The π-Calculus

Overview
- Syntax
- Reaction, actions, and transitions

References
- Robin Milner, “Communicating and Mobil Systems”
- Robin Milner, Joachim Parrow, David Walker, “A Calculus of Mobile Processes, Part I+II”

From CCS to π-Calculus
- The π-calculus is a model of concurrent computation based upon the notion of naming.
- The π-calculus is a calculus in which the topology of communication can evolve dynamically during evaluation.
- In the π-calculus communication links are identified by names, and computation is represented purely as the communication of names across links.
- The π-calculus is an extension of the process algebra CCS, following the work by Engberg and Nielsen who added mobility to CCS while preserving its algebraic properties.

The π-Calculus - Basic Ideas
- The most primitive in the π-calculus is a name. Names, infinitely many, are $x, y, \ldots \in \mathbb{N}$, they have no structure.
- In the π-calculus we only have one other kind of entity: a process. We use $P, Q, \ldots$ to range over processes.
- Processes evolve by performing actions. The capabilities for actions are expressed via prefixes, of which there are four kinds:

$$\pi ::= \pi_1 | x[2] | \pi_1[x = y]$$
Prefixes

- **Output prefix:**
  The output prefix $\pi y$ denotes a process's capability to send the name 'y' via the name 'x'.

- **Input prefix:**
  The input prefix $x(z)$ denotes a process's capability to receive any name, say 'y', via the name 'x'. The role of 'z' is that of placeholder that will be substituted with the received name.

- **Silent prefix:**
  The silent prefix $\tau$ denotes an unobservable action. As in CCS, the silent prefix can be thought of as expressing an internal action of the process.

- **Match prefix:**
  The match prefix $[x = y] x$ is a conditional capability, that is, it if x and y are the same name.

---

The $\pi$-Calculus - Syntax

- The set $P_\pi$ of $\pi$-calculus process expressions is defined by the following syntax:

  $P ::=$ $\Sigma_{i \in I} \pi i.P_i$ | $P_1 | P_2$ | $\nu x P$ | $!P$

  where $I$ is any finite indexing set. The processes $\Sigma_{i \in I} \pi i.P_i$ are called summations or sums. We often write simply $M_1 + M_2$ for $\Sigma_{i \in \{1,2\}} \pi i.P_i$. Note, however, that summation binds more tightly than composition, restriction, and replication. If $I = \emptyset$, then $\Sigma_{i \in I} \pi i.P_i$ is the empty sum, written $0$.

---

Restriction

- In the restriction $\nu x P$, the scope of the name $x$ is restricted to $P$. Components of $P$ can use $x$ to interact with one another, but not with other processes.

- The process $\nu x ((x(z)z)0 + \nu y.0) \pi x.0$ has only two capabilities: to send 'y' via 'w', and to evolve invisibly as an effect of an interaction between its components via 'x'.

- The scope of a restriction may change as the result of an interaction between processes. Unlike restriction in concurrent processes (or CCS), names in $\pi$-calculus can be transmitted. In the case of a restricted name this means that the name may extrude its original scope.
Replication

- The replication !P can be thought of as an infinite composition P | P | ... or, equivalently, a process satisfying the equation !P = P | !P.
- Replication is an operator that makes it possible to express infinite behaviors.
- The process lx(z).!yz.0 can receive names via 'x' repeatedly, and can repeatedly send via 'y' any name it does receive.

Link Passing – Free Channel

The agent P, with x ∉ fn(P'), has a link 'x' to R, and wishes to pass 'x' along its link 'y' to Q. Q is willing to receive it.

P | {x/z}Q' | P' y(z).Q | x.P

Link Passing – Private Channel

The agent P, with x ∉ fn(P'), has a link 'x' to R, and wishes to pass 'x' along the private 'y' to Q. The scope of the private name include both P and Q.

P | x|y(Q' | x(z)).Q | x.P(y

Q'' in the diagram is \{x/z\}Q'.
The agent $P$, with $x \in \text{fn}(P')$, has a link 'x' to R, and wishes to pass 'x' along its link 'y' to Q. Q is willing to receive it, but already possesses a private link 'x' to S. The latter needs to be renamed to avoid the capture of the received free name 'x' in S.

$$\nu \cdot P \vdash \nu y (y(z).Q[S] | R) \rightarrow \nu \cdot P \vdash (y(z).Q[S] | x).P$$

Now suppose both Q and P' possess no 'x'-link.

$$\nu x (\nu y (y(z).Q[S] | R) \rightarrow \nu \cdot P \vdash (y(z).Q[S] | x).P$$

However, there is a general law: $\nu x | P_1 \vdash P_2 \rightarrow \nu x | P_2 \vdash P_1$.

$$\nu x \vdash (P | R | [x/z]Q') = P' \vdash (R | [x/z]Q')$$
### Binding

- In each of $x(z).P$ and $\nu z P$, the name $z$ occurs bound with scope $P$.
- An occurrence of a name in a process is free if it is not bound.
- We write $\text{fn}(P)$ for the set of names that have free occurrences in $P$.

\[
\text{fn}(z y.0 + \nu v.0) = \{z, y, w, v, x, u\}
\]
\[
\text{fn}(\nu \langle x \rangle z y.0 + \nu v.0) = \{y, w, v\}
\]

### Substitution

- The process, $\sigma P$, obtained by applying the name substitution $\sigma$ to $P$ is defined as follows:

\[
\begin{align*}
\sigma 0 &= 0 \\
\sigma (\pi \cdot P) &= \sigma \pi \cdot \sigma P \\
\sigma (M_1 + M_2) &= \sigma M_1 + \sigma M_2 \\
\sigma (P_1 | P_2) &= \sigma P_1 | \sigma P_2 \\
\sigma (\nu z P) &= \nu z \cdot \sigma P \\
\sigma (!P) &= ! \cdot \sigma P
\end{align*}
\]

Note $\sigma (\sum \pi_i P_i) = \sum \sigma \pi_i \sigma P_i$.

### Process Context

- An occurrence of 0 in a process is degenerate if it is the left or right term in a sum $M_1 + M_2$, and non-degenerate otherwise.

- A $\pi$-calculus process context $C$ is obtained when the hole $[\star]$ replaces a non-degenerate occurrence of 0 in a process term given by the $\pi$-calculus grammar.

Examples:
- $\nu z ([\star] \cdot \langle z \rangle \cdot [z] \cdot \nu a.0)$
- $x(z) \cdot \nu w \cdot \langle z \rangle \cdot [y(v)] \cdot 0$

- An equivalence relation $S$ on $\pi$-calculus processes is a congruence if $(P, Q) \in S$ implies $(C[P], C[Q]) \in S$ for every context $C$. 
Structural Congruence Rules

- Change of bound names (alpha-conversion)

Axioms:

- \[ [x = x] \pi.P \equiv \pi.P \]
- \[ \pi(M_1 + M_2 + M_3) \equiv (M_1 + M_2) + M_3 \]
- \[ M_1 + M_2 = M_2 + M_1 \]
- \[ M + 0 = M \]
- \[ \pi(P_1 | (P_2 | P_3)) = (P_1 | P_2) | P_3 \]
- \[ \pi(P_1 | P_2) = P_2 | P_1 \]
- \[ P + 0 = P \]
- \[ w z P \equiv w z P \]
- \[ w 0 = 0 \]
- \[ \pi(P_1 | P_2) = P_1 | w z P_2 \text{ if } z \notin \text{fni}(P_1) \]
- \[ \pi P \equiv P | \pi P \]

Standard Form

A \pi-calculus process expression \( \nu a (M_1 | ... | P_r | ... | R_n) \), where each \( M_i \) is a non-empty summation, is said to be in standard form.

Theorem:
Every \pi-calculus process is structurally congruent to a standard form.

Proof:
For any restriction \( \nu a \) inside a summation, we can bring it to the outermost by using alpha-conversion (if necessary) followed by the rule \( \nu a \) (\( R | O \)) = \( \nu a R | O \) in conjunction with some of the laws of structural congruence.

Reaction Rules

The reaction relation \( \rightarrow \) over \( P \) contains exactly those transitions that can be inferred from the rules in the following table:

- TAU: \( \tau : P + M \rightarrow P \)
- INTER: \( (x|z) : P + M | (y|Q + N) \rightarrow (y|z)P | Q \)
- PAR: \( P \rightarrow P' \)
- TAU: \( P | Q \rightarrow P | Q \)
- RES: \( \nu a P \rightarrow \nu a P' \)
- STRUCT: \( P = Q \rightarrow P' | P = Q' \)
- Q \rightarrow Q'