# Topics in Concurrency

Lecture 2

Jonathan Hayman

16 February 2015

# The Calculus of Communicating Systems

- Introduced by Robin Milner in 1980
- First process calculus developed with its operational semantics
- Supports algebraic reasoning about equivalence
- Simplifies Dijkstra's Guarded Command Language by removing the store (store locations can be encoded as processes)
- Processes communicate by sending values (numbers) on channels.

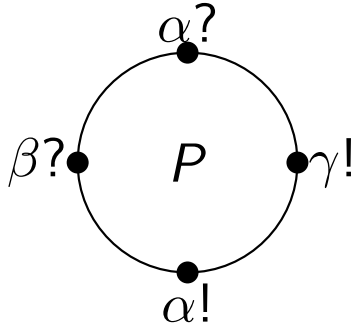
# Action and communication

- As before, communication is synchronous and between two processes.
- Input: $\alpha?x$
- Output: $\alpha!a$
- Silent action: $\tau$. Internal to the process.
- We will use $\lambda$ to range over all the kinds of action.

Processes will structure actions

# Interface diagrams

- *Interface diagrams* describe the channels used by processes for input and output.
- The use of a channel by a process is called a *port*.
- Example: process $P$ inputs on $\alpha, \beta$ and outputs on $\alpha, \gamma$.




- Later examples: links between processes to represent the possibility of communication

# Syntax of CCS

- Expressions: Arithmetic $a$ and Boolean $b$
- Processes:

| | | | |
|---|---|---|---|
| $p$ | ::= | **nil** | nil process |
| | $\mid$ | $(\tau \to p)$ | silent/internal action |
| | $\mid$ | $(\alpha!a \to p)$ | output |
| | $\mid$ | $(\alpha?x \to p)$ | input |
| | $\mid$ | $(b \to p)$ | Boolean guard |
| | $\mid$ | $p_0 + p_1$ | non-deterministic choice |
| | $\mid$ | $p_0 \parallel p_1$ | parallel composition |
| | $\mid$ | $p \backslash L$ | restriction ($L$ a set of channel identifiers) |
| | $\mid$ | $p[f]$ | relabelling ($f$ a function on channel identifiers) |
| | $\mid$ | $P(a_1, \cdots, a_k)$ | process identifier |

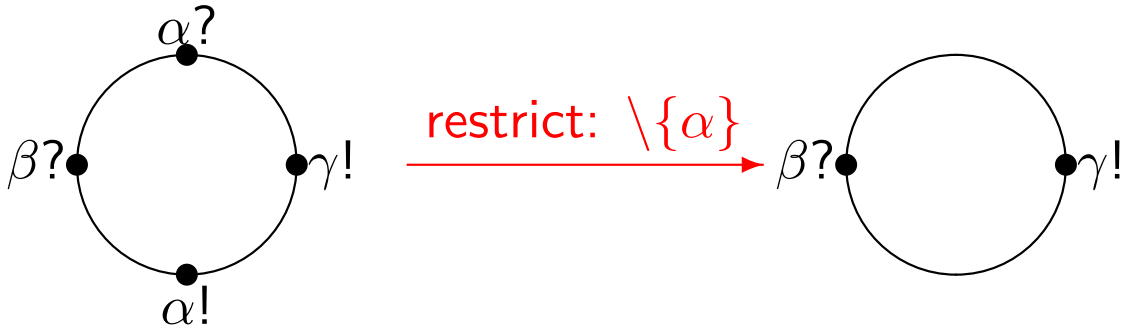
- Process definitions:

$$P(x_1, \cdots, x_k) \stackrel{\text{def}}{=} p$$

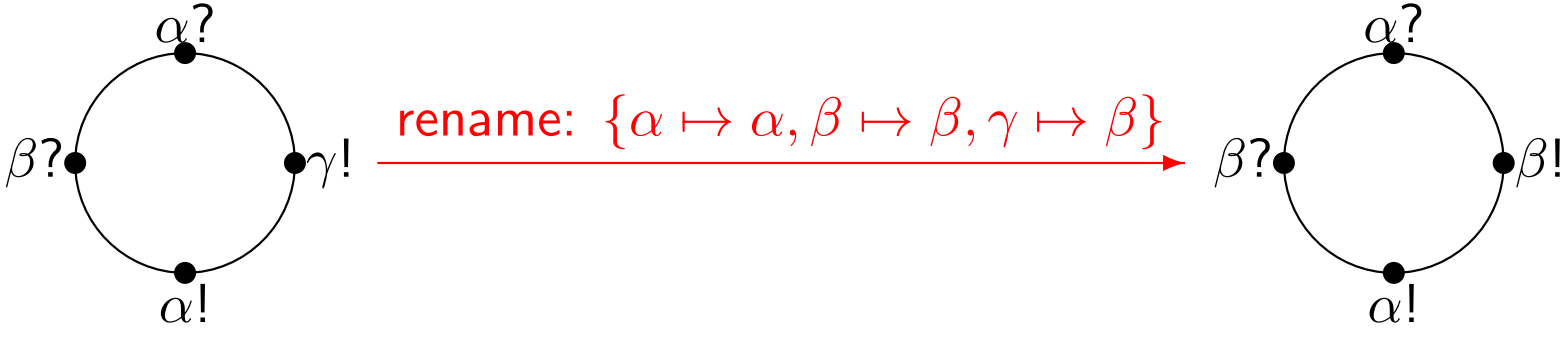
(free variables of $p \subseteq \{x_1, \cdots, x_n\}$)

# Restriction and relabelling: interface diagrams

- $p \setminus L$: Disallow external interaction on channels in $L$




- $p[f]$: Rename external interface to channels by $f$



# Operational semantics of CCS

- **Guarded processes**

$$(\tau \to p) \xrightarrow{\tau} p$$

$$\frac{a \to n}{(\alpha!a \to p) \xrightarrow{\alpha!n} p} \qquad (\alpha?x \to p) \xrightarrow{\alpha?n} p[n/x]$$

$$\frac{b \to \text{true} \qquad p \xrightarrow{\lambda} p'}{(b \to p) \xrightarrow{\lambda} p'}$$

- **Sum**

$$\frac{p_0 \xrightarrow{\lambda} p_0'}{p_0 + p_1 \xrightarrow{\lambda} p_0'} \qquad \frac{p_1 \xrightarrow{\lambda} p_1'}{p_0 + p_1 \xrightarrow{\lambda} p_1'}$$

- **Parallel composition**

$$\frac{p_0 \xrightarrow{\lambda} p_0'}{p_0 \parallel p_1 \xrightarrow{\lambda} p_0' \parallel p_1} \qquad \frac{p_0 \xrightarrow{\alpha?n} p_0' \qquad p_1 \xrightarrow{\alpha!n} p_1'}{p_0 \parallel p_1 \xrightarrow{\tau} p_0' \parallel p_1'}$$

$$\frac{p_1 \xrightarrow{\lambda} p_1'}{p_0 \parallel p_1 \xrightarrow{\lambda} p_0 \parallel p_1'} \qquad \frac{p_0 \xrightarrow{\alpha!n} p_0' \qquad p_1 \xrightarrow{\alpha?n} p_1'}{p_0 \parallel p_1 \xrightarrow{\tau} p_0' \parallel p_1'}$$

- **Restriction**

$$\frac{p \xrightarrow{\lambda} p'}{p \setminus L \xrightarrow{\lambda} p' \setminus L} \quad \text{where if } \lambda \equiv \alpha?n \text{ or } \lambda \equiv \alpha!n \text{ then } \alpha \notin L$$

- **Relabelling**

$$\frac{p \xrightarrow{\lambda} p'}{p[f] \xrightarrow{f(\lambda)} p'[f]}$$

where $f$ is extended to labels as $f(\tau) = \tau$ and $f(\alpha?n) = f(\alpha)?n$ and $f(\alpha!n) = f(\alpha)!n$

- **Identifiers**

$$\frac{p[a_1/x_1, \cdots, a_n/x_n] \xrightarrow{\lambda} p'}{P(a_1, \cdots, a_n) \xrightarrow{\lambda} p'}$$

- **Nil process** no rules

# A derivation from the operational semantics

$$\dfrac{\dfrac{(\alpha!3 \to \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil}}{(\alpha!3 \to \mathbf{nil}) + P \xrightarrow{\alpha!3} \mathbf{nil}}}{((\alpha!3 \to \mathbf{nil}) + P) \parallel (\tau \to \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil} \parallel (\tau \to \mathbf{nil})}$$

# A derivation from the operational semantics

$$\dfrac{\dfrac{\dfrac{(\alpha!3 \to \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil}}{(\alpha!3 \to \mathbf{nil}) + P \xrightarrow{\alpha!3} \mathbf{nil}}}{((\alpha!3 \to \mathbf{nil}) + P) \parallel (\tau \to \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil} \parallel (\tau \to \mathbf{nil})} \qquad (\alpha?x \to \mathbf{nil}) \xrightarrow{\alpha?3} \mathbf{nil}}{(((\alpha!3 \to \mathbf{nil}) + P) \parallel (\tau \to \mathbf{nil})) \parallel (\alpha?x \to \mathbf{nil}) \xrightarrow{\tau} (\mathbf{nil} \parallel (\tau \to \mathbf{nil})) \parallel \mathbf{nil}}$$

# A derivation from the operational semantics

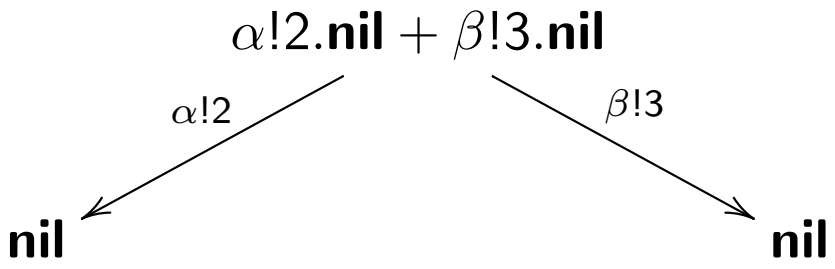
$$\dfrac{\dfrac{\dfrac{(\alpha!3 \to \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil}}{(\alpha!3 \to \mathbf{nil}) + P \xrightarrow{\alpha!3} \mathbf{nil}}}{((\alpha!3 \to \mathbf{nil}) + P) \parallel (\tau \to \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil} \parallel (\tau \to \mathbf{nil})} \qquad (\alpha?x \to \mathbf{nil}) \xrightarrow{\alpha?3} \mathbf{nil}}{(((\alpha!3 \to \mathbf{nil}) + P) \parallel (\tau \to \mathbf{nil})) \parallel (\alpha?x \to \mathbf{nil}) \xrightarrow{\tau} (\mathbf{nil} \parallel (\tau \to \mathbf{nil})) \parallel \mathbf{nil}}$$

$$\overline{(((\alpha!3 \to \mathbf{nil} + P) \parallel \tau \to \mathbf{nil}) \parallel \alpha?x \to \mathbf{nil}) \setminus \{\alpha\} \xrightarrow{\tau} ((\mathbf{nil} \parallel \tau \to \mathbf{nil}) \parallel \mathbf{nil}) \setminus \{\alpha\}}$$

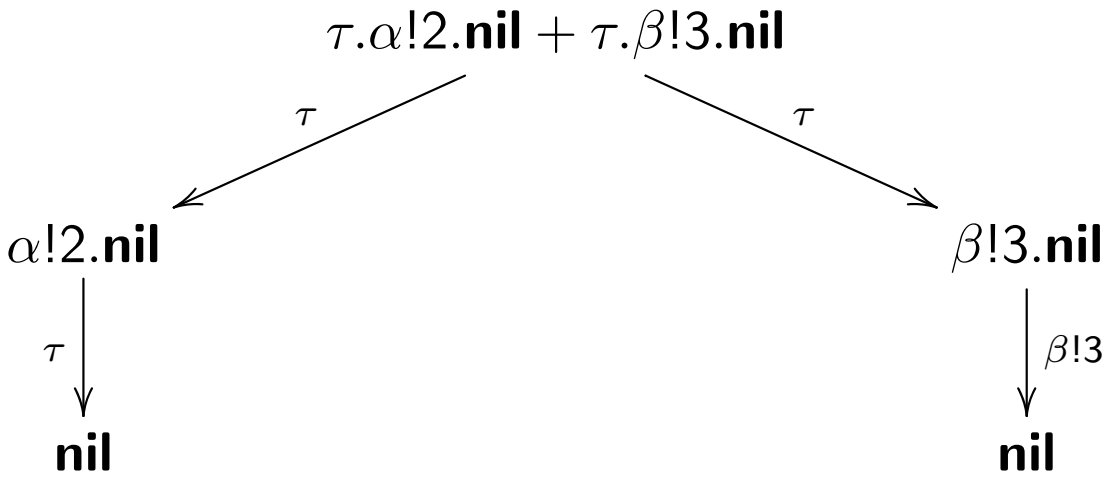
# Further examples

(Write . for $\rightarrow$)
Each step justified by a derivation:

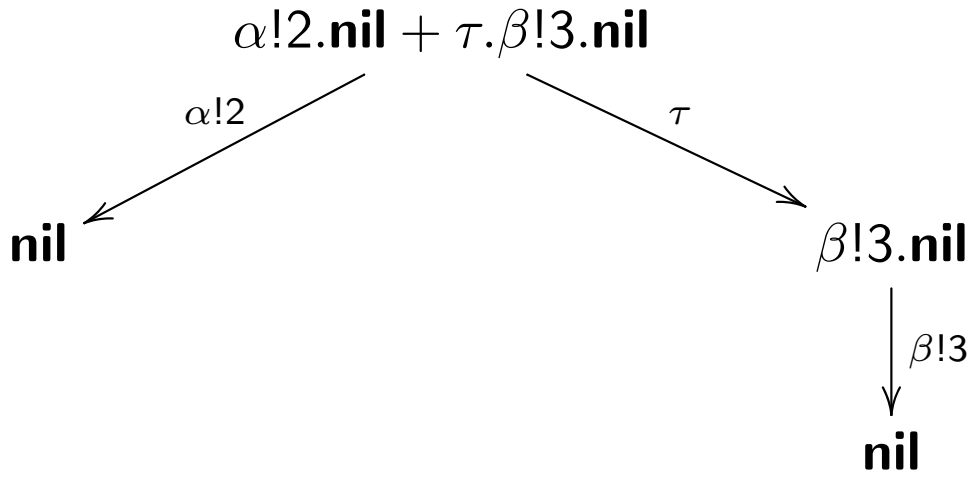
- External choice




- Internal choice



- Mixed choice




- Exercise:

$$\alpha!3.\mathbf{nil} \parallel \alpha?x.\beta!x.\mathbf{nil}$$

- Exercise:

$$(\alpha!3.\mathbf{nil} \parallel \alpha?x.\beta!x.\mathbf{nil}) \setminus \{\alpha\}$$

- Exercise:

$$(\alpha?x.\mathbf{nil} \parallel \beta!4)[\alpha \mapsto \beta, \beta \mapsto \beta]$$

- Exercise: $P$ where

$$P \stackrel{\text{def}}{=} \alpha?x \to x = 1 \to Q(x) \qquad Q(x) \stackrel{\text{def}}{=} \beta!x \to \gamma!x \to Q(x)$$

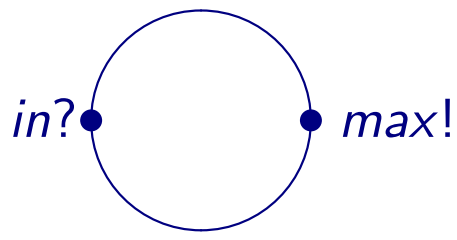
# Conditionals

- Encoding of conditionals:

$$\texttt{if } b \texttt{ then } p_0 \texttt{ else } p_1 \equiv (b \to p_0) + (\neg b \to p_1)$$

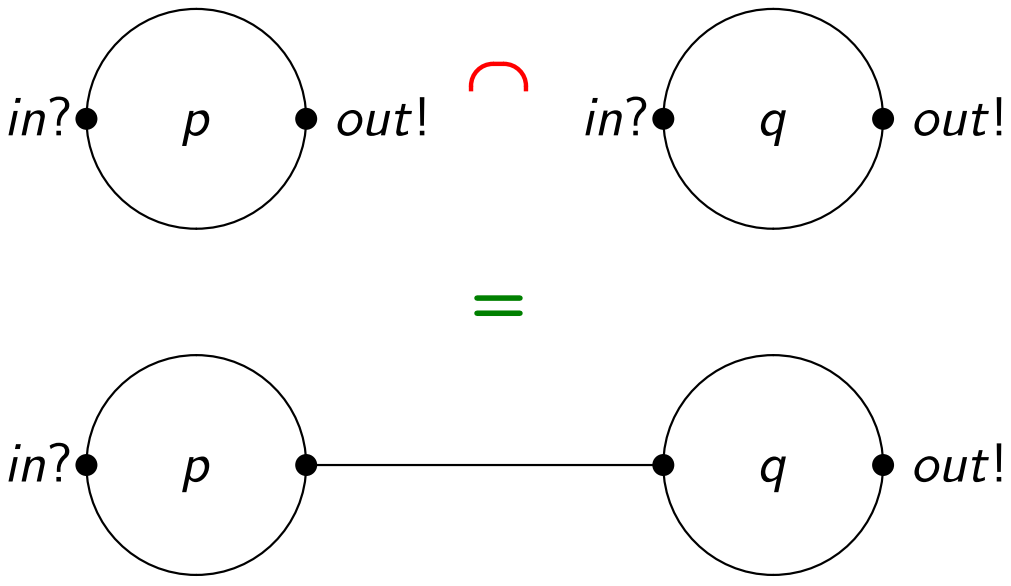
- Example: Maximum of two inputs

$$\begin{array}{l} in?x \to (in?y \to \\ \quad (\quad x \leq y \to max!y \\ \quad\; + \\ \quad\quad y \leq x \to max!x \quad )) \end{array}$$



# Linking processes

Connect $p$'s output port to $q$'s input port:




**Definition:**

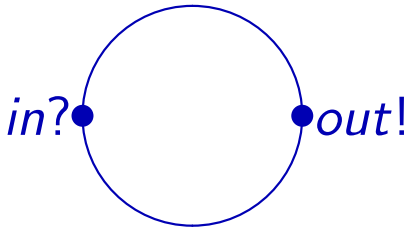
$$p \frown q = (p[c/out] \parallel q[c/in]) \setminus c$$

where $c$ is a fresh channel name

# Buffers

- **Definition:**

$$B \stackrel{\text{def}}{=} in?x \rightarrow (out!x \rightarrow B)$$




- $n$-ary buffer

$$\underbrace{B \frown B \frown \ldots \frown B}_{n \text{ times}}$$

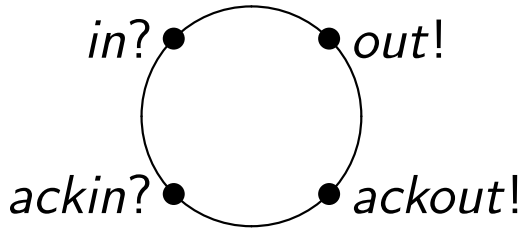
- Exercise: Draw the transition system for $B \frown B$

Remember: $p \frown q = (p[c/out] \parallel q[c/in]) \setminus c$

# Buffer with acknowledgements

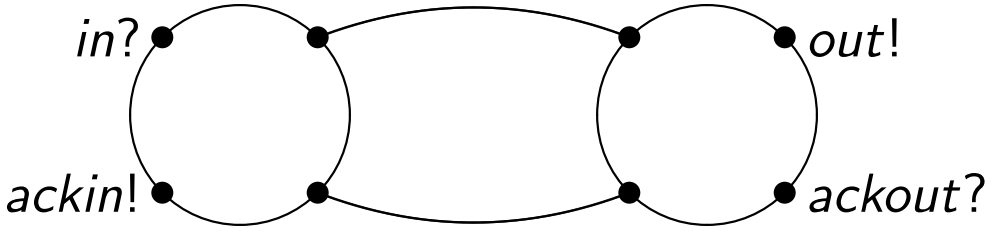
- **Definition:**

$$D \stackrel{\text{def}}{=} in?x \rightarrow out!x \rightarrow ackout? \rightarrow ackin! \rightarrow D$$




- Chaining now establishes two links:

$D \Cap D$

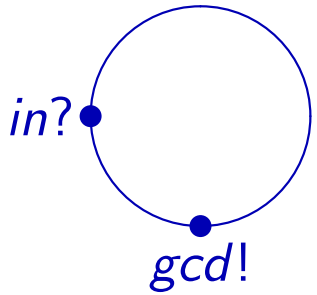



- How would this differ from the following process?

$$D' \stackrel{\text{def}}{=} in?x \rightarrow ackin! \rightarrow out!x \rightarrow ackout? \rightarrow D'$$

# Euclid's algorithm in CCS

**Interface:**




**Implementation:**

$$
\begin{aligned}
E(x,y) \quad \stackrel{\text{def}}{=} \quad & x = y \to gcd!x \to \mathbf{nil} \\
+ \quad & x < y \to E(x, y-x) \\
+ \quad & y < x \to E(x-y, x)
\end{aligned}
$$

$$
Euclid \stackrel{\text{def}}{=} in?x \to in?y \to E(x,y)
$$

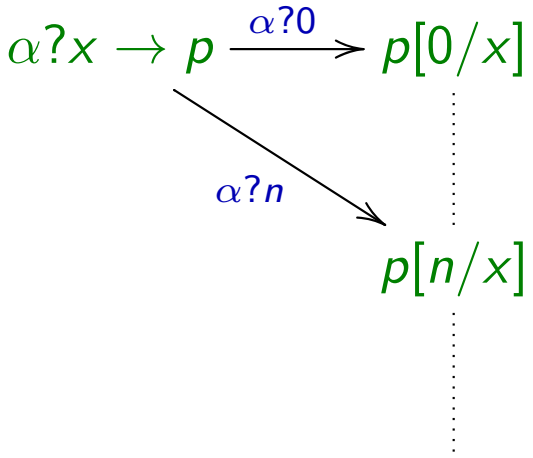
# Euclid's algorithm in CCS (without parameterized processes)

$$
\begin{aligned}
\mathit{Step} \stackrel{\text{def}}{=}\ & \mathit{in}?x \rightarrow \\
& \mathit{in}?y \rightarrow \\
& \quad (x = y \rightarrow \mathit{gcd}!x \rightarrow \mathbf{nil}) \\
& + \\
& \quad (x < y \rightarrow \mathit{out}!x \rightarrow \mathit{out}!(y - x) \rightarrow \mathbf{nil}) \\
& + \\
& \quad (y < x \rightarrow \mathit{out}!(x - y) \rightarrow \mathit{out}!y \rightarrow \mathbf{nil})
\end{aligned}
$$

$$
\mathit{Euclid} \stackrel{\text{def}}{=} \mathit{Step}^{\frown}\, \mathit{Euclid}
$$

# Towards a more basic language

- Transitions for value passing carry labels $\tau$, $a?n$, $a!n$




- This suggests introducing prefix $\alpha?n.p$ (as well as $\alpha!n.p$) and view $\alpha?x \to p$ as a sum $\sum_n \alpha?n.p[n/x]$ ← infinite sum
- View $\alpha?n$ and $\alpha!n$ as complementary actions
- Synchronization can only occur on complementary actions

# Pure CCS

- Actions: $a$, $b$, $c$, ...
- Complementary actions: $\overline{a}$, $\overline{b}$, $\overline{c}$, ...
- Internal action: $\tau$
- Notational convention: $\overline{\overline{a}} = a$
- Processes:

| | | | | |
|---|---|---|---|---|
| $p$ | ::= | $\lambda.p$ | prefix | $\lambda$ ranges over $\tau$, $a$, $\overline{a}$ for any action label $a$ |
| | \| | $\sum_{i \in I} p_i$ | sum | $I$ is an indexing set |
| | \| | $p_0 \parallel p_1$ | parallel | |
| | \| | $p \backslash L$ | restriction | $L$ a set of channel identifiers |
| | \| | $p[f]$ | relabelling | $f$ a function on channel identifiers |
| | \| | $P$ | | process identifier |

- Process definitions:

$$P \stackrel{\text{def}}{=} p$$

# Transition rules for Pure CCS

- **Nil process** no rules
- **Guarded processes**

$$\lambda.p \xrightarrow{\lambda} p$$

- **Sum**

$$\frac{p_j \xrightarrow{\lambda} p' \qquad j \in I}{\sum_{i \in I} p_i \to \lambda p'_0}$$

- **Parallel composition**

$$\frac{p_0 \xrightarrow{\lambda} p'_0}{p_0 \parallel p_1 \xrightarrow{\lambda} p'_0 \parallel p_1} \qquad \frac{p_1 \xrightarrow{\lambda} p'_1}{p_0 \parallel p_1 \xrightarrow{\lambda} p_0 \parallel p'_1}$$

$$\frac{p_0 \xrightarrow{a} p'_0 \qquad p_1 \xrightarrow{\overline{a}} p'_1}{p_0 \parallel p_1 \xrightarrow{\tau} p'_0 \parallel p'_1}$$

- **Restriction**

$$\frac{p \xrightarrow{\lambda} p' \qquad \lambda \notin L \cup \overline{L}}{p \setminus L \xrightarrow{\lambda} p' \setminus L} \quad \text{where } \overline{L} = \{\overline{a} \mid a \in L\}$$

- **Relabelling**

$$\frac{p \xrightarrow{\lambda} p'}{p[f] \xrightarrow{f(\lambda)} p'[f]}$$

  where $f$ is a function such that $f(\tau) = \tau$ and $f(\overline{a}) = \overline{f(a)}$

- **Identifiers**

$$\frac{p \xrightarrow{\lambda} p' \qquad P \stackrel{\text{def}}{=} p}{P \xrightarrow{\lambda} p'}$$