OCCURRENCES

$$\left.\begin{array}{l} a(x).P \\ (\nu x)P \end{array}\right\}\ \text{bind } x \text{ on } P$$

$$\left.\bar{a}x.P\ \right\}\ \text{does NOT bind } x \text{ on } P$$

EXERCISE

$fn(a(x).P) =$

$fn((\nu x)P) =$

$fn(\bar{a}x.P) =$

$bn(a(x).P) =$

$bn((\nu x)P) =$

$bn(\bar{a}x.P) =$

to be defined in terms of a, x, $fn(P)$ and $bn(P)$.

STRUCTURAL CONGRUENCE

P and Q are structurally congruent, if they represent the same thing, although syntactically different. We write $P \equiv Q$, if

① P and Q are variants of α-conversion.

② Abelian laws for $|$ and $+$.

$$P | Q \equiv Q | P$$
$$(P | Q) | R \equiv P | (Q | R)$$
$$P | 0 \equiv P$$

③ Unfolding law

$$A(\tilde{y}) \equiv P\{\tilde{y}/\tilde{x}\} \quad \text{if} \quad A(\tilde{x}) \stackrel{\text{def}}{=} P.$$

④ Scope extension laws

$$\begin{aligned}
(\nu x)\, 0 &\equiv 0 \\
(\nu x)\,(P | Q) &\equiv P \,|\, (\nu x)\, Q \\
(\nu x)\,(P + Q) &\equiv P + (\nu x)\, Q
\end{aligned} \left.\vphantom{\begin{aligned}a\\a\end{aligned}}\right\} \text{if } x \notin fn(P)$$

$$\begin{aligned}
(\nu x) \text{ if } u = v \text{ then } P &\equiv \text{if } u = v \text{ then } (\nu x) P \\
(\nu x) \text{ if } u \neq v \text{ then } P &\equiv \text{if } u \neq v \text{ then } (\nu x) P \\
(\nu x)(\nu y)\, P &\equiv (\nu y)(\nu x)\, P
\end{aligned}$$

SCOPE EXTRUSION

$a(x).P \mid (\nu b)\,\bar{a}b.Q \quad , \quad b \notin fn(P)$

$\equiv (\nu b)\,(a(x).P \mid \bar{a}b.Q)$

$\xrightarrow{\tau} (\nu b)\,(P\{b/x\} \mid Q)$

If $b \in fn(P)$, rename b to $b' \notin fn(P)$

$$a(x).\bar{c}x \mid (\nu b)\,\bar{a}b$$

$$\equiv (\nu b)\,(a(x).\bar{c}x \mid \bar{a}b)$$

$$\xrightarrow{\tau} (\nu b)\,(\bar{c}b \mid 0)$$

$$\xrightarrow{\tau} (\nu b)\,\bar{c}b$$

$$a(b).\bar{c}b \mid (\nu b)\,\bar{a}b$$

$$\equiv a(b).\bar{c}b \mid (\nu d)\,\bar{a}d$$

$$\equiv (\nu d)\,(a(b).\bar{c}b \mid \bar{a}d)$$

$$\xrightarrow{\tau} (\nu d)\,\bar{c}d$$

EXERCISES:

Q:

1. $(\nu a)\,\bar{a}b.P \mid \bar{a}c.Q \mid a(x).R \xrightarrow{\tau} ?$
2. $(\nu b)\,\bar{a}b \mid a(x).\bar{b}x$

EXECUTOR EXAMPLE

$$\mathcal{E}xec(x) = x(y).\bar{y}$$

$$A(x) = (\nu z)(\bar{x}z \mid z.P)$$

Agent $A(x)$ will behave as P when combined with $\mathcal{E}xec(x)$.

Proof.

$$A(x) \mid \mathcal{E}xec(x)$$

REPLICATION

$$(!P) \stackrel{def}{=} P \mid !P$$

A REFERENCE CELL IN π-CALCULUS

$$\mathrm{Ref}(r,w,i) = (\nu \ell)(\,\bar{\ell} i \mid \mathrm{ReadServer}(\ell, r) \mid \mathrm{WriteServer}(\ell, w))$$

$$\mathrm{ReadServer}(\ell, r) = !\, r(c).\, \ell(v).\, (\bar{c} v \mid \bar{\ell} v)$$

$$\mathrm{WriteServer}(\ell, w) = !\, w(c, v').\, \ell(v).\, (\bar{c} \mid \bar{\ell} v')$$

Example using reference cell:

$$(\nu c)\, \bar{w}\langle c, v\rangle .\, c.\, (\nu d)\, \bar{r} d.\, d(e).\, Q$$

will receive the value v over the channel d assuming no other processes interacting with the reference cell.