A Calculus of Communicating Systems

Rocco De Nicola

Dipartimento di Sistemi ed Informatica
Università di Firenze

Process Algebras and Concurrent Systems

CCS Basics (Sequential Fragment)

- *Nil* (or 0) process (the only atomic process)
- action prefixing ($a.P$)
- names and recursive definitions ($\stackrel{\mathrm{def}}{=}$)
- 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 (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)

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

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

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

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

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. $A \stackrel{\text{def}}{=} \overline{a}.A \mid A$.

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:

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

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

$$\text{ACT} \quad \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad\qquad \text{SUM}_j \quad \frac{P_j \xrightarrow{\alpha} P'_j}{\sum_{i \in I} P_i \xrightarrow{\alpha} P'_j} \; j \in I$$

$$\text{COM1} \quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \qquad\qquad \text{COM2} \quad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\text{COM3} \quad \frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$\text{RES} \quad \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \; \alpha, \overline{\alpha} \notin L \qquad\qquad \text{REL} \quad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$\text{CON} \quad \frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'} \; K \stackrel{\text{def}}{=} P$$

Deriving Transitions in CCS

Let $A \stackrel{\mathrm{def}}{=} a.A$. 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?

Deriving Transitions in CCS

Let $A \stackrel{\mathrm{def}}{=} a.A$. Then

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

Why?

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

Let $A \stackrel{\mathrm{def}}{=} a.A$. 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 $A \stackrel{\mathrm{def}}{=} a.A$. Then

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

Why?

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

Deriving Transitions in CCS

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

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

Why?

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

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

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

Why?

$$\text{REL}\ \dfrac{\text{COM1}\ \dfrac{\text{COM1}\ \dfrac{\text{CON}\ \dfrac{\text{ACT}\ \dfrac{}{a.A \xrightarrow{a} A}\quad A \stackrel{\text{def}}{=} a.A}{A \xrightarrow{a} A}}{A \,|\, \overline{a}.Nil \xrightarrow{a} A \,|\, \overline{a}.Nil}}{(A \,|\, \overline{a}.Nil) \,|\, b.Nil \xrightarrow{a} (A \,|\, \overline{a}.Nil) \,|\, b.Nil}}{((A \,|\, \overline{a}.Nil) \,|\, b.Nil)[c/a] \xrightarrow{c} ((A \,|\, \overline{a}.Nil) \,|\, b.Nil)[c/a]}$$

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

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

Value Passing CCS

Main Idea

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

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

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: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

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: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

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: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

Translation of Value Passing CCS to Standard CCS

$$\longrightarrow$$

Value Passing CCS

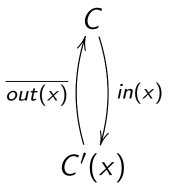
$$C \stackrel{\text{def}}{=} in(x).C'(x)$$

$$C'(x) \stackrel{\text{def}}{=} \overline{out(x)}.C$$

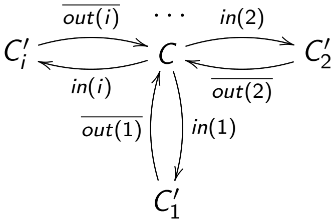
Standard CCS

$$C \stackrel{\text{def}}{=} \sum_{i \in \mathbb{N}} in(i).C'_i$$

$$C'_i \stackrel{\text{def}}{=} \overline{out(i)}.C$$



symbolic LTS



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 $(p_1, p_2) \in R$.

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

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* $\alpha \in Act$
- $P + R \sim Q + R$ *and* $R + P \sim R + Q$ *for each CCS process* R
- $P \mid R \sim Q \mid R$ *and* $R \mid P \sim R \mid Q$ *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.

Other Properties of Strong Bisimilarity

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

- $P + Q \sim Q + P$
- $P \mid Q \sim Q \mid P$
- $P + Nil \sim P$
- $P \mid Nil \sim P$
- $(P + Q) + R \sim P + (Q + R)$
- $(P \mid Q) \mid R \sim P \mid (Q \mid R)$

Example – Buffer

Buffer of Capacity 1

$B_0^1 \stackrel{\mathrm{def}}{=} in.B_1^1$

$B_1^1 \stackrel{\mathrm{def}}{=} \overline{out}.B_0^1$

Example – Buffer

Buffer of Capacity 1

$$B_0^1 \stackrel{\text{def}}{=} in.B_1^1$$
$$B_1^1 \stackrel{\text{def}}{=} \overline{out}.B_0^1$$

Buffer of Capacity n

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

Example – Buffer

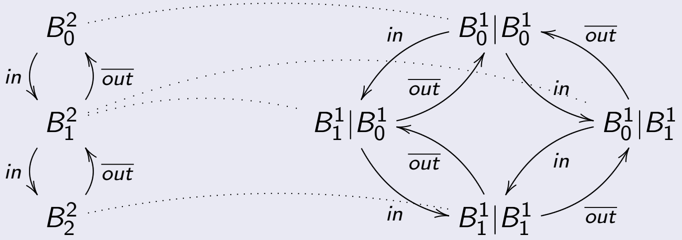
Buffer of Capacity 1

$$B_0^1 \stackrel{\text{def}}{=} in.B_1^1$$
$$B_1^1 \stackrel{\text{def}}{=} \overline{out}.B_0^1$$

Buffer of Capacity n

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

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



Example – Buffer

Theorem

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

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

$\square$

Strong Bisimilarity – Summary

Properties of $\sim$

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

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???

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:

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

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

Definition

Same as for the strong bisimulation game except that

- defender can now answer using $\stackrel{a}{\Longrightarrow}$ moves.

The attacker is still using only $\xrightarrow{a}$ moves.

Let's play!

Weak Bisimulation Game

Definition

Same as for the strong bisimulation game except that

- defender can now answer using $\stackrel{a}{\Longrightarrow}$ moves.

The attacker is still using only $\xrightarrow{a}$ moves.

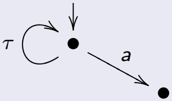
Let's play!

Theorem

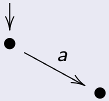
- States s and t are weakly bisimilar if and only if the defender has a universal winning strategy starting from the configuration (s, t).
- States s and t are not weakly bisimilar if and only if the attacker has a universal winning strategy starting from the configuration (s, t).

Properties of $\approx$

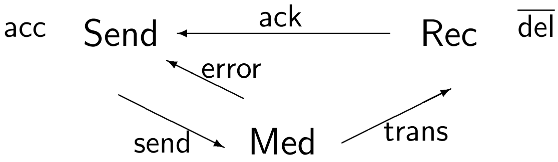
- an equivalence relation
- the largest weak bisimulation
- validates lots of natural laws, e.g.
  - $a.\tau.P \approx a.P$
  - $P + \tau.P \approx \tau.P$
  - $a.(P + \tau.Q) \approx a.(P + \tau.Q) + a.Q$
  - $P + Q \approx Q + P \qquad P|Q \approx Q|P \qquad P + Nil \approx P \qquad \ldots$
- strong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



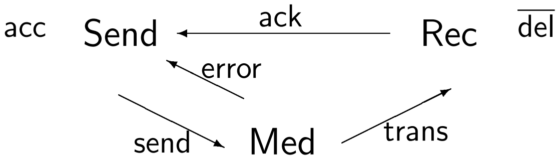
$\approx$



Case Study: Communication Protocol



Case Study: Communication Protocol



$$
\begin{aligned}
\text{Send} &\stackrel{\text{def}}{=} \text{acc}.\text{Sending} \\
\text{Sending} &\stackrel{\text{def}}{=} \overline{\text{send}}.\text{Wait} \\
\text{Wait} &\stackrel{\text{def}}{=} \text{ack}.\text{Send} + \text{error}.\text{Sending}
\end{aligned}
\qquad
\begin{aligned}
\text{Rec} &\stackrel{\text{def}}{=} \text{trans}.\text{Del} \\
\text{Del} &\stackrel{\text{def}}{=} \overline{\text{del}}.\text{Ack} \\
\text{Ack} &\stackrel{\text{def}}{=} \overline{\text{ack}}.\text{Rec}
\end{aligned}
$$

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

$$\mathsf{Impl} \stackrel{\text{def}}{=} (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \setminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

Verification Question

$$\mathsf{Impl} \stackrel{\text{def}}{=} (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \smallsetminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

$$\mathsf{Spec} \stackrel{\text{def}}{=} \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

Verification Question

$$\mathsf{Impl} \stackrel{\mathrm{def}}{=} (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \setminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

$$\mathsf{Spec} \stackrel{\mathrm{def}}{=} \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

Question

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

Verification Question

$$\mathsf{Impl} \stackrel{\mathrm{def}}{=} (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \setminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

$$\mathsf{Spec} \stackrel{\mathrm{def}}{=} \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

Question

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

1. Draw the LTS of Impl and Spec and prove (by hand) the equivalence.

Verification Question

$$\mathsf{Impl} \stackrel{\mathrm{def}}{=} (\mathsf{Send} \mid \mathsf{Med} \mid \mathsf{Rec}) \setminus \{\mathsf{send}, \mathsf{trans}, \mathsf{ack}, \mathsf{error}\}$$

$$\mathsf{Spec} \stackrel{\mathrm{def}}{=} \mathsf{acc}.\overline{\mathsf{del}}.\mathsf{Spec}$$

Question

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

1. Draw the LTS of Impl and Spec and prove (by hand) the equivalence.
2. Use TAPAS.

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* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *and* $R \mid P \approx R \mid Q$ *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.

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* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *and* $R \mid P \approx R \mid Q$ *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?

$\tau.a.Nil \approx a.Nil$ but $\tau.a.Nil + b.Nil \not\approx a.Nil + b.Nil$

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* $\alpha \in Act$
- $P \mid R \approx Q \mid R$ *and* $R \mid P \approx R \mid Q$ *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?

$\tau.a.Nil \approx a.Nil$ but $\tau.a.Nil + b.Nil \not\approx a.Nil + b.Nil$

Conclusion

Weak bisimilarity is not a congruence for CCS.