# Process Calculus and CCS

## Syntax, Operational Semantics, and Intuition

CS691: Concurrency Theory

# Lecture Plan (2 Hours)

- Motivation and Process Calculi (15 min)
- CCS: Actions and Syntax (20 min)
- Operational Semantics: Rules + Intuition (45 min)
- Worked Examples and LTS Construction (20 min)
- Behavioral Equivalence (Bisimulation Intuition) (10 min)
- Exercises and Discussion (10 min)

# Why Concurrency Needs New Models

- Sequential models focus on input/output and termination
- Concurrent systems are:
  - Interactive
  - Non-terminating
  - Composed of independently evolving agents
- Correctness depends on how components interact

**Intuition**

We do not ask: *What result does the program compute?*
We ask: *What behaviors can the system exhibit over time?*

# What Is a Process Calculus?

- A formal language for describing interacting processes
- Emphasizes observable actions and interaction patterns
- Comes with:
  - Syntax (how processes are built)
  - Operational semantics (how they evolve)
  - Behavioral equivalences (when two processes are the same)

**Key Idea**

Processes are *not functions*. They are ongoing entities.

# Why CCS?

- Introduced by Milner as a minimal calculus of communication
- Captures the essence of:
  - Synchronization
  - Nondeterminism
  - Parallelism
- Serves as a foundation for many later calculi

# Actions and Complementarity

- Let $a$ range over channel names
- Actions/Channel-names:
  - $a$ : input (receive)
  - $\bar{a}$ : output (send)
  - $\tau$ : internal (silent)

**Intuition**

Communication happens only when an input meets a matching output. The interaction itself becomes invisible ($\tau$).

# CCS Syntax

$$P ::= 0 \mid \alpha.P \mid P + Q \mid P \mid\mid Q \mid P \setminus L \mid A \mid P[f]$$

where $A, P, Q$ are processes (or CCS expressions), $\alpha \in$ Actions and $f$ : Actions $\mapsto$ Actions.

- 0: inactive process
- $\alpha.P$: prefix
- $P + Q$: choice
- $P||Q$: parallel composition
- $P \setminus L$: restriction
- $A$: process constant (recursion)
- $P[f]$: renaming.

# Labelled Transition Systems

- Semantics given by transitions:

$$P \xrightarrow{\alpha} P'$$

- Nodes: process expressions
- Edges: observable or internal actions

**Intuition**

A process is a *state machine*, but with rich structure.

# Prefix Rule

$$\alpha.P \xrightarrow{\alpha} P$$

## Intuition

The process is forced to perform its leading action. Afterwards, the continuation remains.

## Example

$a.0 \xrightarrow{a} 0$

# Choice Rules

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

## Intuition

Choice represents *external nondeterminism*. The environment observes which action resolves the choice.

# Parallel: Independent Actions

$$\frac{P \xrightarrow{\alpha} P'}{P||Q \xrightarrow{\alpha} P'||Q}$$

**Intuition**

Parallel components may evolve independently. Concurrency does not imply synchronization.

# Synchronization Rule

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P||Q \xrightarrow{\tau} P'||Q'}$$

**Intuition**

Communication is atomic and internal. Both sides must be ready simultaneously.

# Restriction Rule

$$\frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad (\alpha \notin L \cup \bar{L})$$

**Intuition**

Restriction hides channels from the environment. Used to model private communication.

# Process Constants and Recursion

$$A \triangleq P$$

### Intuition

Allows infinite behavior and looping processes. Models servers, protocols, and controllers.

# Recursion in CCS: An Example

$$A \triangleq a.A$$

- $A$ is a process constant
- The definition says: after performing $a$, behave like $A$ again

**Intuition**

This process can perform action $a$ forever. It models a system that is always ready to engage in $a$.

$$A \xrightarrow{a} A \xrightarrow{a} A \xrightarrow{a} \cdots$$

# Renaming: Semantics

$$\frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

## Intuition

Renaming does not change the control structure of a process. It uniformly changes the *interface* through which actions are observed.

# Renaming: Example

Let:

$$P = a.0 \qquad \text{and} \qquad f(a) = b$$

Then:

$$P \xrightarrow{a} 0$$

By the renaming rule:

$$P[f] \xrightarrow{b} 0$$

**Intuition**

The behavior of the process is unchanged, but its visible action has been renamed from $a$ to $b$.

Renaming preserves synchronization:

$$(a.0 \mid \bar{a}.0)[f] \;\equiv\; b.0 \mid \bar{b}.0$$

# Example 1: Simple Communication

$$P = a.0 \mid\mid \bar{a}.0$$

- Both sides can synchronize
- Single $\tau$ step

$$P \xrightarrow{\tau} 0 \mid\mid 0$$

# Example 2: Choice and Parallelism

$$P = (a.0 + b.0) || \bar{a}.0$$

- Synchronization on $a$ possible
- Or independent execution of $b$

**Teaching Point**

Nondeterminism arises from both choice and concurrency.

# Example 3: Restriction

$$(a.0 || \bar{a}.0) \setminus \{a\}$$

- External observer sees only $\tau$
- Communication becomes private

# Why Behavioral Equivalence?

- Many syntactically different processes behave the same
- We want equivalence based on observable behavior

**Question**

When can one process safely replace another?

# Bisimulation (Intuition)

- Two processes are bisimilar if they can mimic each other
- Step-by-step matching of actions
- Strong notion of behavioral equivalence

**Key Insight**

Bisimulation respects the branching structure of behavior.

# Exercises

1. Construct the LTS of:

$$(a.0 + b.0)||(\bar{a}.0 + c.0)$$

2. Identify all $\tau$-transitions
3. Apply restriction on $a$ and compare behavior

# Takeaway

- CCS provides a precise language for concurrency
- Operational rules have clear behavioral meaning
- Forms the foundation for verification and extensions