The π-Calculus: A Comprehensive Course

Syntax, Semantics, Mobility, and Security

Concurrency Theory

February 22, 2026

Syntax of the π-calculus

Definition. Processes P, Q are built from a set of names $x, y, u, v \in \mathcal{N}$ representing communication channels:

$$
\begin{aligned}
P, Q ::= \; & \mathbf{0} && \text{(Nil: Inactive process)} \\
\mid \; & \overline{x}\langle y \rangle.P && \text{(Output: Send name } y \text{ on channel } x\text{, then } P\text{)} \\
\mid \; & x(z).P && \text{(Input: Receive on } x\text{, bind to } z \text{ in } P\text{)} \\
\mid \; & P \mid Q && \text{(Parallel: Concurrent execution)} \\
\mid \; & (\nu x)P && \text{(Restriction: Local name } x \text{ in } P\text{)} \\
\mid \; & !P && \text{(Replication: Persistent service / Recursion)} \\
\mid \; & P + Q && \text{(Choice: Non-deterministic sum)}
\end{aligned}
$$

Example: $\overline{a}\langle b \rangle.\mathbf{0} \mid a(x).\overline{x}\langle c \rangle.\mathbf{0}$

Operator Intuition: The Power of Names

In the π-calculus, **names are values**.

- **Input** $x(z).P$**:** Acts as a binder for z. x is the "address."
- **Output** $\overline{x}\langle y \rangle.P$**:** Message y is sent to address x.
- **Restriction** $(\nu x)P$**:** Creates a "private wire." Only processes inside P can see x.
- **Replication** $!P$**:** Equivalent to $P \mid P \mid P \mid \ldots$.

The "First-Class" Nature

Because names can be communicated, the **network topology** can change during execution. This is the definition of **Mobility**.

Comparison: CCS vs. π-calculus

Feature	**CCS (Static)**	**π-calculus (Mobile)**
Links	Fixed at design time	**Dynamically created**
Communication	Synchronization only	**Name Passing**
Topology	Rigid graph	**Reconfigurable graph**
Decidability	Decidable bisimulation	**Undecidable**

Structural Congruence I: The Intuition

Definition. Structural Congruence ($\equiv$) identifies processes that are semantically identical regardless of their syntactic representation.

- **Syntactic Noise:** The order of parallel processes shouldn't matter ($P \mid Q \equiv Q \mid P$).
- **Rearrangement:** It allows us to move a sender next to a receiver to enable interaction.

Example: $((\nu a)\mathbf{0} \mid P) \mid Q \equiv P \mid Q$. $\equiv$ "cleans up" the system.

Structural Congruence II: Monoid and Alpha Rules

1. Parallel and Summation as Monoids

$(P, |, \mathbf{0})$ and $(P, +, \mathbf{0})$ are symmetric monoids:

- **Commutativity:** $P \mid Q \equiv Q \mid P$
- **Associativity:** $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
- **Identity (0):** $P \mid \mathbf{0} \equiv P$

2. Alpha-Conversion (α)

Bound names can be renamed: $a(x).\overline{x}\langle y\rangle \equiv a(z).\overline{z}\langle y\rangle$.

Structural Congruence III: Replication and Restriction

3. Replication Law

$!P \equiv P \mid !P$ (Allows "spinning up" a copy of a server).

4. Restriction Laws

- **Swapping:** $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$.
- **Garbage Collection:** $(\nu x)\mathbf{0} \equiv \mathbf{0}$.
- **Scope Extrusion:** $(\nu x)(P \mid Q) \equiv P \mid (\nu x)Q$ **if** $x \notin \mathrm{fn}(P)$.

LTS Rule: Input (IN) and Output (OUT)

$$\frac{}{a(x).P \xrightarrow{a(b)} P\{b/x\}}\ (\text{In}) \qquad \frac{}{\overline{a}\langle b\rangle.P \xrightarrow{\overline{a}\langle b\rangle} P}\ (\text{Out})$$

- **In:** Ready to receive *any* name b on a; x is substituted by b.
- **Out:** Offers to send b on a.

Example: $\overline{ch}\langle msg\rangle \xrightarrow{\overline{ch}\langle msg\rangle} \mathbf{0}$.

LTS Rule: Parallel (PAR) and Communication (COM)

$$\frac{P \xrightarrow{\alpha} P' \quad \mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q} \text{ (Par)}$$

$$\frac{P \xrightarrow{\overline{a}\langle b \rangle} P' \quad Q \xrightarrow{a(b)} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \text{ (Com)}$$

Com: The "Handshake." A sender and receiver synchronize, resulting in an internal τ step.

LTS Rule: Opening (OPEN) and Closing (CLOSE)

$$\frac{P \xrightarrow{\overline{a}\langle b\rangle} P' \quad a \neq b}{(\nu b)P \xrightarrow{\overline{a}(b)} P'} \text{(Open)}$$

$$\frac{P \xrightarrow{\overline{a}(b)} P' \quad Q \xrightarrow{a(b)} Q'}{P|Q \xrightarrow{\tau} (\nu b)(P'|Q')} \text{(Close)}$$

Intuition: **Open** prepares a private name b for export. **Close** completes the communication, expanding the restriction νb over the recipient.

Toy Example: Mobile Radio Handover

Goal: A Mobile Station (MS) moves from the range of Tower A to Tower B. Tower A must "hand over" the private link to Tower B.

The Setup (Initial Topology)

$TowerA(talk, switch) \triangleq talk(msg).P \mid \overline{switch}\langle talk \rangle.\mathbf{0}$
$TowerB(switch) \triangleq switch(new\_link).TowerB'(new\_link)$
$MS(talk) \triangleq \overline{talk}\langle \text{“Hello”} \rangle.\mathbf{0}$

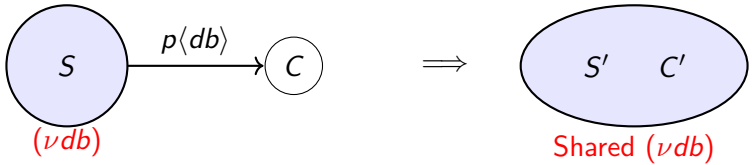
The Execution (Handover):

1. **Initial System:** $(\nu talk)(MS(talk) \mid TowerA(talk, sw)) \mid TowerB(sw)$
2. **Mobility Step:** $TowerA$ sends the *private name* $talk$ over the *public name* sw.
3. **Result:** $(\nu talk)(MS(talk) \mid \mathbf{0} \mid TowerB'(talk))$

Why CCS Fails: In CCS, the alphabet of a process is fixed. $TowerB$ cannot "learn" a new port name $talk$ at runtime. In π, $TowerB$ evolves to include the port $talk$ in its interface, effectively "receiving" the connection.

Scope Extrusion in Depth

Initial: $(\nu db)(S \mid C)$ where S has a private database handle. **Action:** S sends db to C. The scope of (νdb) "stretches."



Mobility: The link structure changes dynamically.

Early Bisimulation ($\sim_e$)

Definition. A symmetric relation $\mathcal{R}$ such that $P\mathcal{R}Q$ implies: If $P \xrightarrow{a(w)} P'$, then $\exists Q'$ such that $Q \xrightarrow{a(w)} Q'$ and $P'\mathcal{R}Q'$.

Intuition: The name w is chosen **first**. Q reacts to the specific name provided.

Late Bisimulation ($\sim_l$)

Definition. $P\mathcal{R}Q$ implies: If $P \xrightarrow{a(x)} P'$, then $\exists Q'$ such that $Q \xrightarrow{a(x)} Q'$ and **for all** w, $P'\{w/x\}\mathcal{R}Q'\{w/x\}$.

Intuition: Q must commit to a transition **before** the name w is seen.

Result: Late is stricter than Early ($\sim_l \subset \sim_e$).

Undecidability I: Encoding Counters

Counters are recursive chains.

- **Zero Cell:** $Z(c) \triangleq !c(inc, dec, z).\overline{z}\langle\rangle$
- **Successor Cell:** $S(c, p) \triangleq !c(inc, dec, z).\overline{dec}\langle p\rangle$

Example (Increment): To increment $Z(c)$, the controller creates $(\nu c')(S(c', c) \mid Z(c))$.

Undecidability II: Controller Logic

- **INC(C1):** $(\nu c_{new})(S(c_{new}, c_1) \mid L_{next}\{c_{new}/c_1\})$.
- **DECJ(C1):** Synchronize on *dec*. If successful, S returns the p pointer; the controller updates its "head" to p.

Theorem. Halting is undecidable $\implies$ Reachability and Bisimilarity in π-calculus are undecidable.

Summary of Applications

- **Protocol Verification:** Using tools like **ProVerif** to check for leaks in TLS, SSH, or Signal protocols.
- **Programming Languages:** Theoretical basis for **Go** (channels), **Erlang** (actor isolation), and **Pict**.
- **Business Processes (BPMN):** Modeling complex workflows where permissions/roles change dynamically.
- **Biological Modeling:** Using π-calculus to represent protein-protein interactions as concurrent communications.

Mobility is the fundamental abstraction for the modern connected world.

Suggested Reading: The Canon of π-Calculus

- **The Definitive Textbook:**
  - **Sangiorgi, D., & Walker, D. (2001).** *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press.
  - *Note: The most exhaustive formal treatment of the subject available.*
- **On Bisimulation and Equivalences:**
  - **Sangiorgi, D. (1996).** *A Theory of Bisimulation for the π-Calculus*. Information and Computation.
  - *Focus: Introduced the concept of "Open Bisimulation" and refined the Early/Late distinction.*
- **Foundational Origins:**
  - **Milner, R., Parrow, J., & Walker, D. (1992).** *A Calculus of Mobile Processes, I and II.*
  - *Focus: The original papers that transitioned the community from CCS to π.*
- **Modern Perspectives:**
  - **Sangiorgi, D. (2011).** *Introduction to Bisimulation and Coinduction.*
  - *Focus: A broader look at the mathematical tools used to prove process equivalence.*