Communication and Concurrency
Lecture 1

Colin Stirling (cps)

School of Informatics

September 19th 2013

First example

A clock that perpetually ticks

$$\mathtt{Cl} \stackrel{\mathrm{def}}{=} \mathtt{tick.Cl}$$

- $\mathtt{tick}$ action name
- $\mathtt{Cl}$ process name
- $\stackrel{\mathrm{def}}{=}$ ties a process name to a process expression
- $\mathtt{tick.Cl}$ process expression
- $\mathtt{.}$ prefix operator

Behaviour: transitions

Behaviour of processes is captured by transitions

$$E \xrightarrow{a} F$$

Goal-directed rules for deriving transitions

- axiom (.)

$$\mathrm{R}(.) \quad a.E \xrightarrow{a} E$$

Behaviour: transitions

Behaviour of processes is captured by transitions

$$E \xrightarrow{a} F$$

Goal-directed rules for deriving transitions

- axiom (.)

$$\mathrm{R}(.) \quad a.E \xrightarrow{a} E$$

- $\stackrel{\mathrm{def}}{=}$

$$\mathrm{R}(\stackrel{\mathrm{def}}{=}) \quad \frac{P \xrightarrow{a} F}{E \xrightarrow{a} F} \; P \stackrel{\mathrm{def}}{=} E$$

Behaviour: transitions

Behaviour of processes is captured by transitions

$$E \xrightarrow{a} F$$

Goal-directed rules for deriving transitions

- axiom (.)

$$\mathrm{R}(.) \quad a.E \xrightarrow{a} E$$

- $\stackrel{\mathrm{def}}{=}$

$$\mathrm{R}(\stackrel{\mathrm{def}}{=}) \quad \frac{P \xrightarrow{a} F}{E \xrightarrow{a} F} \; P \stackrel{\mathrm{def}}{=} E$$

Example

$$\mathtt{Cl} \xrightarrow{\mathtt{tick}} \mathtt{Cl}$$

Behaviour: transition graphs

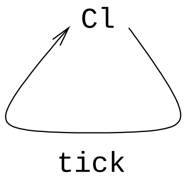


Figure: The transition graph for Cl

Behaviour: transition graphs

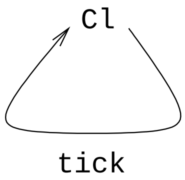


Figure: The transition graph for Cl

Labelled graph

- vertices: process expressions
- labelled edges: transitions
- Each derivable transition of a vertex is depicted
- Abstract from the derivations of transitions

Interlude: exercise

Draw the transition graphs for the following clocks

1. $\mathtt{Cl}_1 \stackrel{\text{def}}{=} \mathtt{tick.tock.Cl}_1$

Interlude: exercise

Draw the transition graphs for the following clocks

1. $\texttt{Cl}_1 \stackrel{\text{def}}{=} \texttt{tick.tock.Cl}_1$
2. $\texttt{Cl}_2 \stackrel{\text{def}}{=} \texttt{tick.tick.Cl}_2$

Interlude: exercise

Draw the transition graphs for the following clocks

1. $\mathtt{Cl}_1 \stackrel{\text{def}}{=} \mathtt{tick.tock.Cl}_1$
2. $\mathtt{Cl}_2 \stackrel{\text{def}}{=} \mathtt{tick.tick.Cl}_2$
3. $\mathtt{Cl}_3 \stackrel{\text{def}}{=} \mathtt{tick.Cl}$

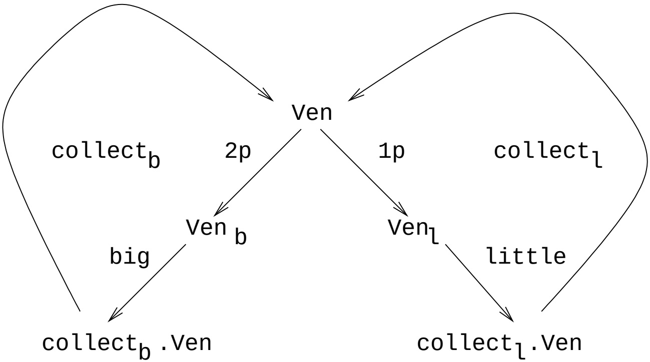
The + operator

$$
\begin{aligned}
\mathtt{Ven} &\stackrel{\mathrm{def}}{=} \mathtt{2p.Ven_b} + \mathtt{1p.Ven_l} \\
\mathtt{Ven_b} &\stackrel{\mathrm{def}}{=} \mathtt{big.collect_b.Ven} \\
\mathtt{Ven_l} &\stackrel{\mathrm{def}}{=} \mathtt{little.collect_l.Ven}
\end{aligned}
$$

Transition Rule

$$
\mathrm{R}(+) \quad \frac{E_1 + E_2 \xrightarrow{a} F}{E_1 \xrightarrow{a} F} \qquad \frac{E_1 + E_2 \xrightarrow{a} F}{E_2 \xrightarrow{a} F}
$$

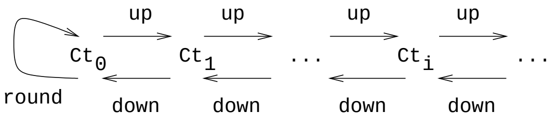
Transition Graph



Generalising: indexed definitions

$$\begin{aligned} \mathtt{Ct}_0 &\stackrel{\text{def}}{=} \mathtt{up}.\mathtt{Ct}_1 + \mathtt{round}.\mathtt{Ct}_0 \\ \mathtt{Ct}_{\mathtt{i}+1} &\stackrel{\text{def}}{=} \mathtt{up}.\mathtt{Ct}_{\mathtt{i}+2} + \mathtt{down}.\mathtt{Ct}_{\mathtt{i}} \end{aligned}$$

$$\cfrac{\cfrac{\mathtt{Ct}_3 \xrightarrow{\mathtt{up}} \mathtt{Ct}_4}{\mathtt{up}.\mathtt{Ct}_4 + \mathtt{down}.\mathtt{Ct}_2 \xrightarrow{\mathtt{up}} \mathtt{Ct}_4}}{\mathtt{up}.\mathtt{Ct}_4 \xrightarrow{\mathtt{up}} \mathtt{Ct}_4}$$



Generalising: indexed sums

$\sum\{E_i : i \in I\}$ I indexing set
($E_1 + E_2$ abbreviates $\sum\{E_i : i \in \{1, 2\}\}$)

Generalising: indexed sums

$\sum\{E_i : i \in I\}$ I indexing set
$(E_1 + E_2$ abbreviates $\sum\{E_i : i \in \{1,2\}\})$

$$\mathtt{Reg}'_{\mathtt{i}} \stackrel{\text{def}}{=} \mathtt{read}_{\mathtt{i}}.\mathtt{Reg}'_{\mathtt{i}} + \sum\{\mathtt{write}_j.\mathtt{Reg}'_j : j \in \mathbb{N}\}$$

Generalising: indexed sums

$\sum\{E_i : i \in I\}$ I indexing set
($E_1 + E_2$ abbreviates $\sum\{E_i : i \in \{1,2\}\}$)

$$\mathtt{Reg}'_{\mathtt{i}} \stackrel{\mathrm{def}}{=} \mathtt{read}_{\mathtt{i}}.\mathtt{Reg}'_{\mathtt{i}} + \sum\{\mathtt{write}_j.\mathtt{Reg}'_j : j \in \mathbb{N}\}$$

Transition Rule for Σ

$$\mathrm{R}(\sum) \quad \frac{\sum\{E_i : i \in I\} \xrightarrow{a} F}{E_j \xrightarrow{a} F} \; j \in I$$

Generalising: indexed sums

$\sum\{E_i \ : \ i \in I\}$ I indexing set
($E_1 + E_2$ abbreviates $\sum\{E_i \ : \ i \in \{1,2\}\}$)

$$\mathtt{Reg}'_{\mathtt{i}} \stackrel{\text{def}}{=} \mathtt{read}_{\mathtt{i}}.\mathtt{Reg}'_{\mathtt{i}} + \sum\{\mathtt{write}_j.\mathtt{Reg}'_j \ : \ j \in \mathbb{N}\}$$

Transition Rule for Σ

$$\mathrm{R}(\textstyle\sum) \quad \frac{\sum\{E_i \ : \ i \in I\} \xrightarrow{a} F}{E_j \xrightarrow{a} F} \; j \in I$$

Special Case $\sum\{E_i \ : \ i \in \emptyset\}$ abbreviated to 0 “nil”

Generalising: parameterized actions

- input of data at port a, $a(x).E$
  $a(x)$ binds free occurrences of x in E. Port a represents $\{a(v) : v \in D\}$ where D is a family of data values

Generalising: parameterized actions

- input of data at port a, $a(x).E$
  $a(x)$ binds free occurrences of x in E. Port a represents $\{a(v) : v \in D\}$ where D is a family of data values
- output of data at port a, $\overline{a}(e).E$
  where e is a data expression.

Generalising: parameterized actions

- input of data at port a, $a(x).E$
  $a(x)$ binds free occurrences of x in E. Port a represents $\{a(v) : v \in D\}$ where D is a family of data values
- output of data at port a, $\overline{a}(e).E$
  where e is a data expression.
- Transition Rules: depend on extra machinery for expression evaluation. Let $\mathrm{Val}(e)$ be data value in D (if there is one) to which e evaluates

Generalising: parameterized actions

- input of data at port a, $a(x).E$
  $a(x)$ binds free occurrences of x in E. Port a represents $\{a(v) : v \in D\}$ where D is a family of data values
- output of data at port a, $\overline{a}(e).E$
  where e is a data expression.
- Transition Rules: depend on extra machinery for expression evaluation. Let Val(e) be data value in D (if there is one) to which e evaluates
- $\mathrm{R(in)} \quad a(x).E \xrightarrow{a(v)} E\{v/x\} \quad \text{if} \;\; v \in D$
  where $\{v/x\}$ is substitution

Generalising: parameterized actions

- input of data at port a, $a(x).E$
  $a(x)$ binds free occurrences of x in E. Port a represents $\{a(v) : v \in D\}$ where D is a family of data values
- output of data at port a, $\overline{a}(e).E$
  where e is a data expression.
- Transition Rules: depend on extra machinery for expression evaluation. Let $\mathrm{Val}(e)$ be data value in D (if there is one) to which e evaluates
- $\mathrm{R(in)} \quad a(x).E \xrightarrow{a(v)} E\{v/x\} \quad \text{if} \;\; v \in D$
  where $\{v/x\}$ is substitution
- $\mathrm{R(out)} \quad \overline{a}(e).E \xrightarrow{\overline{a}(v)} E \quad \text{if } \mathrm{Val}(e) = v$

Examples

$$\mathrm{R(in)} \quad a(x).E \xrightarrow{a(v)} E\{v/x\} \quad \text{if} \;\; v \in D$$

$$\mathrm{R(out)} \quad \overline{a}(e).E \xrightarrow{\overline{a}(v)} E \quad \text{if } \mathrm{Val}(e) = v$$

Examples

$$\mathrm{R(in)} \quad a(x).E \xrightarrow{a(v)} E\{v/x\} \quad \text{if} \;\; v \in D$$

$$\mathrm{R(out)} \quad \overline{a}(e).E \xrightarrow{\overline{a}(v)} E \quad \text{if } \mathrm{Val}(e) = v$$

A Copier: $\mathtt{Cop} \stackrel{\mathrm{def}}{=} \mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop}$

$$\frac{\mathtt{Cop} \xrightarrow{\mathtt{in}(v)} \overline{\mathtt{out}}(v).\mathtt{Cop}}{\mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop} \xrightarrow{\mathtt{in}(v)} \overline{\mathtt{out}}(v).\mathtt{Cop}}$$

Examples

$\mathrm{R(in)} \quad a(x).E \xrightarrow{a(v)} E\{v/x\} \quad \text{if} \;\; v \in D$

$\mathrm{R(out)} \quad \overline{a}(e).E \xrightarrow{\overline{a}(v)} E \quad \text{if Val}(e) = v$

A Copier: $\mathtt{Cop} \stackrel{\text{def}}{=} \mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop}$

$$\frac{\mathtt{Cop} \xrightarrow{\mathtt{in}(v)} \overline{\mathtt{out}}(v).\mathtt{Cop}}{\mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop} \xrightarrow{\mathtt{in}(v)} \overline{\mathtt{out}}(v).\mathtt{Cop}}$$

A Register: $\mathtt{Reg_i} \stackrel{\text{def}}{=} \overline{\mathtt{read}}(i).\mathtt{Reg_i} + \mathtt{write}(x).\mathtt{Reg}_x$

$$\frac{\dfrac{\mathtt{Reg}_5 \xrightarrow{\mathtt{write}(3)} \mathtt{Reg}_3}{\overline{\mathtt{read}}(5).\mathtt{Reg}_5 + \mathtt{write}(x).\mathtt{Reg}_x \xrightarrow{\mathtt{write}(3)} \mathtt{Reg}_3}}{\mathtt{write}(x).\mathtt{Reg}_x \xrightarrow{\mathtt{write}(3)} \mathtt{Reg}_3}$$

Exercise

Assume that the space of values consists of two elements, 0 and 1.

Draw transition graphs for the following three copiers

1. $\mathtt{Cop} \stackrel{\text{def}}{=} \mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop}$

Exercise

Assume that the space of values consists of two elements, 0 and 1. Draw transition graphs for the following three copiers

1. $\mathtt{Cop} \stackrel{\text{def}}{=} \mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop}$
2. $\mathtt{Cop}_1 \stackrel{\text{def}}{=} \mathtt{in}(x).\mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop}_1$

Exercise

Assume that the space of values consists of two elements, 0 and 1.

Draw transition graphs for the following three copiers

1. $\mathtt{Cop} \stackrel{\mathrm{def}}{=} \mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop}$
2. $\mathtt{Cop}_1 \stackrel{\mathrm{def}}{=} \mathtt{in}(x).\mathtt{in}(x).\overline{\mathtt{out}}(x).\mathtt{Cop}_1$
3. $\mathtt{Cop}_2 \stackrel{\mathrm{def}}{=} \mathtt{in}(x).\overline{\mathtt{out}}(x).\overline{\mathtt{out}}(x).\mathtt{Cop}_2$

Summary

- Introduction of process expressions, process combinators

Summary

- Introduction of process expressions, process combinators
- Derivation of transitions between expressions

Summary

- Introduction of process expressions, process combinators
- Derivation of transitions between expressions
- Abstraction of derivations into transition graphs

Summary

- Introduction of process expressions, process combinators
- Derivation of transitions between expressions
- Abstraction of derivations into transition graphs
- Background Reading: Chapter 1 of
  R. Milner, *Communication and Concurrency*, Prentice-Hall