Formal Techniques for Software Engineering: CCS: A Calculus for Communicating Systems

Rocco De Nicola

IMT Institute for Advanced Studies, Lucca
rocco.denicola@imtlucca.it

June 2013

Lesson 10

What have we done

1. Importance of formal methods for proving correctness of concurrent systems
2. Labelled Transition Systems (LTS) as formal models
3. Operators of Process Description Languages (PDL) for modeling systems composition and interaction.
4. Structural Operational Semantics (SOS) for modelling operators and building transition systems corresponding to composite systems.
5. Behavioural Equivalences to abstract from unwanted details of the modelled systems
6. A number of PDL obtained by careful selection of Operators and equivalences
7. Alternative approaches (Equational and Denotational) to defining the Semantics of PDL

What's next

Now we concentrate on one language (CCS), study its theory more in depth and consider a number of small examples (case studies).

CCS Basics

Sequential Fragment

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

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

Parallelism and Renaming

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

CCS Basics

Sequential Fragment

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

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

Parallelism and Renaming

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

CCS Basics

Sequential Fragment

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

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

Parallelism and Renaming

- 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 $\overline{\overline{a}} = a$
- $Act = \mathcal{L} \cup \{\tau\}$ is the set of actions where
  - τ is the internal or silent action

  (e.g. τ, *tea*, $\overline{coffee}$ are actions)
- $\mathcal{K}$ is a set of process names (constants) (e.g. CM).

Definition of CCS (expressions)

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

- $f(\tau) = \tau$
- $f(\overline{a}) = \overline{f(a)}$

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

Notation

$P_1 + P_2 = \sum_{i \in \{1,2\}} P_i$ $\qquad$ $Nil = 0 = \sum_{i \in \emptyset} P_i$

Precedence

Precedence

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

Example: $R + a.P|b.Q \setminus L$ means $R + \big((a.P)|(b.(Q \setminus L))\big)$.

Definition of CCS (defining equations)

CCS program

A collection of defining equations of the form

$$K \triangleq P$$

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. $A \triangleq \overline{a}.A \mid A$.

Structural Operational Semantics for CCS

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 $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$:

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

$$\text{RULE} \quad \frac{\textit{premises}}{\textit{conclusion}} \quad \textit{conditions}$$

SOS rules for CCS ($\alpha \in Act$, $a \in \mathcal{L}$)

ACT $\dfrac{}{\alpha.P \xrightarrow{\alpha} P}$ SUM$_j$ $\dfrac{P_j \xrightarrow{\alpha} P'_j}{\sum_{i \in I} P_i \xrightarrow{\alpha} P'_j}$ $j \in I$

COM1 $\dfrac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$ COM2 $\dfrac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$

COM3 $\dfrac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$

RES $\dfrac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$ $\alpha, \overline{\alpha} \notin L$ REL $\dfrac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$

CON $\dfrac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'}$ $K \triangleq P$

Deriving Transitions in CCS

Let $A \triangleq a.A$. Then

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

Why?

Deriving Transitions in CCS

Let $A \triangleq a.A$. Then

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

Why?

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

Deriving Transitions in CCS

Let $A \triangleq a.A$. Then

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

Why?

$$\text{REL}\ \dfrac{\text{COM1}\ \dfrac{}{(A \,|\, \overline{a}.Nil) \,|\, b.Nil \xrightarrow{a} (A \,|\, \overline{a}.Nil) \,|\, b.Nil}}{\big((A \,|\, \overline{a}.Nil) \,|\, b.Nil\big)[c/a] \xrightarrow{c} \big((A \,|\, \overline{a}.Nil) \,|\, b.Nil\big)[c/a]}$$

Deriving Transitions in CCS

Let $A \triangleq a.A$. Then

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

Why?

$$\text{REL}\ \dfrac{\text{COM1}\ \dfrac{\text{COM1}\ \dfrac{}{A \mid \overline{a}.Nil \xrightarrow{a} A \mid \overline{a}.Nil}}{(A \mid \overline{a}.Nil) \mid b.Nil \xrightarrow{a} (A \mid \overline{a}.Nil) \mid b.Nil}}{\big((A \mid \overline{a}.Nil) \mid b.Nil\big)[c/a] \xrightarrow{c} \big((A \mid \overline{a}.Nil) \mid b.Nil\big)[c/a]}$$

Deriving Transitions in CCS

Let $A \triangleq a.A$. Then

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

Why?

$$\text{REL}\ \dfrac{\text{COM1}\ \dfrac{\text{COM1}\ \dfrac{\text{CON}\ \dfrac{}{A \xrightarrow{a} A}\, A \triangleq a.A}{A \mid \overline{a}.Nil \xrightarrow{a} A \mid \overline{a}.Nil}}{(A \mid \overline{a}.Nil) \mid b.Nil \xrightarrow{a} (A \mid \overline{a}.Nil) \mid b.Nil}}{\big((A \mid \overline{a}.Nil) \mid b.Nil\big)[c/a] \xrightarrow{c} \big((A \mid \overline{a}.Nil) \mid b.Nil\big)[c/a]}$$

Deriving Transitions in CCS

Let $A \triangleq a.A$. Then

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

Why?

$$\text{REL}\ \dfrac{\text{COM1}\ \dfrac{\text{COM1}\ \dfrac{\text{CON}\ \dfrac{\text{ACT}\ \dfrac{}{a.A \xrightarrow{a} A}}{A \xrightarrow{a} A}\, A \triangleq a.A}{A \mid \overline{a}.Nil \xrightarrow{a} A \mid \overline{a}.Nil}}{(A \mid \overline{a}.Nil) \mid b.Nil \xrightarrow{a} (A \mid \overline{a}.Nil) \mid b.Nil}}{\big((A \mid \overline{a}.Nil) \mid b.Nil\big)[c/a] \xrightarrow{c} \big((A \mid \overline{a}.Nil) \mid b.Nil\big)[c/a]}$$

LTS of the Process $a.Nil \mid \overline{a}.Nil$

$$\begin{tikzcd}
 & a.Nil \mid \overline{a}.Nil \arrow[dl, "a"'] \arrow[dr, "\overline{a}"] \arrow[dd, "\tau"] & \\
Nil \mid \overline{a}.Nil \arrow[dr, "\overline{a}"] & & a.Nil \mid Nil \arrow[dl, "a"'] \\
 & Nil \mid Nil &
\end{tikzcd}$$

Puzzle Time: 50 Prisoners Problem

Solve the problem and write down the solution as a process

50 prisoners kept in separate cells got a chance are told that

- From time to time one of them would be taken to a special room (in no particular order, possibly multiple many consecutive times, but with a fair schedule to avoid infinite wait) and then back to the cell.
- The room is completely empty except for an on/off switch a lamp (the light is not visible from outside).
- At any time, if any of them says that all the prisoners have already entered the room at least once and this is true, all prisoners will be released (but if it is false, the play ends and they are life sentenced).

Before the challenge starts, the prisoners have the possibility to discuss together some "protocol" to follow.

Is there a winning strategy for the prisoners?

Note that the initial state of the light in the room is not known.

First Attempt (assume the light is initially ON)

The Light (initially ON)

Light $\triangleq$ *turnOff*.*turnOn*.Light (can be read as $recX.turnOff.turnOn.X$)

One Counting-Down Prisoner

$$\begin{aligned} C_{i+1} &\triangleq \overline{turnOff}.C_i + \tau.C_{i+1} \\ C_0 &\triangleq \overline{freeAll} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} P &\triangleq \overline{turnOn}.IP + \tau.P \\ IP &\triangleq \tau.IP \qquad \textit{(Idle Prisoner)} \end{aligned}$$

Prison

$(Light \mid C_{50} \mid P \mid \ldots \mid P) \setminus \{turnOn, turnOff\}$

First Attempt (assume the light is initially ON)

The Light (initially ON)

Light $\triangleq$ *turnOff*.*turnOn*.Light (can be read as $recX.turnOff.turnOn.X$)

One Counting-Down Prisoner

$$
\begin{aligned}
C_{i+1} &\triangleq \overline{turnOff}.C_i + \tau.C_{i+1} \\
C_0 &\triangleq \overline{freeAll}
\end{aligned}
$$

49 Generic Prisoners

$$
\begin{aligned}
P &\triangleq \overline{turnOn}.IP + \tau.P \\
IP &\triangleq \tau.IP \qquad \textit{(Idle Prisoner)}
\end{aligned}
$$

Prison

$(\text{Light} \mid C_{50} \mid P \mid \ldots \mid P) \setminus \{turnOn, turnOff\}$

First Attempt (assume the light is initially ON)

The Light (initially ON)

Light $\triangleq$ *turnOff* .*turnOn*.Light (can be read as *recX*.*turnOff* .*turnOn*.*X*)

One Counting-Down Prisoner

$$\begin{aligned} C_{i+1} &\triangleq \overline{turnOff}.C_i + \tau.C_{i+1} \\ C_0 &\triangleq \overline{freeAll} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} P &\triangleq \overline{turnOn}.IP + \tau.P \\ IP &\triangleq \tau.IP \end{aligned}$$
(Idle Prisoner)

Prison

$(\text{Light} \mid C_{50} \mid P \mid \ldots \mid P) \setminus \{turnOn, turnOff\}$

First Attempt (assume the light is initially ON)

The Light (initially ON)

$\text{Light} \triangleq \mathit{turnOff}.\mathit{turnOn}.\text{Light}$ (can be read as $\mathit{rec}X.\mathit{turnOff}.\mathit{turnOn}.X$)

One Counting-Down Prisoner

$$\begin{aligned} C_{i+1} &\triangleq \overline{\mathit{turnOff}}.C_i + \tau.C_{i+1} \\ C_0 &\triangleq \overline{\mathit{freeAll}} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} P &\triangleq \overline{\mathit{turnOn}}.IP + \tau.P \\ IP &\triangleq \tau.IP \qquad \textit{(Idle Prisoner)} \end{aligned}$$

Prison

$(\text{Light} \mid C_{50} \mid P \mid \ldots \mid P) \setminus \{\mathit{turnOn}, \mathit{turnOff}\}$

Problems

Prisoners can just perform τs and never interact with the light

The Light (initially ON)

$\text{Light} \triangleq \mathit{turnOff}.\mathit{turnOn}.\text{Light}$

One Counting-Down Prisoner

$$\begin{aligned} C_{i+1} &\triangleq \overline{\mathit{turnOff}}.C_i + \overline{\mathit{turnOn}}.\overline{\mathit{turnOff}}.C_{i+1} \\ C_0 &\triangleq \overline{\mathit{freeAll}} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} P &\triangleq \overline{\mathit{turnOn}}.IP + \overline{\mathit{turnOff}}.\overline{\mathit{turnOn}}.P \\ IP &\triangleq \tau.IP \qquad \textit{(Idle Prisoner)} \end{aligned}$$

Prison

$(\text{Light} \mid C_{50} \mid P \mid \ldots \mid P) \setminus \{\mathit{turnOn}, \mathit{turnOff}\}$

Problems

Prisoners can just perform τs and never interact with the light

The Light (initially ON)

Light $\triangleq$ *turnOff* .*turnOn*.Light

One Counting-Down Prisoner

$$\begin{aligned} C_{i+1} &\triangleq \overline{turnOff}.C_i + \overline{turnOn}.\overline{turnOff}.C_{i+1} \\ C_0 &\triangleq \overline{freeAll} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} P &\triangleq \overline{turnOn}.IP + \overline{turnOff}.\overline{turnOn}.P \\ IP &\triangleq \tau.IP \qquad \textit{(Idle Prisoner)} \end{aligned}$$

Prison

$(\text{Light} \mid C_{50} \mid P \mid \ldots \mid P) \setminus \{turnOn, turnOff\}$

Problems

Prisoners can just perform τs and never interact with the light

The Light (initially ON)

Light $\triangleq$ *turnOff* .*turnOn*.Light

One Counting-Down Prisoner

$$\begin{aligned} C_{i+1} &\triangleq \overline{turnOff}.C_i + \overline{turnOn}.\overline{turnOff}.C_{i+1} \\ C_0 &\triangleq \overline{freeAll} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} P &\triangleq \overline{turnOn}.IP + \overline{turnOff}.\overline{turnOn}.P \\ IP &\triangleq \tau.IP \qquad \textit{(Idle Prisoner)} \end{aligned}$$

Prison

$(\text{Light} \mid C_{50} \mid P \mid \ldots \mid P) \setminus \{turnOn, turnOff\}$

Problems

Prisoners can just perform τs and never interact with the light

The Light (initially ON)

Light $\triangleq$ *turnOff*.*turnOn*.Light

One Counting-Down Prisoner

$$\begin{array}{rcl} C_{i+1} & \triangleq & \overline{turnOff}.C_i + \overline{turnOn}.\overline{turnOff}.C_{i+1} \\ C_0 & \triangleq & \overline{freeAll} \end{array}$$

49 Generic Prisoners

$$\begin{array}{rcl} P & \triangleq & \overline{turnOn}.IP + \overline{turnOff}.\overline{turnOn}.P \\ IP & \triangleq & \tau.IP \qquad \textit{(Idle Prisoner)} \end{array}$$

Prison

$(\text{Light} \mid C_{50} \mid P \mid \ \dots \ \mid P) \setminus \{turnOn, turnOff\}$

Problems

The light must be accessed in mutual exclusion

The Room and The Light (initially ON)

$\mathrm{Room} \triangleq \mathit{roomIn}.\mathit{roomOut}.\mathrm{Room}$ $\qquad$ $\mathrm{Light} \triangleq \mathit{turnOff}.\mathit{turnOn}.\mathrm{Light}$

One Counting-Down Prisoner

$$\begin{aligned} \mathrm{C}_{i+1} &\triangleq \overline{\mathit{roomIn}}.(\overline{\mathit{turnOff}}.\overline{\mathit{roomOut}}.\mathrm{C}_i + \overline{\mathit{turnOn}}.\overline{\mathit{turnOff}}.\overline{\mathit{roomOut}}.\mathrm{C}_{i+1}) \\ \mathrm{C}_0 &\triangleq \overline{\mathit{freeAll}} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} \mathrm{P} &\triangleq \overline{\mathit{roomIn}}.(\overline{\mathit{turnOn}}.\overline{\mathit{roomOut}}.\mathrm{IP} + \overline{\mathit{turnOff}}.\overline{\mathit{turnOn}}.\overline{\mathit{roomOut}}.\mathrm{P}) \\ \mathrm{IP} &\triangleq \overline{\mathit{roomIn}}.\tau.\overline{\mathit{roomOut}}.\mathrm{IP} \end{aligned}$$

Prison

$$(\mathrm{Room} \mid \mathrm{Light} \mid \mathrm{C}_{50} \mid \mathrm{P} \mid \ldots \mid \mathrm{P}) \setminus \{\mathit{turnOn}, \mathit{turnOff}, \mathit{roomIn}, \mathit{roomOut}\}$$

Problems

The light must be accessed in mutual exclusion

The Room and The Light (initially ON)

$\mathrm{Room} \triangleq \mathit{roomIn}.\mathit{roomOut}.\mathrm{Room}$ $\qquad$ $\mathrm{Light} \triangleq \mathit{turnOff}.\mathit{turnOn}.\mathrm{Light}$

One Counting-Down Prisoner

$$\begin{aligned} \mathrm{C}_{i+1} &\triangleq \overline{\mathit{roomIn}}.(\overline{\mathit{turnOff}}.\overline{\mathit{roomOut}}.\mathrm{C}_i + \overline{\mathit{turnOn}}.\overline{\mathit{turnOff}}.\overline{\mathit{roomOut}}.\mathrm{C}_{i+1}) \\ \mathrm{C}_0 &\triangleq \overline{\mathit{freeAll}} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} \mathrm{P} &\triangleq \overline{\mathit{roomIn}}.(\overline{\mathit{turnOn}}.\overline{\mathit{roomOut}}.\mathrm{IP} + \overline{\mathit{turnOff}}.\overline{\mathit{turnOn}}.\overline{\mathit{roomOut}}.\mathrm{P}) \\ \mathrm{IP} &\triangleq \overline{\mathit{roomIn}}.\tau.\overline{\mathit{roomOut}}.\mathrm{IP} \end{aligned}$$

Prison

$$(\mathrm{Room} \mid \mathrm{Light} \mid \mathrm{C}_{50} \mid \mathrm{P} \mid \ldots \mid \mathrm{P}) \setminus \{\mathit{turnOn}, \mathit{turnOff}, \mathit{roomIn}, \mathit{roomOut}\}$$

Problems

The light must be accessed in mutual exclusion

The Room and The Light (initially ON)

$\mathrm{Room} \triangleq \mathit{roomIn}.\mathit{roomOut}.\mathrm{Room} \qquad \mathrm{Light} \triangleq \mathit{turnOff}.\mathit{turnOn}.\mathrm{Light}$

One Counting-Down Prisoner

$$\begin{aligned} \mathrm{C}_{i+1} &\triangleq \overline{\mathit{roomIn}}.(\overline{\mathit{turnOff}}.\overline{\mathit{roomOut}}.\mathrm{C}_i + \overline{\mathit{turnOn}}.\overline{\mathit{turnOff}}.\overline{\mathit{roomOut}}.\mathrm{C}_{i+1}) \\ \mathrm{C}_0 &\triangleq \overline{\mathit{freeAll}} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} \mathrm{P} &\triangleq \overline{\mathit{roomIn}}.(\overline{\mathit{turnOn}}.\overline{\mathit{roomOut}}.\mathrm{IP} + \overline{\mathit{turnOff}}.\overline{\mathit{turnOn}}.\overline{\mathit{roomOut}}.\mathrm{P}) \\ \mathrm{IP} &\triangleq \overline{\mathit{roomIn}}.\tau.\overline{\mathit{roomOut}}.\mathrm{IP} \end{aligned}$$

Prison

$(\mathrm{Room} \mid \mathrm{Light} \mid \mathrm{C}_{50} \mid \mathrm{P} \mid \ldots \mid \mathrm{P}) \setminus \{\mathit{turnOn}, \mathit{turnOff}, \mathit{roomIn}, \mathit{roomOut}\}$

Problems

The light must be accessed in mutual exclusion

The Room and The Light (initially ON)

$\text{Room} \triangleq \mathit{roomIn}.\mathit{roomOut}.\text{Room}$ $\qquad$ $\text{Light} \triangleq \mathit{turnOff}.\mathit{turnOn}.\text{Light}$

One Counting-Down Prisoner

$$\begin{aligned} \text{C}_{i+1} &\triangleq \overline{\mathit{roomIn}}.(\overline{\mathit{turnOff}}.\overline{\mathit{roomOut}}.\text{C}_i + \overline{\mathit{turnOn}}.\overline{\mathit{turnOff}}.\overline{\mathit{roomOut}}.\text{C}_{i+1}) \\ \text{C}_0 &\triangleq \overline{\mathit{freeAll}} \end{aligned}$$

49 Generic Prisoners

$$\begin{aligned} \text{P} &\triangleq \overline{\mathit{roomIn}}.(\overline{\mathit{turnOn}}.\overline{\mathit{roomOut}}.\text{IP} + \overline{\mathit{turnOff}}.\overline{\mathit{turnOn}}.\overline{\mathit{roomOut}}.\text{P}) \\ \text{IP} &\triangleq \overline{\mathit{roomIn}}.\tau.\overline{\mathit{roomOut}}.\text{IP} \end{aligned}$$

Prison

$$(\text{Room} \mid \text{Light} \mid \text{C}_{50} \mid \text{P} \mid \ldots \mid \text{P}) \setminus \{\mathit{turnOn}, \mathit{turnOff}, \mathit{roomIn}, \mathit{roomOut}\}$$

Problems

Initial state is not known

The Light

$$\begin{array}{rcl} \text{Light} & \triangleq & \tau.\text{LightOn} + \tau.\text{LightOff} \\ \text{LightOn} & \triangleq & \mathit{turnOff}.\text{LightOff} \\ \text{LightOff} & \triangleq & \mathit{turnOn}.\text{LightOn} \end{array}$$

If the light is initially ON and the counting prisoner enters the room first then it must count 50 switching, otherwise only 49... but he cannot know!

Prison

$$(\text{Room} \mid \text{Light} \mid C_{50or49?} \mid \text{P} \mid \ldots \mid \text{P}) \setminus \{\mathit{turnOn}, \mathit{turnOff}, \mathit{roomIn}, \mathit{roomOut}\}$$

Problems

Initial state is not known

The Light

$$\begin{aligned}\text{Light} &\triangleq \tau.\text{LightOn} + \tau.\text{LightOff}\\ \text{LightOn} &\triangleq \mathit{turnOff}.\text{LightOff}\\ \text{LightOff} &\triangleq \mathit{turnOn}.\text{LightOn}\end{aligned}$$

If the light is initially ON and the counting prisoner enters the room first then it must count 50 switching, otherwise only 49... but he cannot know!

Prison

$$(\text{Room} \mid \text{Light} \mid C_{50or49?} \mid \text{P} \mid \ldots \mid \text{P}) \setminus \{\mathit{turnOn}, \mathit{turnOff}, \mathit{roomIn}, \mathit{roomOut}\}$$

Problems

Initial state is not known

The Light

$$\begin{aligned} \text{Light} &\triangleq \tau.\text{LightOn} + \tau.\text{LightOff} \\ \text{LightOn} &\triangleq \mathit{turnOff}.\text{LightOff} \\ \text{LightOff} &\triangleq \mathit{turnOn}.\text{LightOn} \end{aligned}$$

If the light is initially ON and the counting prisoner enters the room first then it must count 50 switching, otherwise only 49... but he cannot know!

Prison

$$(\text{Room} \mid \text{Light} \mid \text{C}_{50or49?} \mid \text{P} \mid \ldots \mid \text{P}) \setminus \{\mathit{turnOn}, \mathit{turnOff}, \mathit{roomIn}, \mathit{roomOut}\}$$

Solution

What about counting twice?

$50 + 49 = 99$

$49 + 49 = 98$

If he can count 98 switchings then all other prisoners must have entered the room at least once (well... most of them twice). If not:

$49 + 48 = 97$

$48 + 48 = 96$

Generic Prisoners & Re-playing Prisoners & Idle Prisoners

$$\begin{array}{rcl} \mathrm{P} & \triangleq & \overline{roomIn}.(\overline{turnOn}.\overline{roomOut}.\mathrm{RP} + \overline{turnOff}.\overline{turnOn}.\overline{roomOut}.\mathrm{P}) \\ \mathrm{RP} & \triangleq & \overline{roomIn}.(\overline{turnOn}.\overline{roomOut}.\mathrm{IP} + \overline{turnOff}.\overline{turnOn}.\overline{roomOut}.\mathrm{RP}) \\ \mathrm{IP} & \triangleq & \overline{roomIn}.\tau.\overline{roomOut}.\mathrm{IP} \end{array}$$

Prison

$$(\mathrm{Room} \mid \mathrm{Light} \mid \mathrm{C}_{98} \mid \mathrm{P} \mid \ldots \mid \mathrm{P}) \setminus \{turnOn, turnOff, roomIn, roomOut\}$$

Solution

What about counting twice?

$50 + 49 = 99$

$49 + 49 = 98$

If he can count 98 switchings then all other prisoners must have entered the room at least once (well... most of them twice). If not:

$49 + 48 = 97$

$48 + 48 = 96$

Generic Prisoners & Re-playing Prisoners & Idle Prisoners

$$\begin{array}{rcl}
\mathrm{P} & \triangleq & \overline{roomIn}.(\overline{turnOn}.\overline{roomOut}.\mathrm{RP} + \overline{turnOff}.\overline{turnOn}.\overline{roomOut}.\mathrm{P}) \\
\mathrm{RP} & \triangleq & \overline{roomIn}.(\overline{turnOn}.\overline{roomOut}.\mathrm{IP} + \overline{turnOff}.\overline{turnOn}.\overline{roomOut}.\mathrm{RP}) \\
\mathrm{IP} & \triangleq & \overline{roomIn}.\tau.\overline{roomOut}.\mathrm{IP}
\end{array}$$

Prison

$(\mathrm{Room} \mid \mathrm{Light} \mid \mathrm{C}_{98} \mid \mathrm{P} \mid \ \ldots \ \mid \mathrm{P}) \backslash \{turnOn, turnOff, roomIn, roomOut\}$

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange data (e.g., integers).

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil$$

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange data (e.g., integers).

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil$$

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange data (e.g., integers).

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil$$

Parametrized Process Constants

For example: $Bank(total) \triangleq save(x).Bank(total + x)$.

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange data (e.g., integers).

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil \mid Bank(100)$$

Parametrized Process Constants

For example: $Bank(total) \triangleq save(x).Bank(total + x)$.

Value Passing CCS

Main Idea

Handshake synchronization is extended with the possibility to exchange data (e.g., integers).

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid Nil \mid Bank(103)$$

Parametrized Process Constants

For example: $Bank(total) \triangleq save(x).Bank(total + x)$.

Translation of Value Passing CCS to Standard CCS

$$\longrightarrow$$

Value Passing CCS

$$C \triangleq in(x).C'(x)$$

$$C'(x) \triangleq \overline{out(x)}.C$$

Standard CCS

$$C \triangleq \sum_{i \in \mathbb{N}} in_i.C'_i$$

$$C'_i \triangleq \overline{out_i}.C$$

$$\begin{tikzcd}
C \arrow[d, bend left, "in(x)"] \\
C'(x) \arrow[u, bend left, "\overline{out(x)}"]
\end{tikzcd}$$

symbolic LTS

$$\begin{tikzcd}
 & \cdots & \\
C'_i \arrow[r, bend left, "\overline{out_i}"] & C \arrow[l, bend left, "in_i"] \arrow[r, bend left, "in_2"] \arrow[d, bend left, "in_1"] & C'_2 \arrow[l, bend left, "\overline{out_2}"] \\
 & C'_1 \arrow[u, bend left, "\overline{out_1}"] &
\end{tikzcd}$$

infinite LTS

CCS Has Full Turing Power

Fact

CCS can simulate a computation of any Turing machine.

Remark

Hence CCS is as expressive as any other programming language but its use is to rather describe the behaviour of reactive systems than to perform specific calculations.

Strong Bisimilarity

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS.

Strong Bisimulation

A binary relation $R \subseteq Proc \times Proc$ is a strong bisimulation iff whenever $(s, t) \in R$ then for each $a \in Act$:

- if $s \xrightarrow{a} s'$ then $t \xrightarrow{a} t'$ for some t' such that $(s', t') \in R$
- if $t \xrightarrow{a} t'$ then $s \xrightarrow{a} s'$ for some s' such that $(s', t') \in R$.

Strong Bisimilarity

Two processes $p_1, p_2 \in Proc$ are strongly bisimilar ($p_1 \sim p_2$) if and only if there exists a strong bisimulation R such that $(p_1, p_2) \in R$.

$$\sim \;=\; \bigcup\{R \mid R \text{ is a strong bisimulation}\}$$

Example – Buffer

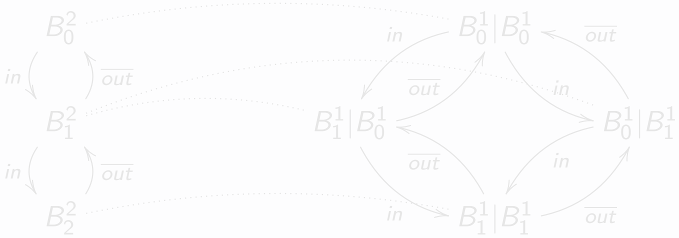
Buffer of Capacity 1

$B_0^1 \triangleq in.B_1^1$
$B_1^1 \triangleq \overline{out}.B_0^1$

Buffer of Capacity n

$B_0^n \triangleq in.B_1^n \quad B_i^n \triangleq in.B_{i+1}^n + \overline{out}.B_{i-1}^n \quad$ for $0 < i < n$
$B_n^n \triangleq \overline{out}.B_{n-1}^n$

Example: $B_0^2 \sim B_0^1|B_0^1$


$$\begin{tikzcd}
B_0^2 \arrow[d, bend right, "in"'] \arrow[rrr, dash, dotted, bend left] & & & B_0^1|B_0^1 \arrow[dl, bend right, "in"'] \arrow[dr, bend left, "in"] & \\
B_1^2 \arrow[u, bend right, "\overline{out}"'] \arrow[d, bend right, "in"'] \arrow[rr, dash, dotted, bend left] \arrow[rrrr, dash, dotted, bend left] & & B_1^1|B_0^1 \arrow[ur, bend right, "\overline{out}"'] \arrow[dr, bend right, "in"'] & & B_0^1|B_1^1 \arrow[ul, bend left, "\overline{out}"] \arrow[dl, bend left, "in"] \\
B_2^2 \arrow[u, bend right, "\overline{out}"'] \arrow[rrr, dash, dotted, bend left] & & & B_1^1|B_1^1 \arrow[ul, bend right, "\overline{out}"'] \arrow[ur, bend left, "\overline{out}"] &
\end{tikzcd}$$

Example – Buffer

Buffer of Capacity 1

$B^1_0 \triangleq in.B^1_1$
$B^1_1 \triangleq \overline{out}.B^1_0$

Buffer of Capacity n

$B^n_0 \triangleq in.B^n_1$ $B^n_i \triangleq in.B^n_{i+1} + \overline{out}.B^n_{i-1}$ for $0 < i < n$
$B^n_n \triangleq \overline{out}.B^n_{n-1}$

Example: $B^2_0 \sim B^1_0|B^1_0$

$$\begin{tikzcd}
B^2_0 \arrow[d, bend right, "in"'] \arrow[rrr, dotted, dash] & & & B^1_0|B^1_0 \arrow[dl, bend right, "in"'] \arrow[dr, bend left, "in"] & \\
B^2_1 \arrow[u, bend right, "\overline{out}"'] \arrow[d, bend right, "in"'] \arrow[rr, dotted, dash] \arrow[rrrr, dotted, dash, bend left] & & B^1_1|B^1_0 \arrow[ur, bend right, "\overline{out}"'] \arrow[dr, bend right, "in"'] & & B^1_0|B^1_1 \arrow[ul, bend left, "\overline{out}"] \arrow[dl, bend left, "in"] \\
B^2_2 \arrow[u, bend right, "\overline{out}"'] \arrow[rrr, dotted, dash] & & & B^1_1|B^1_1 \arrow[ul, bend right, "\overline{out}"'] \arrow[ur, bend left, "\overline{out}"] &
\end{tikzcd}$$

Example – Buffer

Buffer of Capacity 1

$B^1_0 \triangleq in.B^1_1$
$B^1_1 \triangleq \overline{out}.B^1_0$

Buffer of Capacity n

$B^n_0 \triangleq in.B^n_1$ $B^n_i \triangleq in.B^n_{i+1} + \overline{out}.B^n_{i-1}$ for $0 < i < n$
$B^n_n \triangleq \overline{out}.B^n_{n-1}$

Example: $B^2_0 \sim B^1_0 | B^1_0$

$$\begin{tikzcd}
B^2_0 \arrow[d, bend right, "in"'] \arrow[rrr, dash, dotted, bend left] & & & B^1_0|B^1_0 \arrow[dl, bend right, "in"'] \arrow[dr, bend left, "in"] & \\
B^2_1 \arrow[u, bend right, "\overline{out}"'] \arrow[d, bend right, "in"'] \arrow[rr, dash, dotted, bend left] \arrow[rrrr, dash, dotted, bend left] & & B^1_1|B^1_0 \arrow[ur, bend right, "\overline{out}"'] \arrow[dr, bend right, "in"'] & & B^1_0|B^1_1 \arrow[ul, bend right, "\overline{out}"'] \arrow[dl, bend right, "in"'] \\
B^2_2 \arrow[u, bend right, "\overline{out}"'] \arrow[rrr, dash, dotted, bend left] & & & B^1_1|B^1_1 \arrow[ul, bend right, "\overline{out}"'] \arrow[ur, bend right, "\overline{out}"'] &
\end{tikzcd}$$

Example – Buffer

Theorem

For all natural numbers n: $B^n_0 \sim \underbrace{B^1_0 | B^1_0 | \cdots | B^1_0}_{n\ times}$

Proof.

Construct the following binary relation where $i_1, i_2, \ldots, i_n \in \{0, 1\}$.

$$R = \{ \big(B^n_i,\ B^1_{i_1} | B^1_{i_2} | \cdots | B^1_{i_n} \big) \mid \sum_{j=1}^{n} i_j = i \}$$

- $\big(B^n_0,\ B^1_0 | B^1_0 | \cdots | B^1_0 \big) \in R$
- R is strong bisimulation

□

Example – Buffer

Theorem

For all natural numbers n: $B_0^n \sim \underbrace{B_0^1 | B_0^1 | \cdots | B_0^1}_{n \ times}$

Proof.

Construct the following binary relation where $i_1, i_2, \ldots, i_n \in \{0, 1\}$.

$$R = \{ \left(B_i^n,\ B_{i_1}^1 | B_{i_2}^1 | \cdots | B_{i_n}^1 \right) \mid \sum_{j=1}^{n} i_j = i \}$$

- $\left(B_0^n,\ B_0^1 | B_0^1 | \cdots | B_0^1 \right) \in R$
- R is strong bisimulation

□

Question Time

Prove the following (or give counterexamples)

- $\sim$ is reflexive
- $\sim$ is symmetric
- $\sim$ is transitive
- $\sim$ is an equivalence relation
- If $\mathcal{R}$ is a strong bisimulation then $\mathcal{R} \subseteq \sim$
- For all natural numbers n, k: $\quad B_0^{n+k} \sim B_0^n | B_0^k$

Playful digression

Some advanced proof methods

1. Proof by obviousness: So evident it need not to be mentioned
2. Proof by general agreement: All in favor?
3. Proof by majority: When general agreement fails
4. Proof by plausibility: It sounds good
5. Proof by intuition: I have this feeling...
6. Proof by lost reference: I saw it somewhere
7. Proof by obscure reference: It appeared in the Annals of Polish Math. Soc. (1854, in polish)
8. Proof by logic: It is on the textbook, hence it must be true
9. Proof by intimidation: Who is saying that it is false!?
10. Proof by authority: Don Knuth said it was true
11. Proof by deception: Everybody please turn their backs...
12. Proof by divine word: Lord said *let it be true*

Playful digression

Some advanced proof methods

1. Proof by obviousness: So evident it need not to be mentioned
2. Proof by general agreement: All in favor?
3. Proof by majority: When general agreement fails
4. Proof by plausibility: It sounds good
5. Proof by intuition: I have this feeling...
6. Proof by lost reference: I saw it somewhere
7. Proof by obscure reference: It appeared in the Annals of Polish Math. Soc. (1854, in polish)
8. Proof by logic: It is on the textbook, hence it must be true
9. Proof by intimidation: Who is saying that it is false!?
10. Proof by authority: Don Knuth said it was true
11. Proof by deception: Everybody please turn their backs...
12. Proof by divine word: Lord said *let it be true*

Playful digression

Some advanced proof methods

1. Proof by obviousness: So evident it need not to be mentioned
2. Proof by general agreement: All in favor?
3. Proof by majority: When general agreement fails
4. Proof by plausibility: It sounds good
5. Proof by intuition: I have this feeling...
6. Proof by lost reference: I saw it somewhere
7. Proof by obscure reference: It appeared in the Annals of Polish Math. Soc. (1854, in polish)
8. Proof by logic: It is on the textbook, hence it must be true
9. Proof by intimidation: Who is saying that it is false!?
10. Proof by authority: Don Knuth said it was true
11. Proof by deception: Everybody please turn their backs...
12. Proof by divine word: Lord said *let it be true*

Playful digression

Some advanced proof methods

1. Proof by obviousness: So evident it need not to be mentioned
2. Proof by general agreement: All in favor?
3. Proof by majority: When general agreement fails
4. Proof by plausibility: It sounds good
5. Proof by intuition: I have this feeling...
6. Proof by lost reference: I saw it somewhere
7. Proof by obscure reference: It appeared in the Annals of Polish Math. Soc. (1854, in polish)
8. Proof by logic: It is on the textbook, hence it must be true
9. Proof by intimidation: Who is saying that it is false!?
10. Proof by authority: Don Knuth said it was true
11. Proof by deception: Everybody please turn their backs...
12. Proof by divine word: Lord said *let it be true*

Strong Bisimilarity is a Congruence for CCS Operations

Theorem

Let P and Q be CCS processes such that $P \sim Q$. Then

- $\alpha.P \sim \alpha.Q$ *for each action* $\alpha \in Act$
- $P + R \sim Q + R$ *for each CCS process* R
- $R + P \sim R + Q$ *for each CCS process* R
- $P \mid R \sim Q \mid R$ *for each CCS process* R
- $R \mid P \sim R \mid Q$ *for each CCS process* R
- $P[f] \sim Q[f]$ *for each relabelling function* f
- $P \setminus L \sim Q \setminus L$ *for each set of labels* L.

(proof sketch for $\_ + R$) Let $\mathcal{R} = \{\, (P + R, Q + R) \mid R \in Proc \,\} \cup \sim \cup Id$. It suffices to show that $\mathcal{R}$ is a bisimulation. Suppose $P + R \xrightarrow{\mu} S$. If it is because $P \xrightarrow{\mu} S$, then $Q \xrightarrow{\mu} T$ with $(S, T) \in \sim$ because $P \sim Q$ and hence $Q + R \xrightarrow{\mu} T$ with $(S, T) \in \sim \subseteq \mathcal{R}$. Otherwise, it is because $R \xrightarrow{\mu} S$, but then $Q + R \xrightarrow{\mu} S$ with $(S, S) \in Id \subset \mathcal{R}$. The case where $Q + R$ moves is symmetric.

Other Properties of Strong Bisimilarity

The Following Properties Hold for all CCS Processes P, Q, R

- $P + Nil \sim P$ (Neutral element for $+$)
- $P \,|\, Nil \sim P$ (Neutral element for $|$)
- $P + Q \sim Q + P$ (Commutativity of $+$)
- $P \,|\, Q \sim Q \,|\, P$ (Commutativity of $|$)
- $(P + Q) + R \sim P + (Q + R)$ (associativity of $+$)
- $(P \,|\, Q) \,|\, R \sim P \,|\, (Q \,|\, R)$ (associativity of $|$)
- $P + P \sim P$ (Idempotency of $+$)

(proof sketch for commutativity of $+$)
Let $\mathcal{R} = \{\, (P + Q, Q + P) \mid P, Q \in Proc \,\} \cup Id$.
It suffices to show that $\mathcal{R}$ is a bisimulation.
Suppose $P + Q \xrightarrow{\mu} S$ using SUM1 (resp. SUM2), then $Q + P \xrightarrow{\mu} S$ using SUM2 (resp. SUM1) and $(S, S) \in Id \subseteq \mathcal{R}$.

The case where $Q + P$ moves is analogous.

Strong Bisimilarity – Summary

Properties of $\sim$

- an equivalence relation
- the largest strong bisimulation
- a congruence
- enough to prove some natural rules like
  - $P|Q \sim Q|P$
  - $P|Nil \sim P$
  - $(P|Q)|R \sim Q|(P|R)$
  - $\cdots$

Question

Should we look any further???

Strong Bisimilarity – Summary

Properties of $\sim$

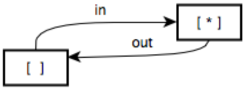
- an equivalence relation
- the largest strong bisimulation
- a congruence
- enough to prove some natural rules like
  - $P|Q \sim Q|P$
  - $P|Nil \sim P$
  - $(P|Q)|R \sim Q|(P|R)$
  - $\cdots$

Question

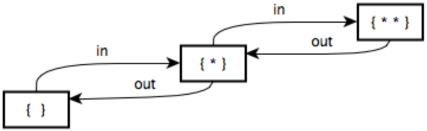
Should we look any further???

Behavioural Equivalence: Cells

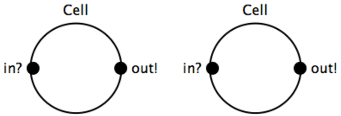
$Cell1$ = One-position cell



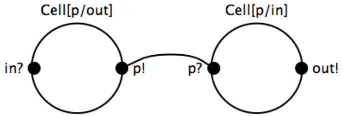
$Cell2$ = Two-positions cell



$ParCell = Cell1 \mid Cell1$



$SeqCell = Cell1 \frown Cell1$

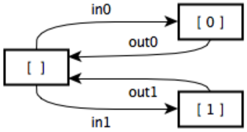


(p restricted)

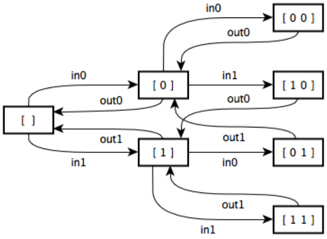
Are *Cell2*, *ParCell* and *SeqCell* equivalent?

Behavioural Equivalence: FIFO buffers

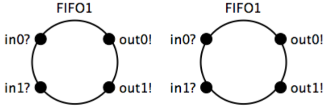
FIFO1 = One-pos. FIFO buffer



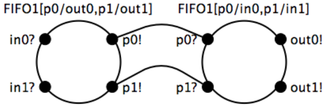
FIFO2 = Two-pos. FIFO buffer



$ParFIFO = FIFO1 \mid FIFO1$



$SeqFIFO = FIFO1 \frown FIFO1$



($p0, p1$ restricted)

Are *FIFO2*, *ParFIFO* and *SeqFIFO* equivalent?

Weak Transition Relation

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS such that $\tau \in Act$.

Definition of the Weak Transition Relations

Let a be an action or ε:

$$\overset{a}{\Longrightarrow} = \begin{cases} (\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^* & \text{if } a \neq \varepsilon \\ (\xrightarrow{\tau})^* & \text{if } a = \varepsilon \end{cases}$$

Definition

If a is an observable action, then $\hat{a} = a$. On the other hand, $\hat{\tau} = \varepsilon$.

Weak Bisimilarity

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS such that $\tau \in Act$.

Weak Bisimulation

A binary relation $R \subseteq Proc \times Proc$ is a weak bisimulation iff whenever $(s,t) \in R$ then for each $a \in Act$ (including τ):

- if $s \xrightarrow{a} s'$ then $t \overset{\hat{a}}{\Longrightarrow} t'$ for some t' such that $(s',t') \in R$
- if $t \xrightarrow{a} t'$ then $s \overset{\hat{a}}{\Longrightarrow} s'$ for some s' such that $(s',t') \in R$.

Weak Bisimilarity

Two processes $p_1, p_2 \in Proc$ are weakly bisimilar ($p_1 \approx p_2$) if and only if there exists a weak bisimulation R such that $(p_1, p_2) \in R$.

$$\approx \; = \bigcup \{R \mid R \text{ is a weak bisimulation}\}$$

Weak Bisimilarity – Properties

Properties of $\approx$

- an equivalence relation
- the largest weak bisimulation
- strong bisimilarity is included in weak bisimilarity ($\sim \subseteq \approx$)
- validates lots of natural laws, e.g.
  - $a.\tau.P \approx a.P$
  - $P + \tau.P \approx \tau.P$
  - $a.(P + \tau.Q) \approx a.(P + \tau.Q) + a.Q$
  - $P + Q \approx Q + P \qquad P|Q \approx Q|P \qquad P + Nil \approx P \qquad \ldots$
- abstracts from τ loops

$$\begin{tikzcd}
\bullet \arrow[loop left, "\tau"] \arrow[dr, "a"] & & \approx & \bullet \arrow[dr, "a"] & \\
& \bullet & & & \bullet
\end{tikzcd}$$

Is Weak Bisimilarity a Congruence for CCS?

Theorem

Let P and Q be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ *for each action* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *for each CCS process* R
- $R \mid P \approx R \mid Q$ *for each CCS process* R
- $P[f] \approx Q[f]$ *for each relabelling function* f
- $P \setminus L \approx Q \setminus L$ *for each set of labels* L.

Conclusion

Weak bisimilarity is not a congruence for CCS.

Is Weak Bisimilarity a Congruence for CCS?

Theorem

Let P and Q be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ *for each action* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *for each CCS process* R
- $R \mid P \approx R \mid Q$ *for each CCS process* R
- $P[f] \approx Q[f]$ *for each relabelling function* f
- $P \setminus L \approx Q \setminus L$ *for each set of labels* L.

What about choice?

Conclusion

Weak bisimilarity is not a congruence for CCS.

Is Weak Bisimilarity a Congruence for CCS?

Theorem

Let P and Q be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ *for each action* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *for each CCS process* R
- $R \mid P \approx R \mid Q$ *for each CCS process* R
- $P[f] \approx Q[f]$ *for each relabelling function* f
- $P \setminus L \approx Q \setminus L$ *for each set of labels* L.

(proof attempt for $\_ + R$) Let $\mathcal{R} = \{ (P + R, Q + R) \mid R \in Proc \} \cup \approx \cup Id$. It suffices to show that $\mathcal{R}$ is a weak bisimulation. Suppose $P + R \xrightarrow{\mu} S$. If it is because $R \xrightarrow{\mu} S$, then $Q + R \xrightarrow{\mu} S$ with $(S, S) \in Id \subseteq \mathcal{R}$. Otherwise, it is because $P \xrightarrow{\mu} S$, then $Q \overset{\hat{\mu}}{\Longrightarrow} T$ with $(S, T) \in \approx$ because $P \approx Q$ (is this true???) and hence $Q + R \overset{\hat{\mu}}{\Longrightarrow} T$ with $(S, T) \in \approx \subseteq \mathcal{R}$.

Conclusion

Weak bisimilarity is not a congruence for CCS.

Is Weak Bisimilarity a Congruence for CCS?

Theorem

Let P and Q be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ *for each action* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *for each CCS process* R
- $R \mid P \approx R \mid Q$ *for each CCS process* R
- $P[f] \approx Q[f]$ *for each relabelling function* f
- $P \setminus L \approx Q \setminus L$ *for each set of labels* L.

(proof attempt for $\_ + R$) Let $\mathcal{R} = \{ (P+R, Q+R) \mid R \in Proc \} \cup \approx \cup Id$. It suffices to show that $\mathcal{R}$ is a weak bisimulation. Suppose $P + R \xrightarrow{\mu} S$. If it is because $R \xrightarrow{\mu} S$, then $Q + R \xrightarrow{\mu} S$ with $(S, S) \in Id \subseteq \mathcal{R}$. Otherwise, it is because $P \xrightarrow{\mu} S$, then $Q \overset{\hat{\mu}}{\Longrightarrow} T$ with $(S, T) \in \approx$ because $P \approx Q$ (is this true???) and hence $Q + R \overset{\hat{\mu}}{\Longrightarrow} T$ with $(S, T) \in \approx \subseteq \mathcal{R}$.
(counterexample) $\tau.a.Nil \approx a.Nil$ but $\tau.a.Nil + b.Nil \not\approx a.Nil + b.Nil$

Conclusion

Is Weak Bisimilarity a Congruence for CCS?

Theorem

Let P and Q be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ *for each action* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *for each CCS process* R
- $R \mid P \approx R \mid Q$ *for each CCS process* R
- $P[f] \approx Q[f]$ *for each relabelling function* f
- $P \setminus L \approx Q \setminus L$ *for each set of labels* L.

(proof attempt for $\_ + R$) Let $\mathcal{R} = \{ (P+R, Q+R) \mid R \in Proc \} \cup \approx \cup Id$. It suffices to show that $\mathcal{R}$ is a weak bisimulation. Suppose $P + R \xrightarrow{\mu} S$. If it is because $R \xrightarrow{\mu} S$, then $Q + R \xrightarrow{\mu} S$ with $(S,S) \in Id \subseteq \mathcal{R}$. Otherwise, it is because $P \xrightarrow{\mu} S$, then $Q \overset{\hat{\mu}}{\Longrightarrow} T$ with $(S,T) \in \approx$ because $P \approx Q$ (is this true???) and hence $Q + R \overset{\hat{\mu}}{\Longrightarrow} T$ with $(S,T) \in \approx \subseteq \mathcal{R}$.

(counterexample) $\tau.a.Nil \approx a.Nil$ but $\tau.a.Nil + b.Nil \not\approx a.Nil + b.Nil$

What is wrong and which one is it the right alternative?

Is Weak Bisimilarity a Congruence for CCS?

Theorem

Let P and Q be CCS processes such that $P \approx Q$. Then

- $\alpha.P \approx \alpha.Q$ *for each action* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *for each CCS process* R
- $R \mid P \approx R \mid Q$ *for each CCS process* R
- $P[f] \approx Q[f]$ *for each relabelling function* f
- $P \setminus L \approx Q \setminus L$ *for each set of labels* L.

Conclusion

Weak bisimilarity is not a congruence for CCS.

Case Study: Communication Protocol

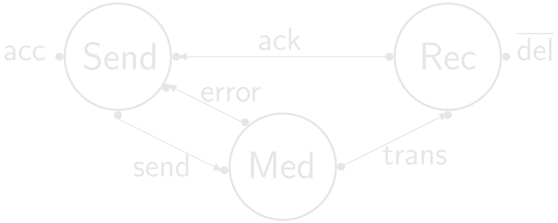
A message delivery system, accept a message transfer request, deliver the message and then it becomes ready again to serve a new request.

The implementation must take into account that errors can arise during the message transfer, in which case the message must be resent.

Case Study: Communication Protocol

$$\mathsf{Spec} \triangleq \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

$$\mathsf{Impl} \triangleq (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \setminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

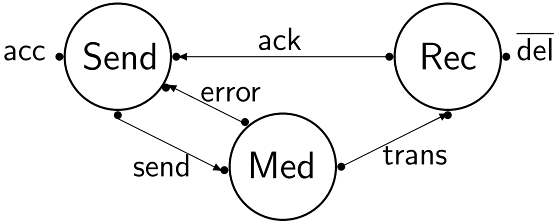


$$\mathsf{Impl} \overset{?}{\approx} \mathsf{Spec}$$

Case Study: Communication Protocol

$$\mathsf{Spec} \triangleq \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

$$\mathsf{Impl} \triangleq (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \setminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

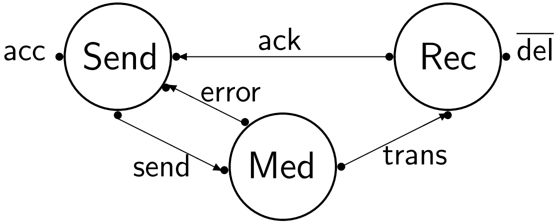


$$\mathsf{Impl} \overset{?}{\approx} \mathsf{Spec}$$

Case Study: Communication Protocol

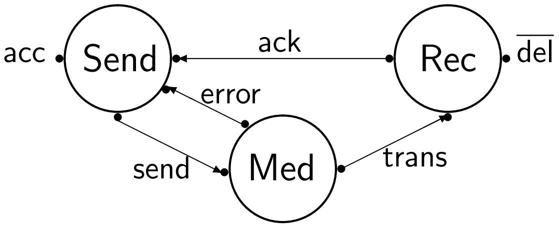
$$\mathrm{Spec} \triangleq \mathrm{acc}.\overline{\mathrm{del}}.\mathrm{Spec}$$

$$\mathrm{Impl} \triangleq (\mathrm{Send} \mid \mathrm{Med} \mid \mathrm{Rec}) \setminus \{\mathrm{send}, \mathrm{trans}, \mathrm{ack}, \mathrm{error}\}$$



$$\mathrm{Impl} \overset{?}{\approx} \mathrm{Spec}$$

Case Study: Communication Protocol

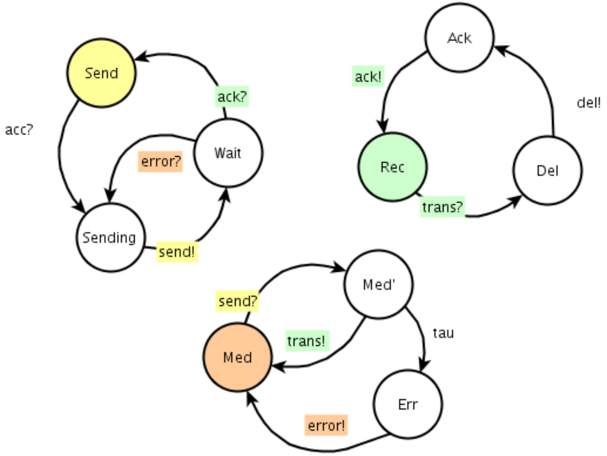


Send	$\triangleq$	acc.Sending
Sending	$\triangleq$	$\overline{\text{send}}$.Wait
Wait	$\triangleq$	ack.Send + error.Sending

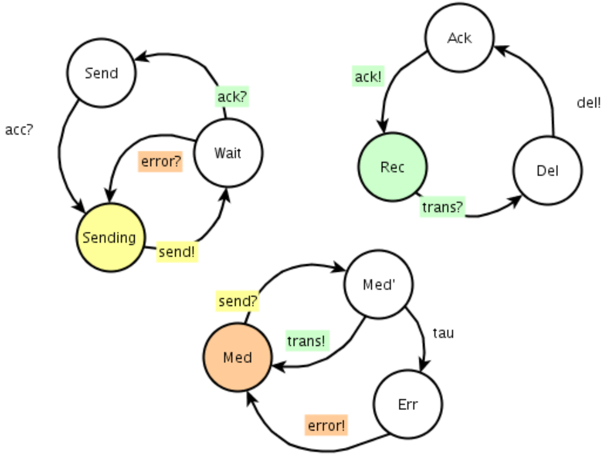
Rec	$\triangleq$	trans.Del
Del	$\triangleq$	$\overline{\text{del}}$.Ack
Ack	$\triangleq$	$\overline{\text{ack}}$.Rec

$$\begin{aligned} \text{Med} &\triangleq \text{send}.\text{Med}' \\ \text{Med}' &\triangleq \tau.\text{Err} + \overline{\text{trans}}.\text{Med} \\ \text{Err} &\triangleq \overline{\text{error}}.\text{Med} \end{aligned}$$

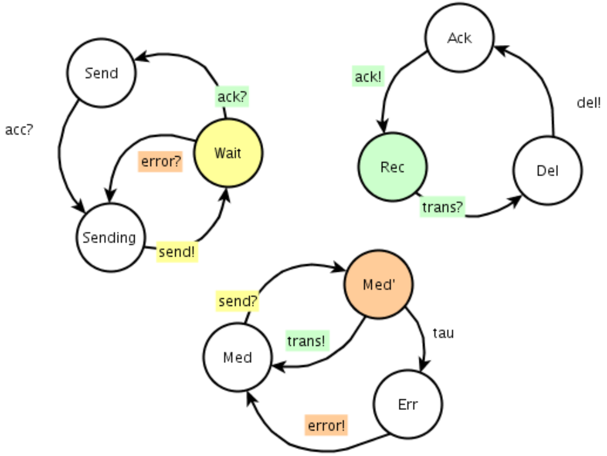
A visual execution



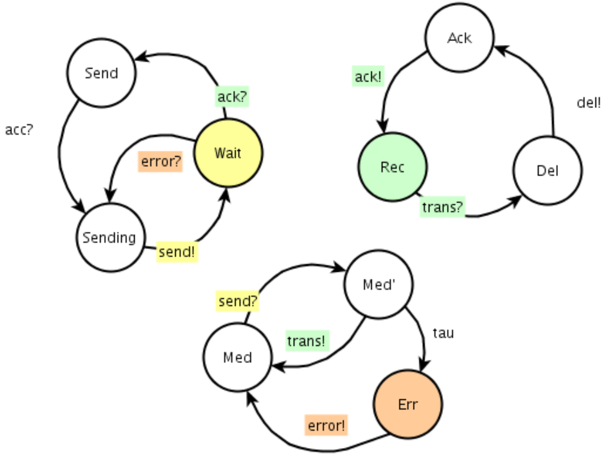
A visual execution



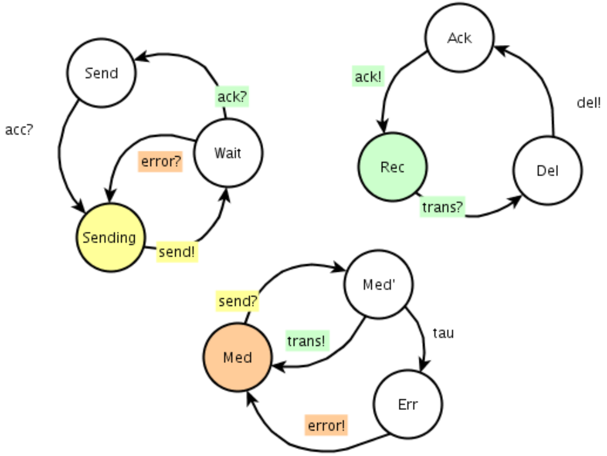
A visual execution



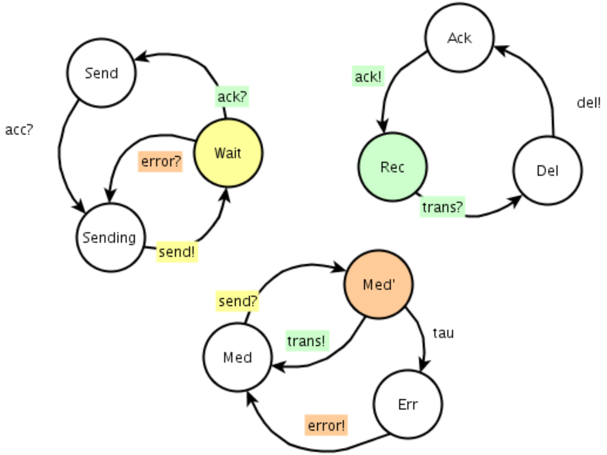
A visual execution



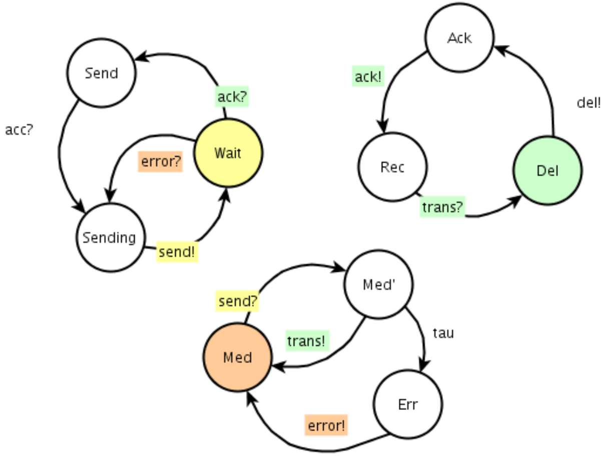
A visual execution



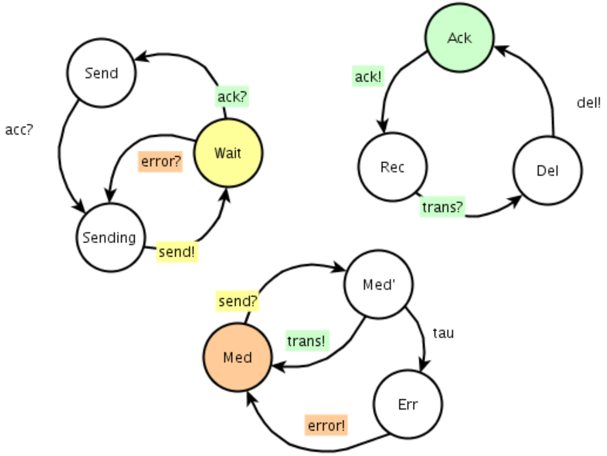
A visual execution



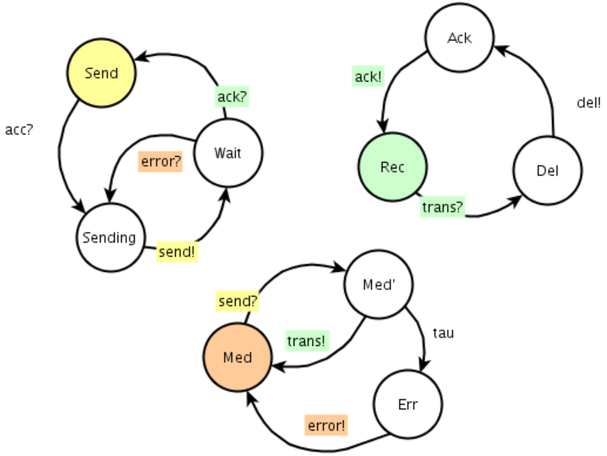
A visual execution



A visual execution



A visual execution



Question Time

$$\mathsf{Spec} \triangleq \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

$$\mathsf{Impl} \triangleq (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \setminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

Question

$$\mathsf{Impl} \overset{?}{\approx} \mathsf{Spec}$$

1. Draw the LTS of Spec (by hand)
2. Draw the LTS of Impl (by hand)
3. Prove (by hand) their equivalence
4. Use TAPAS!