Plan for the lecture:

- Informal introduction to CCS
- Syntax of CCS
- Semantics of CCS
CCS Basics (Sequential Fragment)

- *Nil* (or 0) process (the only atomic process)
- action prefixing (*a.P*)
- names and recursive definitions (*\text{def}*)
- nondeterministic choice (+)

This is Enough to Describe Sequential Processes

Any finite LTS can be described (up to isomorphism) by using the operations above.
CCS Basics (Sequential Fragment)

- *Nil* (or 0) process (the only atomic process)
- action prefixing (*a.P*)
- names and recursive definitions (*def*)
- nondeterministic choice (+)

This is Enough to Describe Sequential Processes

Any finite LTS can be described (up to isomorphism) by using the operations above.
parallel composition (|)
(synchronous communication between two components = handshake synchronization)

- restriction ($P \setminus L$)
- relabelling ($P[f]$)
parallel composition (|)  
(synchronous communication between two components = handshake synchronization)

restriction ($P \setminus L$)

relabelling ($P[f]$)
parallel composition (|)
(synchronous communication between two components = handshake synchronization)

restriction (\(P \setminus L\))

relabelling (\(P[f]\))
Definition of CCS (channels, actions, process names)

Let

- \( \mathcal{A} \) be a set of channel names (e.g. tea, coffee are channel names)

- \( \mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}} \) be a set of labels where
  - \( \overline{\mathcal{A}} = \{ \overline{a} \mid a \in \mathcal{A} \} \)
    (elements of \( \mathcal{A} \) are called names and those of \( \overline{\mathcal{A}} \) are called co-names)
  - by convention \( \overline{\overline{a}} = a \)

- \( \text{Act} = \mathcal{L} \cup \{ \tau \} \) is the set of actions where
  - \( \tau \) is the internal or silent action
    (e.g. \( \tau \), tea, coffee are actions)

- \( \mathcal{K} \) is a set of process names (constants) (e.g. CM).
Definition of CCS (channels, actions, process names)

Let

- $\mathcal{A}$ be a set of channel names (e.g. *tea*, *coffee* are channel names)

- $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$ be a set of labels where
  - $\overline{\mathcal{A}} = \{\overline{a} \mid a \in \mathcal{A}\}$
    (elements of $\mathcal{A}$ are called names and those of $\overline{\mathcal{A}}$ are called co-names)
  - by convention $\overline{\overline{a}} = a$

- $\text{Act} = \mathcal{L} \cup \{\tau\}$ is the set of actions where
  - $\tau$ is the internal or silent action
    (e.g. $\tau$, *tea*, *coffee* are actions)

- $\mathcal{K}$ is a set of process names (constants) (e.g. CM).
Definition of CCS (channels, actions, process names)

Let

- $\mathcal{A}$ be a set of channel names (e.g. tea, coffee are channel names)
- $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$ be a set of labels where
  - $\overline{\mathcal{A}} = \{\overline{a} \mid a \in \mathcal{A}\}$
    (elements of $\mathcal{A}$ are called names and those of $\overline{\mathcal{A}}$ are called co-names)
  - by convention $\overline{\overline{a}} = a$
- $\text{Act} = \mathcal{L} \cup \{\tau\}$ is the set of actions where
  - $\tau$ is the internal or silent action
    (e.g. $\tau$, tea, coffee are actions)
- $\mathcal{K}$ is a set of process names (constants) (e.g. CM).
Definition of CCS (channels, actions, process names)

Let

- $\mathcal{A}$ be a set of channel names (e.g. *tea*, *coffee* are channel names)
- $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$ be a set of labels where
  - $\overline{\mathcal{A}} = \{ \overline{a} \mid a \in \mathcal{A} \}$
    (elements of $\mathcal{A}$ are called names and those of $\overline{\mathcal{A}}$ are called co-names)
  - by convention $\overline{\overline{a}} = a$

- $\text{Act} = \mathcal{L} \cup \{ \tau \}$ is the set of actions where
  - $\tau$ is the internal or silent action
    (e.g. $\tau$, *tea*, *coffee* are actions)

- $\mathcal{K}$ is a set of process names (constants) (e.g. *CM*).
\[ P := K \quad \mid \quad \alpha.P \quad \mid \quad \sum_{i \in I} P_i \quad \mid \quad P_1|P_2 \quad \mid \quad P \setminus L \quad \mid \quad P[f] \]

- **process constants** \((K \in \mathcal{K})\)
- **prefixing** \((\alpha \in \text{Act})\)
- **summation** \((l \text{ is an arbitrary index set})\)
- **parallel composition**
- **restriction** \((L \subseteq A)\)
- **relabelling** \((f : \text{Act} \to \text{Act})\) such that
  - \(f(\tau) = \tau\)
  - \(f(\overline{a}) = \overline{f(a)}\)

The set of all terms generated by the abstract syntax is the set of **CCS process expressions** (and is denoted by \(\mathcal{P}\)).

**Notation**

\[ P_1 + P_2 = \sum_{i \in \{1,2\}} P_i \quad \text{Nil} = 0 = \sum_{i \in \emptyset} P_i \]
Definition of CCS (expressions)

\[ P := K \mid \alpha.P \mid \sum_{i \in I} P_i \mid P_1 \mid P_2 \mid P \setminus L \mid P[f] \]

- process constants \((K \in \mathcal{K})\)
- prefixing \((\alpha \in \text{Act})\)
- summation \((I\) is an arbitrary index set\)
- parallel composition
- restriction \((L \subseteq A)\)
- relabelling \((f : \text{Act} \to \text{Act})\) such that
  - \(f(\tau) = \tau\)
  - \(f(\tau) = f(\tau)\)

The set of all terms generated by the abstract syntax is the set of CCS process expressions (and is denoted by \(\mathcal{P}\)).

Notation

\[ P_1 + P_2 = \sum_{i \in \{1,2\}} P_i \quad \text{Nil} = 0 = \sum_{i \in \emptyset} P_i \]
Definition of CCS (expressions)

\[ P := K \mid \alpha.P \mid \sum_{i \in I} P_i \mid P_1|P_2 \mid P \setminus L \mid P[f] \]

- process constants \((K \in \mathcal{K})\)
- prefixing \((\alpha \in \text{Act})\)
- summation \((I\) is an arbitrary index set\)
- parallel composition
- restriction \((L \subseteq A)\)
- relabelling \((f : \text{Act} \rightarrow \text{Act})\) such that
  \[ f(\tau) = \tau \]
  \[ f(a) = f(a) \]

The set of all terms generated by the abstract syntax is the set of CCS process expressions (and is denoted by \(\mathcal{P}\)).

Notation

\[ P_1 + P_2 = \sum_{i \in \{1,2\}} P_i \]
\[ \text{Nil} = 0 = \sum_{i \in \emptyset} P_i \]
Definition of CCS (expressions)

\[ P := \begin{array}{l}
K \\
\alpha.P \\
\sum_{i \in I} P_i \\
P_1 | P_2 \\
P \setminus L \\
P[f]
\end{array} \quad \begin{array}{l}
\text{process constants } (K \in \mathcal{K}) \\
\text{prefixing } (\alpha \in \text{Act}) \\
\text{summation } (I \text{ is an arbitrary index set}) \\
\text{parallel composition} \\
\text{restriction } (L \subseteq A) \\
\text{relabelling } (f : \text{Act} \rightarrow \text{Act}) \text{ such that}
\begin{align*}
& f(\tau) = \tau \\
& f(\overline{a}) = \overline{f(a)}
\end{align*}
\]

The set of all terms generated by the abstract syntax is the set of CCS process expressions (and is denoted by \( \mathcal{P} \)).

**Notation**

\[ P_1 + P_2 = \sum_{i \in \{1, 2\}} P_i \quad \text{Nil} = 0 = \sum_{i \in \emptyset} P_i \]
Definition of CCS (expressions)

\[ P ::= \begin{array}{ll}
K & \text{process constants} (K \in \mathcal{K}) \\
\alpha.P & \text{prefixing} (\alpha \in \text{Act}) \\
\sum_{i \in I} P_i & \text{summation} (I \text{ is an arbitrary index set}) \\
P_1 | P_2 & \text{parallel composition} \\
P \setminus L & \text{restriction} (L \subseteq \mathcal{A}) \\
P[f] & \text{relabelling} (f : \text{Act} \to \text{Act}) \text{ such that} \\
& \begin{array}{ll}
& f(\tau) = \tau \\
& f(\overline{a}) = \overline{f(a)}
\end{array}
\end{array} \]

The set of all terms generated by the abstract syntax is the set of CCS process expressions (and is denoted by \( \mathcal{P} \)).

Notation

\[ P_1 + P_2 = \sum_{i \in \{1, 2\}} P_i \quad \quad \text{Nil} = 0 = \sum_{i \in \emptyset} P_i \]
\[ P := \begin{align*}
K & \quad \text{process constants} \ (K \in \mathcal{K}) \\
\alpha.P & \quad \text{prefixing} \ (\alpha \in \text{Act}) \\
\sum_{i \in I} P_i & \quad \text{summation} \ (I \text{ is an arbitrary index set}) \\
P_1|P_2 & \quad \text{parallel composition} \\
P \setminus L & \quad \text{restriction} \ (L \subseteq A) \\
P[f] & \quad \text{relabelling} \ (f : \text{Act} \to \text{Act}) \text{ such that} \\
& \quad \begin{align*}
& \quad f(\tau) = \tau \\
& \quad f(a) = \overline{f(a)}
\end{align*}
\end{align*} \]

The set of all terms generated by the abstract syntax is the set of CCS process expressions (and is denoted by \( \mathcal{P} \)).

**Notation**

\[ P_1 + P_2 = \sum_{i \in \{1,2\}} P_i \quad \text{Nil} = 0 = \sum_{i \in \emptyset} P_i \]
Definition of CCS (expressions)

\[ P := \begin{align*}
K & : \text{process constants} \ (K \in \mathcal{K}) \\
\alpha.P & : \text{prefixing} \ (\alpha \in \text{Act}) \\
\sum_{i \in I} P_i & : \text{summation} \ (I \text{ is an arbitrary index set}) \\
P_1|P_2 & : \text{parallel composition} \\
P \setminus L & : \text{restriction} \ (L \subseteq \mathcal{A}) \\
P[f] & : \text{relabelling} \ (f : \text{Act} \to \text{Act}) \text{ such that} \\
& \quad f(\tau) = \tau \\
& \quad f(\overline{a}) = \overline{f(a)}
\end{align*} \]

The set of all terms generated by the abstract syntax is the set of CCS process expressions (and is denoted by \( \mathcal{P} \)).

**Notation**

\[ P_1 + P_2 = \sum_{i \in \{1,2\}} P_i \]
\[ \text{Nil} = 0 = \sum_{i \in \emptyset} P_i \]
Definition of CCS (expressions)

\[ P ::= K \mid \alpha.P \mid \sum_{i \in I} P_i \mid P_1 \parallel P_2 \mid P \setminus L \mid P[f] \]

- \( K \): process constants \((K \in \mathcal{K})\)
- \( \alpha.P \): prefixing \((\alpha \in \text{Act})\)
- \( \sum_{i \in I} P_i \): summation \((I \text{ is an arbitrary index set})\)
- \( P_1 \parallel P_2 \): parallel composition
- \( P \setminus L \): restriction \((L \subseteq A)\)
- \( P[f] \): relabelling \((f : \text{Act} \rightarrow \text{Act})\) such that
  - \( f(\tau) = \tau \)
  - \( f(a) = f(a) \)

The set of all terms generated by the abstract syntax is the set of CCS process expressions (and is denoted by \( \mathcal{P} \)).

**Notation**

\[ P_1 + P_2 = \sum_{i \in \{1, 2\}} P_i \quad \text{Nil} = 0 = \sum_{i \in \emptyset} P_i \]
Precedence

1. restriction and relabelling (tightest binding)
2. action prefixing
3. parallel composition
4. summation

Example: \( R + a.P \mid b.Q \setminus L \) means \( R + ((a.P)\mid(b.(Q \setminus L))) \).
Precedence

1. restriction and relabelling (tightest binding)
2. action prefixing
3. parallel composition
4. summation

Example: $R + a.P|b.Q \setminus L$ means $R + ((a.P)|(b.(Q \setminus L)))$. 
Definition of CCS (defining equations)

A collection of defining equations of the form

\[ K \overset{\text{def}}{=} P \]

where \( K \in \mathcal{K} \) is a process constant and \( P \in \mathcal{P} \) is a CCS process expression.

- Only one defining equation per process constant.
- Recursion is allowed: e.g. \( A \overset{\text{def}}{=} \overline{a}.A \mid A \).
Semantics of CCS

Syntax
CCS
(collection of defining equations)

Semantics
LTS
(labelled transition systems)

HOW?
Semantics of CCS

Syntax
CCS
(collection of defining equations)

Semantics
LTS
(labelled transition systems)

HOW?
Semantics of CCS

Syntax
CCS
(collection of defining equations)

Semantics
LTS
(labelled transition systems)

HOW?
Small-step operational semantics where the behaviour of a system is inferred using syntax driven rules.

Given a collection of CCS defining equations, we define the following LTS \((\text{Proc}, \text{Act}, \{\xrightarrow{a} \mid a \in \text{Act}\})\):

- \(\text{Proc} = \mathcal{P}\) (the set of all CCS process expressions)
- \(\text{Act} = \mathcal{L} \cup \{\tau\}\) (the set of all CCS actions including \(\tau\))
- transition relation is given by SOS rules of the form:

\[
\text{RULE} \quad \frac{\text{premises}}{\text{conclusion}} \quad \text{conditions}
\]
Structural Operational Semantics (SOS)—G. Plotkin 1981

Small-step operational semantics where the behaviour of a system is inferred using syntax driven rules.

Given a collection of CCS defining equations, we define the following LTS \((\text{Proc}, \text{Act}, \{\xrightarrow{a} \mid a \in \text{Act}\})\):

- \(\text{Proc} = \mathcal{P}\) (the set of all CCS process expressions)
- \(\text{Act} = \mathcal{L} \cup \{\tau\}\) (the set of all CCS actions including \(\tau\))
- transition relation is given by SOS rules of the form:

\[
\text{RULE} \quad \frac{\text{premises}}{\text{conclusion}} \quad \text{conditions}
\]
SOS rules for CCS \((\alpha \in \text{Act}, \ a \in \mathcal{L})\)

\[\begin{align*}
\text{ACT} & \quad \alpha. P \xrightarrow{\alpha} P \\
\text{COM1} & \quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \\
\text{SUM}_j & \quad \frac{P_j \xrightarrow{\alpha} P_j'}{\sum_{i \in I} P_i \xrightarrow{\alpha} P_j'} \\
\text{COM2} & \quad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'} \\
\text{COM3} & \quad \frac{P \xrightarrow{a} P'}{P|Q \xrightarrow{\tau} P'|Q'} \\
\text{RES} & \quad \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \\
\text{REL} & \quad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]} \\
\text{CON} & \quad \frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'} \\[K \overset{\text{def}}{=} P
\end{align*}\]
Deriving Transitions in CCS

Let $A \overset{\text{def}}{=} a.A$. Then

$$((A | \overline{a}.\text{Nil}) | b.\text{Nil})[c/a] \xrightarrow{c} ((A | \overline{a}.\text{Nil}) | b.\text{Nil})[c/a].$$

Why?
Let $A \overset{\text{def}}{=} a.A$. Then

$$
((A \mid \overline{a}.\text{Nil}) \mid b.\text{Nil})[c/a] \xrightarrow{c} ((A \mid \overline{a}.\text{Nil}) \mid b.\text{Nil})[c/a].
$$

Why?

REL

$$
((A \mid \overline{a}.\text{Nil}) \mid b.\text{Nil})[c/a] \xrightarrow{c} ((A \mid \overline{a}.\text{Nil}) \mid b.\text{Nil})[c/a]
$$
Let $A \overset{\text{def}}{=} a.A$. Then

$$(A | \overline{a}.Nil) | b.Nil) [c/a] \xrightarrow{c} ((A | \overline{a}.Nil) | b.Nil) [c/a].$$

Why?
Let $A \overset{\text{def}}{=} a.A$. Then

\[(A | \overline{a}.\text{Nil}) \mid b.\text{Nil})[c/a] \xrightarrow{c} (A | \overline{a}.\text{Nil}) \mid b.\text{Nil})[c/a].\]

Why?

\[
\begin{array}{c}
\text{COM1} \\
\hline
A \mid \overline{a}.\text{Nil} \xrightarrow{a} A \mid \overline{a}.\text{Nil}
\end{array}
\]

\[
\begin{array}{c}
\text{COM1} \\
\hline
(A \mid \overline{a}.\text{Nil}) \mid b.\text{Nil} \xrightarrow{a} (A \mid \overline{a}.\text{Nil}) \mid b.\text{Nil}
\end{array}
\]

\[
\begin{array}{c}
\text{REL} \\
\hline
((A \mid \overline{a}.\text{Nil}) \mid b.\text{Nil})[c/a] \xrightarrow{c} ((A \mid \overline{a}.\text{Nil}) \mid b.\text{Nil})[c/a]
\end{array}
\]
Let $A \overset{\text{def}}{=} a.A$. Then

$$( (A | \overline{a}.Nil) | b.Nil ) [c/a] \xrightarrow{c} ( (A | \overline{a}.Nil) | b.Nil ) [c/a].$$

Why?

\[
\begin{align*}
\text{CON} & \quad A \overset{\text{def}}{=} a.A \\
\text{COM1} & \quad A \rightarrow A \\
\text{COM1} & \quad (A | \overline{a}.Nil) \rightarrow (A | \overline{a}.Nil) \\
\text{REL} & \quad (A | \overline{a}.Nil) | b.Nil \rightarrow (A | \overline{a}.Nil) | b.Nil \\
\end{align*}
\]
Deriving Transitions in CCS

Let $A \overset{\text{def}}{=} a.A$. Then

$$(A \mid a.\text{Nil}) \mid b.\text{Nil})[c / a] \xrightarrow{c} (A \mid a.\text{Nil}) \mid b.\text{Nil})[c / a].$$

Why?

\[
\begin{align*}
\text{ACT} & \quad a.A \xrightarrow{a} A \\
\text{CON} & \quad A \xrightarrow{a} A \\
\text{COM1} & \quad A \mid a.\text{Nil} \xrightarrow{a} A \mid a.\text{Nil} \\
\text{COM1} & \quad (A \mid a.\text{Nil}) \mid b.\text{Nil} \xrightarrow{a} (A \mid a.\text{Nil}) \mid b.\text{Nil} \\
\text{REL} & \quad ((A \mid a.\text{Nil}) \mid b.\text{Nil})[c / a] \xrightarrow{c} ((A \mid a.\text{Nil}) \mid b.\text{Nil})[c / a]
\end{align*}
\]
LTS of the Process $a.Nil \mid \overline{a}.Nil$