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,..."

\(\lambda\)-Calculus \hspace{2cm} Functional Programming & Imperative Logic Programming

Predicate Calculus

\(\pi\)-Calculus & Actor Model \hspace{1cm} Concurrent & Distributed Programming

Those theoretically inclined, check out:

A Theory of Objects, by Abadi & Cardelli.
\[ b(a).S \mid b(c).\exists d. P \]

\[ \exists a. S \mid \exists d. P \{a/c\} \]

\[ S \mid \exists a. P \]

\[ S \mid \exists d. P \]
**T-PI Calculus Notation**

**Channels/Names**

\( a, b, c, \ldots \)

**Processes/Agents**

\( P, Q, R, \ldots \)

**e.g.**:

\[
\begin{align*}
\overrightarrow{ba.S} & \mid b(c).\overrightarrow{ed.P} \\
\overleftarrow{s} & \mid \overrightarrow{ed.P{a/c}} \\
\overleftarrow{s} & \mid \overrightarrow{\overline{a}d.P{a/c}}
\end{align*}
\]
**π-Calculus Syntax**

**Prefixes**

\[ \alpha ::= \bar{\alpha} \]
\[ \alpha(x) \]
\[ \varepsilon \]

**Output**

**Input**

**Silent**

**Agents**

\[ P ::= \text{Nil} \]
\[ \text{Prefix} \]
\[ \text{Sum} \]
\[ \text{Parallel} \]
\[ \text{Match} \]
\[ \text{Mismatch} \]
\[ \text{Restriction} \]
\[ \text{Identifier} \]

\[ P ::= O \]
\[ \alpha . P \]
\[ P + P \]
\[ P \parallel P \]
\[ \text{if } x = y \text{ then } P \]
\[ \text{if } x \neq y \text{ then } P \]
\[ (\forall x) P \]
\[ A(y_1, ..., y_n) \]

**Definitions**

\[ A(x_1, ..., x_n) \overset{\text{def}}{=} P \quad \text{(where } i \neq j \Rightarrow x_i \neq x_j \text{)} \]
Free and Bound Occurrences

\[ a(x).P \] bind \( x \) on \( P \)

\[ (y x)P \]

\[ \overline{a}x.P \] does NOT bind \( x \) on \( P \)
\[
\begin{align*}
\log_a (a(x)) & = \ln \left( \frac{\ln (\ln (x))}{x} \right) \\
\log_a (\ln (x)) & = \ln \left( \frac{\ln (\ln (x))}{x} \right) \\
\log_a (\ln (x)) & = \log_{\ln (x)} (x)
\end{align*}
\]

\textbf{Exercise}

Define \(a \log_x P\) as \(\log_x (aP)\) or \(\log_x \frac{a}{P}\) as appropriate.
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 $\alpha$-conversion.
2. Abelian laws for $\cdot$ and $+$.
   
   $P \cdot Q \equiv Q \cdot P$
   $(P \cdot Q) \cdot R \equiv P \cdot (Q \cdot R)$
   $P \cdot O \equiv P$

3. Unfolding law
   
   $A(\bar{y}) \equiv P[\bar{y}/x]$ if $A(x) \equiv P$.

4. Scope extension laws
   
   $(y_x) \cdot 0 \equiv 0$
   $(y_x) (P \cdot Q) \equiv P \cdot (y_x) Q$ if $x \notin fmla$
   $(y_x) (P + Q) \equiv P + (y_x) Q$
   $(y_x) \text{ if } u = v \text{ then } P \equiv \text{ if } u = v \text{ then } (y_x) P$
   $(y_x) \text{ if } u \neq v \text{ then } P \equiv \text{ if } u \neq v \text{ then } (y_x) P$
   $(y_x) (y_y) P \equiv (y_y)(y_x) P$
Scope: Extausion

\[ a(x).P \mid (\forall b) \bar{a}b.Q \quad b \notin \text{fn}(P) \]

\[ \forall (\forall b) (a(x).P \mid \bar{a}b.Q) \]

\[ \exists (\forall b) (P \{b/x\} \mid Q) \]

If \( b \in \text{fn}(P) \), rename \( b \) to \( b' \notin \text{fn}(P) \).
\( a(x).\exists x \mid (\forall b) \exists b \)

\( \equiv (\forall b) (a(x).\exists x \mid \exists b) \)

\( \equiv (\forall b) (\exists b) \exists b \)

\[\exists (\forall b) \exists b \]

**EXERCISES:**

1. \( (\forall a) \exists b. P \mid \exists c. Q \mid a(x). R \)

2. \( (\forall b) \exists b \mid \exists a(x). \exists x \)
Executor Example

\[ \text{Exec}(x) = x(y).\overline{y} \]

\[ A(x) = (\forall z)(\overline{xz} \mid x. P) \]

Agent \( A(x) \) will behave as \( P \) when combined with \( \text{Exec}(x) \).

Proof.

\[ A(x) \mid \text{Exec}(x) \]

}\]
REPPLICATION

(\pi B) \overset{def}{=} P \mid \pi B
A Reference Cell in \( \Pi \)-Calculus

\[
\text{Ref}(r,w,i) = (\forall \ell) (\bar{\ell} : | \text{ReadServer}(\ell, r) \\
| \text{WriteServer}(\ell, w))
\]

\[
\text{ReadServer}(\ell, r) = ! r(c). \ell(v). (\exists v \mid \bar{\ell}v)
\]

\[
\text{WriteServer}(\ell, w) = ! w(c,v'). \ell(v). (\exists | \bar{\ell}v')
\]

Example using reference cell:

\[
(\forall c) \bar{\omega} <c, v> \cdot c. (\forall d) \bar{d}d(e). \bar{\omega}
\]

will receive the value \( v \) over the channel \( d \)

assuming no other processes interacting with the reference cell.