

Topics in Concurrency

Lecture 2

Glynn Winskel

21 January 2020

The Calculus of Communicating Systems (CCS)

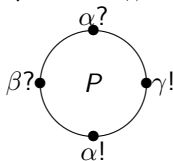
- Introduced by Robin Milner in 1980
- First process calculus developed with its operational semantics
- Supports algebraic reasoning about equivalence
- Simplifies Dijkstra's Guarded Command Language by removing the store (store locations can be encoded as processes)

Action and communication

- Processes communicate values (numbers) on channels.
- Communication is synchronous and between two processes.
- a is an arithmetic expression. Evaluation is written $a \rightarrow n$
- Input: $\alpha?x$
- Output: $\alpha!a$
- **Silent** action: τ . Internal to the process.
- We will use λ to range over all the kinds of action.

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 α, β and outputs on α, γ .



- 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
		$(\tau \rightarrow p)$	silent/internal action
		$(\alpha!a \rightarrow p)$	output
		$(\alpha?x \rightarrow p)$	input
		$(b \rightarrow p)$	Boolean guard
		$p_0 + p_1$	non-deterministic choice
		$p_0 \parallel p_1$	parallel composition
		$p \setminus L$	restriction (L a set of channel identifiers)
		$p[f]$	relabelling (f a function on channel identifiers)
		$P(a_1, \dots, a_k)$	process identifier

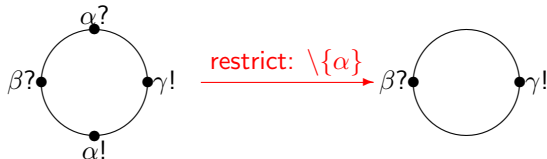
- Process definitions:

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

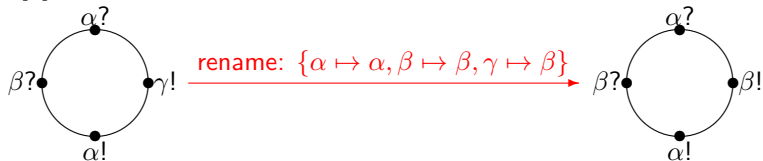
(free variables of $p \subseteq \{x_1, \dots, x_k\}$)

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**

Silent action

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

Output

$$\frac{a \longrightarrow n}{(\alpha!a \rightarrow p) \xrightarrow{\alpha!n} p}$$

Input

$$(\alpha?x \rightarrow p) \xrightarrow{\alpha?n} p[n/x]$$

Boolean

$$\frac{b \rightarrow \text{true} \quad p \xrightarrow{\lambda} p'}{(b \rightarrow p) \xrightarrow{\lambda} p'}$$

- Sum

$$\frac{p_0 \xrightarrow{\lambda} p'_0}{p_0 + p_1 \xrightarrow{\lambda} p'_0}$$

$$\frac{p_1 \xrightarrow{\lambda} p'_1}{p_0 + p_1 \xrightarrow{\lambda} p'_1}$$

- Parallel composition

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

$$\frac{p_0 \xrightarrow{\alpha?n} p'_0 \quad p_1 \xrightarrow{\alpha!n} p'_1}{p_0 \parallel p_1 \xrightarrow{\tau} p'_0 \parallel p'_1}$$

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

$$\frac{p_0 \xrightarrow{\alpha!n} p'_0 \quad p_1 \xrightarrow{\alpha?n} p'_1}{p_0 \parallel p_1 \xrightarrow{\tau} p'_0 \parallel p'_1}$$

- **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**

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

- **Nil process** no rules

A derivation from the operational semantics

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

A derivation from the operational semantics

$$\frac{\frac{(\alpha!3 \rightarrow \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil}}{(\alpha!3 \rightarrow \mathbf{nil}) + P \xrightarrow{\alpha!3} \mathbf{nil}}}{\frac{((\alpha!3 \rightarrow \mathbf{nil}) + P) \parallel (\tau \rightarrow \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil} \parallel (\tau \rightarrow \mathbf{nil}) \quad (\alpha?x \rightarrow \mathbf{nil}) \xrightarrow{\alpha?3} \mathbf{nil}}{(((\alpha!3 \rightarrow \mathbf{nil}) + P) \parallel (\tau \rightarrow \mathbf{nil})) \parallel (\alpha?x \rightarrow \mathbf{nil}) \xrightarrow{\tau} (\mathbf{nil} \parallel (\tau \rightarrow \mathbf{nil})) \parallel \mathbf{nil}}}$$

A derivation from the operational semantics

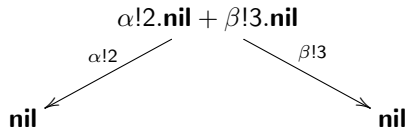
$$\begin{array}{c}
 \frac{(\alpha!3 \rightarrow \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil}}{(\alpha!3 \rightarrow \mathbf{nil}) + P \xrightarrow{\alpha!3} \mathbf{nil}} \\
 \hline
 \frac{((\alpha!3 \rightarrow \mathbf{nil}) + P) \parallel (\tau \rightarrow \mathbf{nil}) \xrightarrow{\alpha!3} \mathbf{nil} \parallel (\tau \rightarrow \mathbf{nil}) \quad (\alpha?x \rightarrow \mathbf{nil}) \xrightarrow{\alpha?3} \mathbf{nil}}{(((\alpha!3 \rightarrow \mathbf{nil}) + P) \parallel (\tau \rightarrow \mathbf{nil})) \parallel (\alpha?x \rightarrow \mathbf{nil}) \xrightarrow{\tau} (\mathbf{nil} \parallel (\tau \rightarrow \mathbf{nil})) \parallel \mathbf{nil}} \\
 \hline
 (((\alpha!3 \rightarrow \mathbf{nil}) + P) \parallel \tau \rightarrow \mathbf{nil}) \parallel \alpha?x \rightarrow \mathbf{nil} \setminus \{\alpha\} \xrightarrow{\tau} ((\mathbf{nil} \parallel \tau \rightarrow \mathbf{nil}) \parallel \mathbf{nil}) \setminus \{\alpha\}
 \end{array}$$

Further examples

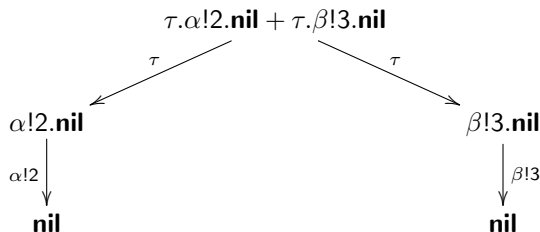
(Write . for $\rightarrow$)

Each step justified by a derivation:

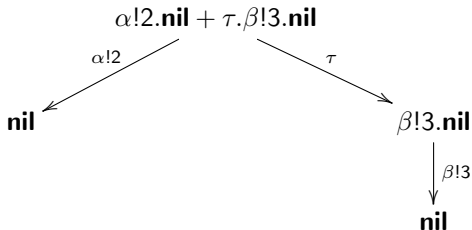
- External choice



- Internal choice



- Mixed choice



- Exercise:

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

- Exercise:

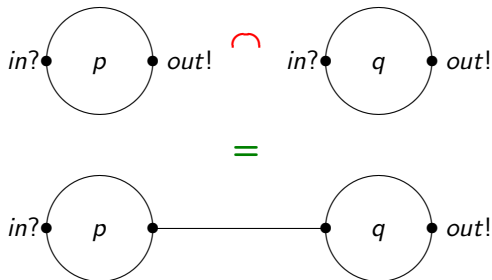
$$(\alpha!3.\mathbf{nil} \parallel \alpha?x.\beta!x.\mathbf{nil}) \setminus \{\alpha\}$$

- Exercise:

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

Linking processes

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



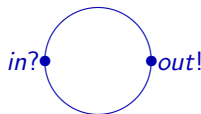
Definition:

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

where c is a **fresh** channel name

- **Definition:**

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



- n -ary buffer

$$\underbrace{B \cap B \cap \dots \cap B}_{n \text{ times}}$$

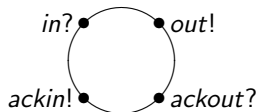
- Exercise: Draw the transition system for $B \cap B$

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

Buffer with acknowledgements

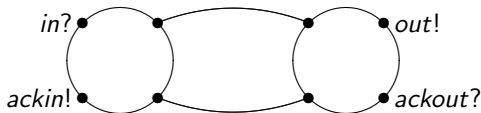
- **Definition:**

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



- Chaining now establishes two links:

$$D \bowtie D$$

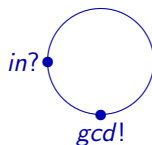


- How would this differ with the following process?

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

Euclid's algorithm in CCS

Interface:



Implementation:

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

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

Euclid's algorithm in CCS (without parameterized processes)

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

$$\text{Euclid} \stackrel{\text{def}}{=} \text{Step}^{\cap} \text{Euclid}$$