Concurrency Theory

Georgiana Caltais, University of Twente

The 2026 Dutch Winter School on Logic and Verification, 20-23 January 2026, University of Twente, NL

Concurrency Theory
What?

- **Umbrella Theory**
  - Systems in which multiple components execute independently and interact: reactive systems / cyber-physical systems
    - E.g., (autonomous) cars, power plants, smart-homes, robots, computer scientists :-)
  - Modelling, specification & verification of cy-phy systems
    - Communication, synchronization, equivalence / correctness
  - Pioneers - highly selective and non-exhaustive list:
    - Carl A. Petri (Petri Nets), Toni Hoare (CSP), Gordon Plotkin (SOS), David Park / Robin Milner (bisimulation), Amir Pnueli (temporal logic), Edsger W. Dijkstra (mutual exclusion)

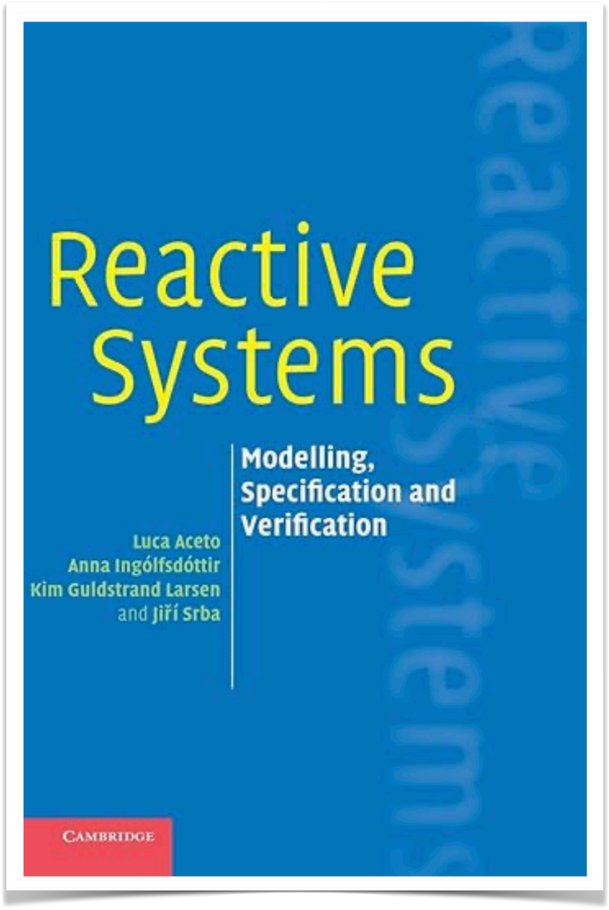
Outline

- Process Algebra CCS
- Analysis / Verification of CCS Models
  - Bisimilarity
  - Hennessy-Milner Logic / Model Checking
  - Axiomatic Reasoning
- Rule Formats

Process Algebra CCS

Calculus of Communicating Systems

- Robin Milner: *A Calculus of Communicating Systems*. LNCS, 1980.
- Credits:



CCS Syntax

Definition The collection $\mathcal{P}$ of *CCS expressions* is given by the following grammar:

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \,|\, Q \quad | \quad P[f] \quad | \quad P \setminus L \ ,$$

where

- K is a process name in $\mathcal{K}$;
- α is an action in **Act**;
- I is a possibly infinite index set;
- $f : \mathbf{Act} \to \mathbf{Act}$ is a *relabelling function* satisfying the following constraints:

$$\begin{aligned} f(\tau) &= \tau \text{ and} \\ f(\bar{a}) &= \overline{f(a)} \text{ for each label } a \ ; \end{aligned}$$

- L is a set of labels from $\mathcal{L}$.

$\alpha \,.\, P$ – sequential composition

Σ – non-deterministic choice / if-else

$|$ – parallel composition / concurrency

$P[f]$ – relabelling

$P \backslash L$ – restriction

$\alpha \in Act$ vs. $\overline{\alpha} \in Act$ – synchr. communication

$$\mathbf{0} = \sum_{i \in \emptyset} P_i \qquad P_1 + P_2 = \sum_{i \in \{1,2\}} P_i \qquad K \stackrel{\text{def}}{=} P$$

CCS Process Terms

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

Examples

$$\mathrm{CM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$$

$$\mathrm{CTM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$$

$$\mathrm{CS} \stackrel{\mathrm{def}}{=} \overline{\mathrm{pub}}.\overline{\mathrm{coin}}.\mathrm{coffee}.\mathrm{CS}$$

$$\mathrm{SmUni} \stackrel{\mathrm{def}}{=} (\mathrm{CM} \mid \mathrm{CS}) \setminus \mathrm{coin} \setminus \mathrm{coffee}$$

$$\begin{aligned} \mathrm{CHM} &\stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{choc}}.\mathrm{CHM} \\ \mathrm{DFM} &\stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{figs}}.\mathrm{DFM} \\ \mathrm{CRM} &\stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{crisps}}.\mathrm{CRM} \end{aligned}$$

$$\mathrm{VM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{item}}.\mathrm{VM}$$

$$\mathrm{CHM} \stackrel{\mathrm{def}}{=} \mathrm{VM}[\mathrm{choc}/\mathrm{item}]$$

CCS (Operational) Semantics

$$\text{ACT} \quad \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

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

$$\text{COM1} \quad \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

$$\text{COM2} \quad \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}$$

$$\text{COM3} \quad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\text{RES} \quad \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad \text{where } \alpha, \bar{\alpha} \notin L$$

$$\text{REL} \quad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

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

G.D. Plotkin, *A Structural Approach to Operational Semantics*, JLAMP 2004 (80's)

Behaviour of CCS Process Terms

Example

$$\text{ACT}\ \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

$$\text{RES}\ \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}\ \text{where } \alpha, \bar{\alpha} \notin L$$

$$\text{SUM}_j\ \frac{P_j \xrightarrow{\alpha} P'_j}{\sum_{i \in I} P_i \xrightarrow{\alpha} P'_j}\ \text{where } j \in I$$

$$\text{REL}\ \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$\text{COM1}\ \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

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

$$\text{COM2}\ \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}$$

$$\text{COM3}\ \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\text{CM} \stackrel{\text{def}}{=} \text{coin}.\overline{\text{coffee}}.\text{CM}$$

$$\text{CS} \stackrel{\text{def}}{=} \overline{\text{pub}}.\overline{\text{coin}}.\text{coffee}.\text{CS}$$

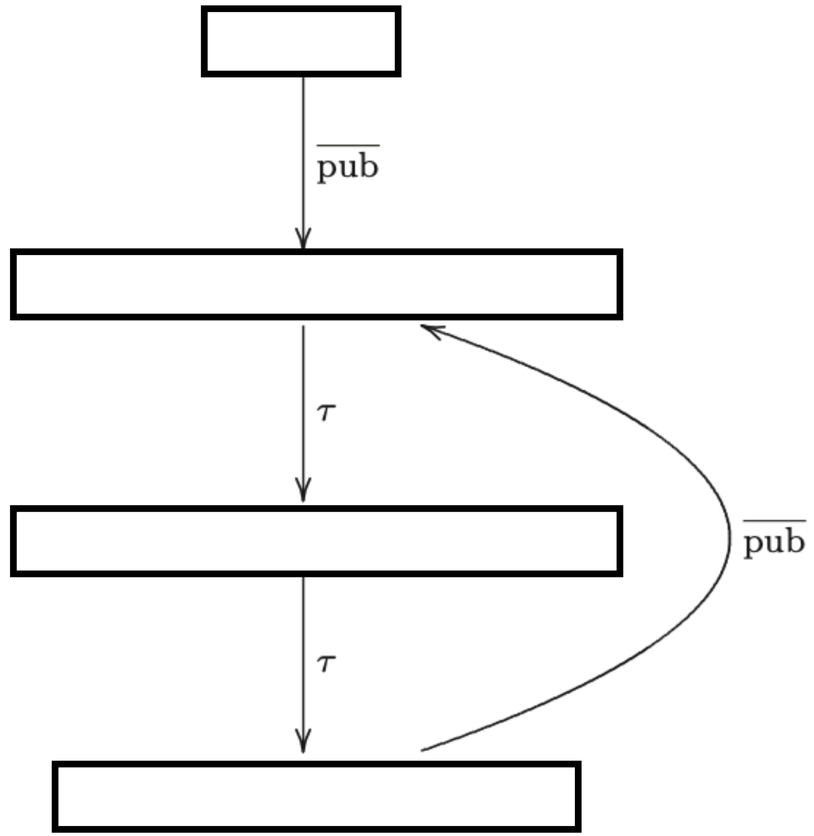
$$\text{SmUni} \stackrel{\text{def}}{=} (\text{CM} \mid \text{CS}) \setminus \text{coin} \setminus \text{coffee}$$

Labelled Transition Systems (LTSs)

CCS syntax + SOS semantics => LTS behaviour models

Definition [Labelled transition system] A *labelled transition system (LTS)* (at times also called a *transition graph*) is a triple $(\mathsf{Proc}, \mathsf{Act}, \{\overset{\alpha}{\rightarrow} \mid \alpha \in \mathsf{Act}\})$, where:

- Proc is a set of *states* (or *processes*);
- Act is a set of *actions* (or *labels*);
- $\overset{\alpha}{\rightarrow} \subseteq \mathsf{Proc} \times \mathsf{Proc}$ is a *transition relation*, for every $\alpha \in \mathsf{Act}$. As usual, we shall use the more suggestive notation $s \overset{\alpha}{\rightarrow} s'$ in lieu of $(s, s') \in \overset{\alpha}{\rightarrow}$, and write $s \overset{\alpha}{\not\rightarrow}$ (read 's refuses a') iff $s \overset{\alpha}{\rightarrow} s'$ for no state s'.



$\mathrm{SmUni} \overset{\mathrm{def}}{=} (\mathrm{CM} \mid \mathrm{CS}) \setminus \mathrm{coin} \setminus \mathrm{coffee}$

Many Other Calculi
(Selection)

- J.C.M. Baeten, W.P. Weijland. *Process Algebra* [ACP]. Cambridge University Press, 1990.
- J.C.M. Baeten, J.A. Bergstra, S.A. Smolka. *Axiomatizing probabilistic processes: ACP with generative probabilities*. Information and Computation, 1995.
- R. Milner: *Communicating and mobile systems - the Pi-calculus*. Cambridge University Press 1999.
- L. Cardelli, A.D. Gordon. *Mobile ambients* [ambient calculus]. TCS, 2000.
  - ETAPS 2024 Test-of-Time Award
- J.F. Groote, M.R. Mousavi: *Modeling and Analysis of Communicating Systems* [mCRL2 with data and time]. MIT Press 2014.

Questions?

Correctness of CCS Models

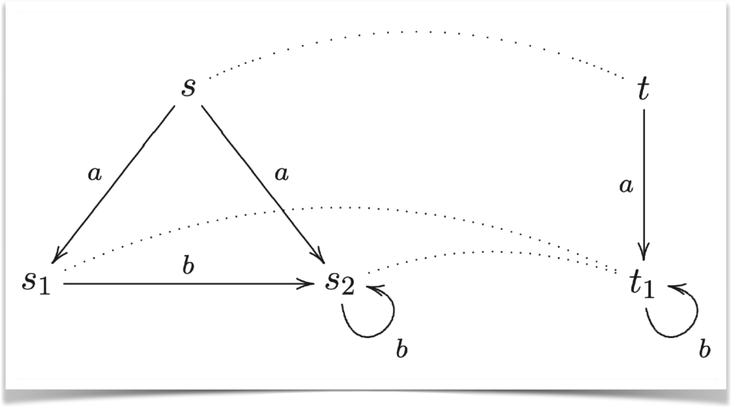
Verification

- How do I know my CCS model behaves as expected?
  - $Impl$, $Spec$
  - $Impl \;\; R \;\; Spec$?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - $Impl, Spec$ — CCS process terms / LTSs
  - $E \vdash Impl = Spec$?
    - Syntactic equivalence / axiomatisation E
    - $Impl, Spec$ — CCS process terms
  - $Impl \models Spec$?
    - Model checking
    - $Impl$ — CCS process term / LTS, $Spec$ — logical formula (e.g., Hennessy-Milner logic)

Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - $Impl$, $Spec$
  - $Impl \;\; R \;\; Spec$?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - $Impl, Spec$ — CCS process terms / LTSs
  - $E \vdash Impl = Spec$?
    - Syntactic equivalence / axiomatisation E
    - $Impl, Spec$ — CCS process terms
  - $Impl \models Spec$?
    - Model checking
    - $Impl$ — CCS process term / LTS, $Spec$ — logical formula (e.g., Hennessy-Milner logic)



Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - $Impl$, $Spec$
  - $Impl \;\; R \;\; Spec$?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - $Impl, Spec$ — CCS process terms / LTSs
  - $E \vdash Impl = Spec$?
    - Syntactic equivalence / axiomatisation E
    - $Impl, Spec$ — CCS process terms
  - $Impl \models Spec$?
    - Model checking
    - $Impl$ — CCS process term / LTS, $Spec$ — logical formula (e.g., Hennessy-Milner logic)

$$x + y = y + x$$

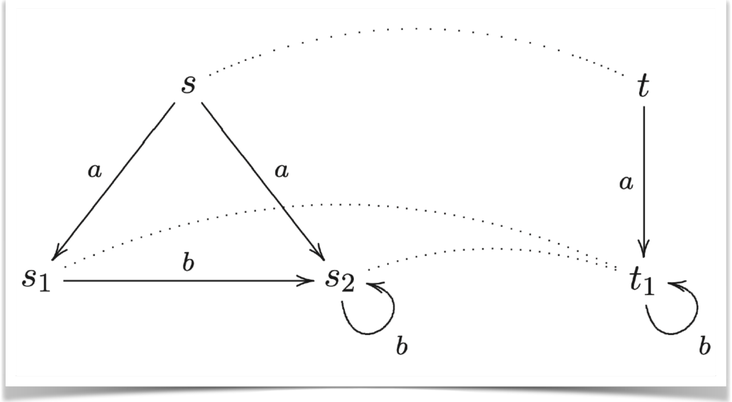
$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$x \,|\, y = y \,|\, x$$

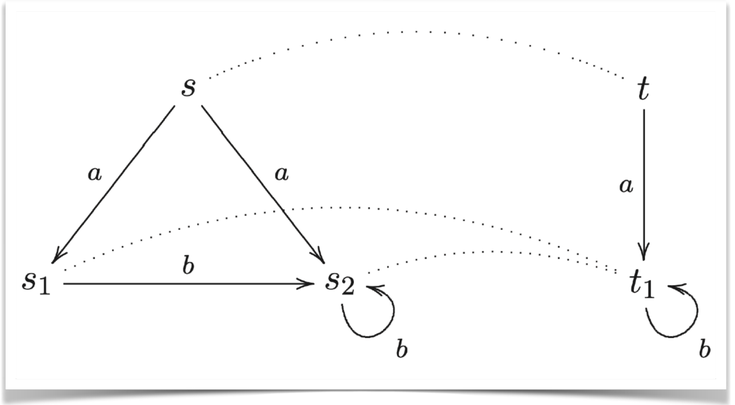
$$x \,|\, (y \,|\, z) = (x \,|\, y) \,|\, z$$



Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - $Impl$, $Spec$
  - $Impl \;\; R \;\; Spec$?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - $Impl, Spec$ — CCS process terms / LTSs
  - $E \vdash Impl = Spec$?
    - Syntactic equivalence / axiomatisation E
    - $Impl, Spec$ — CCS process terms
  - $Impl \models Spec$?
    - Model checking
    - $Impl$ — CCS process term / LTS, $Spec$ — logical formula (e.g., Hennessy-Milner logic)



$$x + y = y + x$$

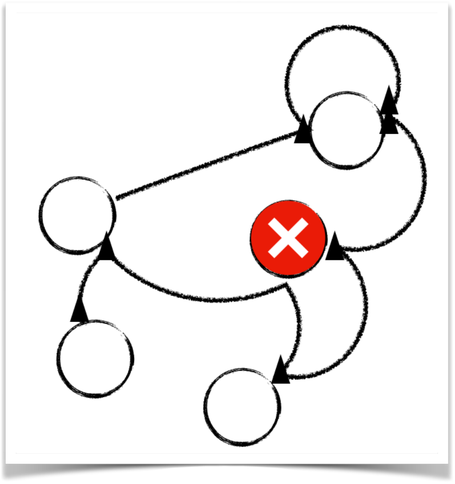
$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$x \,|\, y = y \,|\, x$$

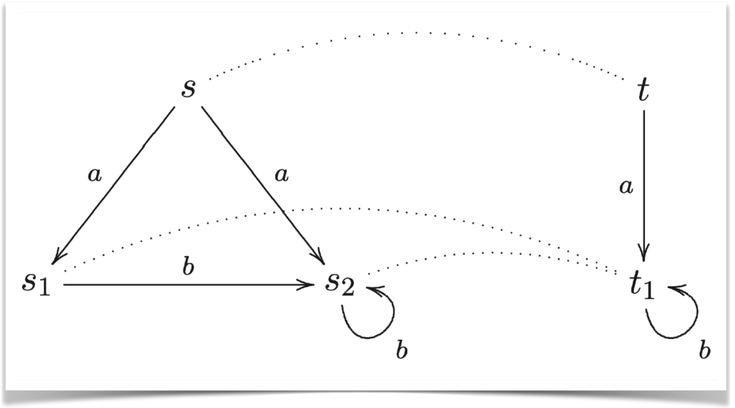
$$x \,|\, (y \,|\, z) = (x \,|\, y) \,|\, z$$



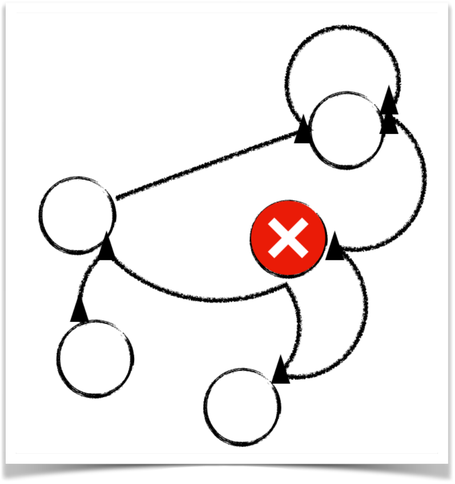
Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - $Impl$, $Spec$
  - $Impl \;\; R \;\; Spec$?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - $Impl, Spec$ — CCS process terms / LTSs
  - $E \vdash Impl = Spec$?
    - Syntactic equivalence / axiomatisation E
    - $Impl, Spec$ — CCS process terms
  - $Impl \models Spec$?
    - Model checking
    - $Impl$ — CCS process term / LTS, $Spec$ — logical formula (e.g., Hennessy-Milner logic)



$$x + y = y + x$$
$$x + (y + z) = (x + y) + z$$
$$x + x = x$$
$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$
$$x \,|\, y = y \,|\, x$$
$$x \,|\, (y \,|\, z) = (x \,|\, y) \,|\, z$$



Verification by Behavioural Equivalence Checking

“Good” notion of behavioural compliance (R)?

- Stakeholder requirement: The coffee machine should deliver coffee & tea :-)

Requirements Engineer: *Spec*

$$\mathrm{CTM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$$

Software Developer: *Impl*

$$\mathrm{CTM}' \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CTM}' + \mathrm{coin}.\overline{\mathrm{tea}}.\mathrm{CTM}'$$

$$Impl \;\; R \;\; Spec\,?$$

Verification by Behavioural Equivalence Checking

Trace Equivalence

Trace semantics. $\sigma \in Act^{\star}$ is a *trace* of a process p, if there is a process q, such that $p \xrightarrow{\sigma} q$. Let $T(p)$ denote the set of traces of p. Two processes p and q are *trace equivalent* if $T(p) = T(q)$. In trace semantics two processes are identified iff they are trace equivalent.

$$\mathrm{CTM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM}) \qquad \mathrm{CTM}' \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CTM}' + \mathrm{coin}.\overline{\mathrm{tea}}.\mathrm{CTM}'$$

T(CTM) = T(CTM')

Verification by Behavioural Equivalence Checking

"Good" notion of behavioural compliance (R)?

- The states that *Spec* and *Impl* reach should be equivalent

Definition [Strong bisimulation] A binary relation $\mathcal{R}$ over the set of states of an LTS is a *bisimulation* iff whenever $s_1 \; \mathcal{R} \; s_2$ and α is an action:

- if $s_1 \xrightarrow{\alpha} s_1'$, then there is a transition $s_2 \xrightarrow{\alpha} s_2'$ such that $s_1' \; \mathcal{R} \; s_2'$;
- if $s_2 \xrightarrow{\alpha} s_2'$, then there is a transition $s_1 \xrightarrow{\alpha} s_1'$ such that $s_1' \; \mathcal{R} \; s_2'$.

Two states s and s' are *bisimilar*, written $s \sim s'$, iff there is a bisimulation that relates them. Henceforth the relation $\sim$ will be referred to as *strong bisimulation equivalence* or *strong bisimilarity*.

$$\sim \; = \bigcup \{ \mathcal{R} \in 2^{(\mathsf{Proc} \times \mathsf{Proc})} \mid \mathcal{R} \text{ is a strong bisimulation} \}$$

Impl $\sim$ *Spec*

Strong Bisimilarity (~)

Examples

Definition [Strong bisimulation] A binary relation $\mathcal{R}$ over the set of states of an LTS is a *bisimulation* iff whenever $s_1 \; \mathcal{R} \; s_2$ and α is an action:

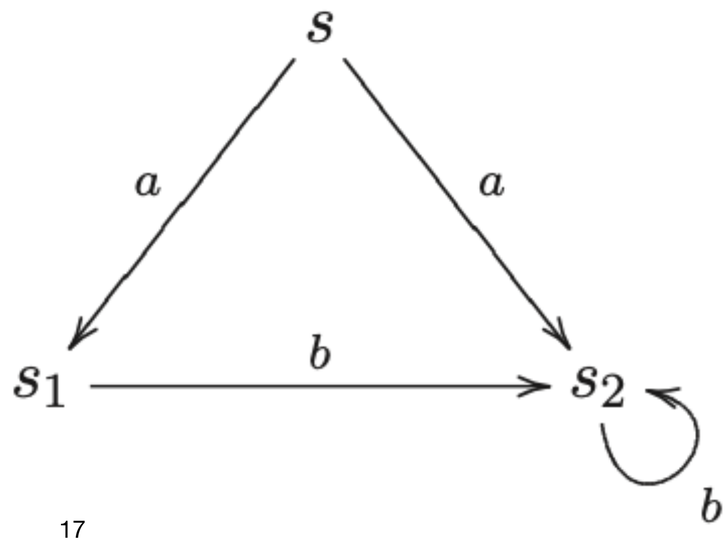
- if $s_1 \xrightarrow{\alpha} s_1'$, then there is a transition $s_2 \xrightarrow{\alpha} s_2'$ such that $s_1' \; \mathcal{R} \; s_2'$;
- if $s_2 \xrightarrow{\alpha} s_2'$, then there is a transition $s_1 \xrightarrow{\alpha} s_1'$ such that $s_1' \; \mathcal{R} \; s_2'$.

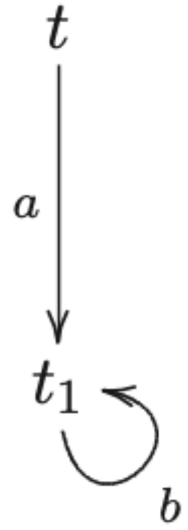
Two states s and s' are *bisimilar*, written $s \sim s'$, iff there is a bisimulation that relates them. Henceforth the relation $\sim$ will be referred to as *strong bisimulation equivalence* or *strong bisimilarity*.

$$\sim \; = \bigcup \{\mathcal{R} \in 2^{(\mathsf{Proc} \times \mathsf{Proc})} \mid \mathcal{R} \text{ is a strong bisimulation}\}$$

$$s \sim t?$$

$$R = \ldots ?$$





Strong Bisimilarity (~)

Examples

Definition [Strong bisimulation] A binary relation $\mathcal{R}$ over the set of states of an LTS is a *bisimulation* iff whenever $s_1 \; \mathcal{R} \; s_2$ and α is an action:

- if $s_1 \xrightarrow{\alpha} s_1'$, then there is a transition $s_2 \xrightarrow{\alpha} s_2'$ such that $s_1' \; \mathcal{R} \; s_2'$;
- if $s_2 \xrightarrow{\alpha} s_2'$, then there is a transition $s_1 \xrightarrow{\alpha} s_1'$ such that $s_1' \; \mathcal{R} \; s_2'$.

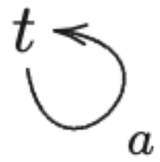
Two states s and s' are *bisimilar*, written $s \sim s'$, iff there is a bisimulation that relates them. Henceforth the relation $\sim$ will be referred to as *strong bisimulation equivalence* or *strong bisimilarity*.

$$\sim \; = \bigcup \{\mathcal{R} \in 2^{(\mathsf{Proc} \times \mathsf{Proc})} \mid \mathcal{R} \text{ is a strong bisimulation}\}$$

$$s \sim t?$$

$$R = \ldots ?$$

$$s_1 \xrightarrow{a} s_2 \xrightarrow{a} s_3 \xrightarrow{a} s_4 \xrightarrow{a} \cdots$$



Strong Bisimilarity (~)

Examples

Definition [Strong bisimulation] A binary relation $\mathcal{R}$ over the set of states of an LTS is a *bisimulation* iff whenever $s_1 \; \mathcal{R} \; s_2$ and α is an action:

- if $s_1 \xrightarrow{\alpha} s_1'$, then there is a transition $s_2 \xrightarrow{\alpha} s_2'$ such that $s_1' \; \mathcal{R} \; s_2'$;
- if $s_2 \xrightarrow{\alpha} s_2'$, then there is a transition $s_1 \xrightarrow{\alpha} s_1'$ such that $s_1' \; \mathcal{R} \; s_2'$.

Two states s and s' are *bisimilar*, written $s \sim s'$, iff there is a bisimulation that relates them. Henceforth the relation $\sim$ will be referred to as *strong bisimulation equivalence* or *strong bisimilarity*.

$$\sim \; = \bigcup \{\mathcal{R} \in 2^{(\mathsf{Proc} \times \mathsf{Proc})} \mid \mathcal{R} \text{ is a strong bisimulation}\}$$

$$CTM \sim CTM'?$$

$$R = \ldots ?$$

$$\mathrm{CTM}' \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CTM}' + \mathrm{coin}.\overline{\mathrm{tea}}.\mathrm{CTM}'$$

$$\mathrm{CTM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$$

Strong Bisimilarity (~)

Properties

Theorem For all LTSs, the relation $\sim$ is

1. an equivalence relation,
2. the largest strong bisimulation, and
3. satisfies the following property:

   $s_1 \sim s_2$ iff for each action α,

   - if $s_1 \xrightarrow{\alpha} s_1'$, then there is a transition $s_2 \xrightarrow{\alpha} s_2'$ such that $s_1' \sim s_2'$;
   - if $s_2 \xrightarrow{\alpha} s_2'$, then there is a transition $s_1 \xrightarrow{\alpha} s_1'$ such that $s_1' \sim s_2'$.

Exercise (optional): Prove the theorem above

Strong Bisimilarity (~)

Complexity

- Deciding strong bisimilarity between finite LTSs (finite no. of transitions and states) is P-complete
  - (See Balcazar, Gabarro and Santha, 1992)
  - one of the “hardest problems” in the class of problems solvable in polynomial time
  - P-complete problems are of interest because they appear to lack highly parallel solutions; (see, Greenlaw, Hoover and Ruzzo, 1995)

Strong Bisimilarity (~)

Fixed Point

Theorem [Tarski's fixed point theorem] Let $(D, \sqsubseteq)$ be a complete lattice, and let $f : D \to D$ be monotonic. Then f has a largest fixed point $z_{\max}$ and a least fixed point $z_{\min}$ given by

$$z_{\max} = \bigsqcup \{x \in D \mid x \sqsubseteq f(x)\} \quad \text{and}$$
$$z_{\min} = \bigsqcap \{x \in D \mid f(x) \sqsubseteq x\} \ .$$

Show that bisimilarity ($\sim$) is a largest fixed point, where:

$$\sim \ = \bigcup \{\mathcal{R} \in 2^{(\mathsf{Proc} \times \mathsf{Proc})} \mid \mathcal{R} \text{ is a strong bisimulation}\}$$

Exercise (optional): Prove the statement above

Strong Bisimilarity (~)

Fixed Point

Theorem [Tarski's fixed point theorem] Let $(D, \sqsubseteq)$ be a complete lattice, and let $f : D \to D$ be monotonic. Then f has a largest fixed point $z_{\max}$ and a least fixed point $z_{\min}$ given by

$$z_{\max} = \bigsqcup \{x \in D \mid x \sqsubseteq f(x)\} \quad \text{and}$$
$$z_{\min} = \bigsqcap \{x \in D \mid f(x) \sqsubseteq x\} \ .$$

Hints:

First, identify your relevant complete lattice structure.

Consider now a binary relation $\mathcal{R}$ over Proc—that is, an element of the set $2^{(\mathsf{Proc} \times \mathsf{Proc})}$. We define the set $\mathcal{F}(\mathcal{R})$ as follows:

$(p, q) \in \mathcal{F}(\mathcal{R})$, for all $p, q \in \mathsf{Proc}$, if and only if

1. $p \xrightarrow{\alpha} p'$ implies $q \xrightarrow{\alpha} q'$ for some q' such that $(p', q') \in \mathcal{R}$;
2. $q \xrightarrow{\alpha} q'$ implies $p \xrightarrow{\alpha} p'$ for some p' such that $(p', q') \in \mathcal{R}$.

$\mathcal{R}$ is a bisimulation if and only if $\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})$

Consequently:

$$\sim \ = \bigcup \{\mathcal{R} \in 2^{(\mathsf{Proc} \times \mathsf{Proc})} \mid \mathcal{R} \subseteq \mathcal{F}(\mathcal{R})\}$$

Semantic Lattice

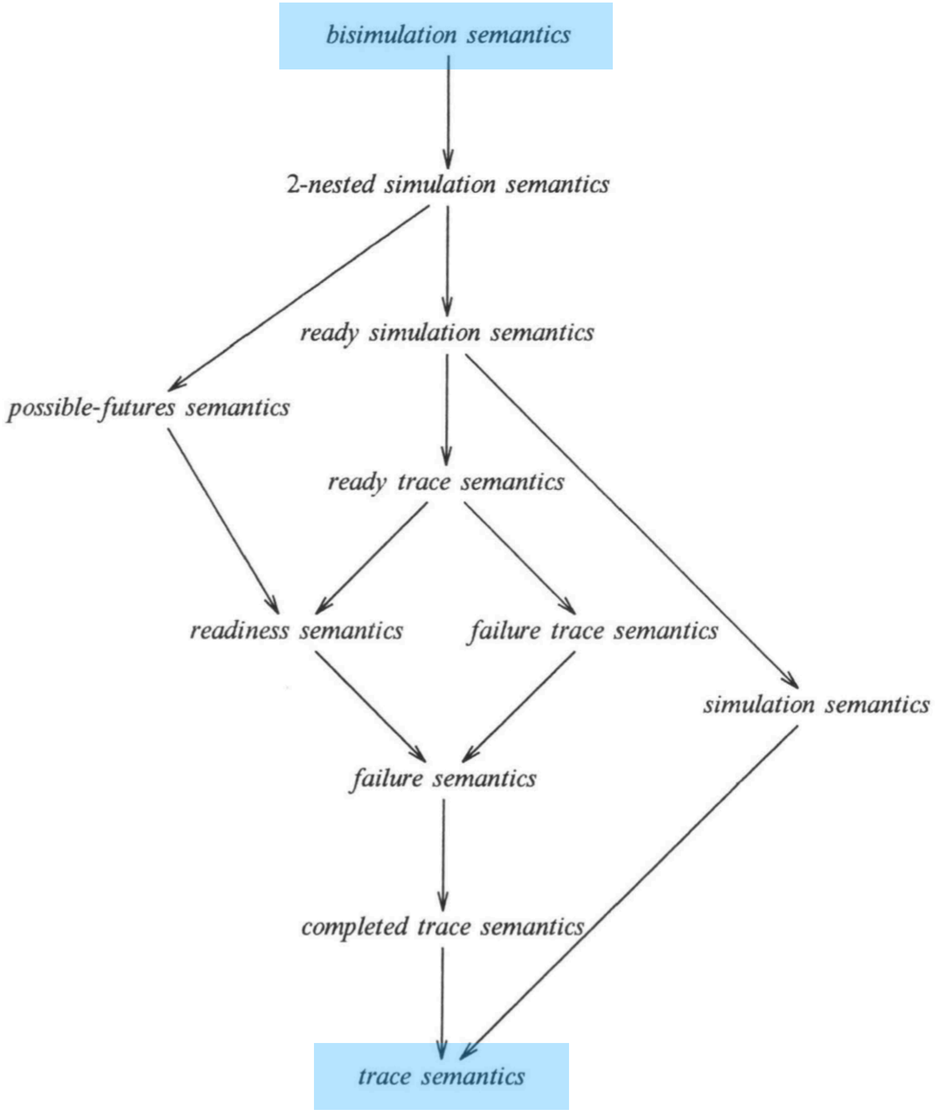


FIGURE 1. *The linear time - branching time spectrum*

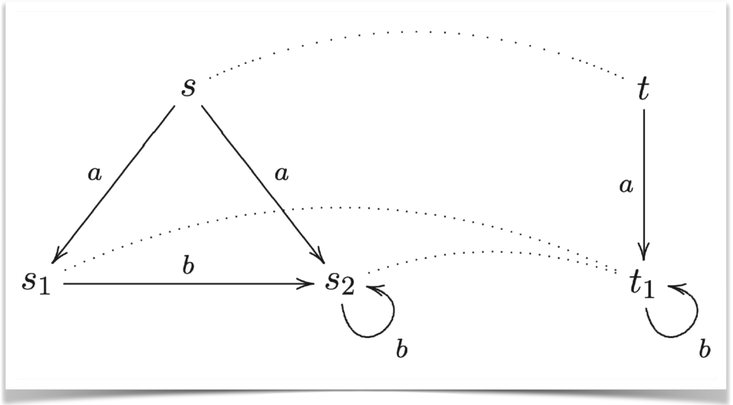
R.J. van Glabbeek: *The Linear Time - Branching Time Spectrum I.* Handbook of Process Algebra 2001

Questions?

Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - $Impl$, $Spec$
  - $Impl \;\; R \;\; Spec$?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - $Impl, Spec$ — CCS process terms / LTSs
  - $E \vdash Impl = Spec$?
    - Syntactic equivalence / axiomatisation E
    - $Impl, Spec$ — CCS process terms
  - $Impl \models Spec$?
    - Model checking
    - $Impl$ — CCS process term / LTS, $Spec$ — logical formula (e.g., Hennessy-Milner logic)



$$x + y = y + x$$

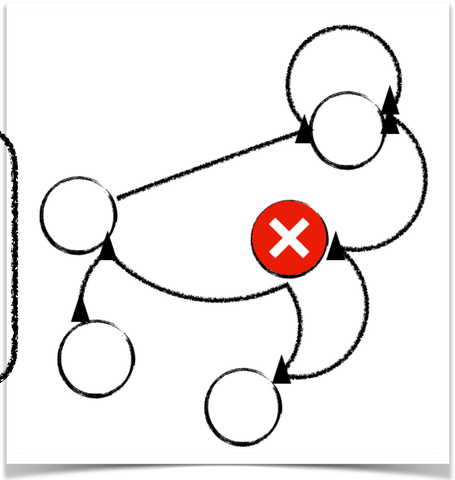
$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$x \,|\, y = y \,|\, x$$

$$x \,|\, (y \,|\, z) = (x \,|\, y) \,|\, z$$



Hennessy-Milner Logic (HML)

Specification language

Definition The set $\mathcal{M}$ of *Hennessy-Milner formulae* over a set of actions Act is given by the following abstract syntax:

$$F, G ::= tt \mid ff \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a]F \ ,$$

where $a \in \mathsf{Act}$, and we use tt and ff to denote 'true' and 'false', respectively. If

Definition [Denotational semantics] We define $[\![F]\!] \subseteq \mathsf{Proc}$ for $F \in \mathcal{M}$ by

1. $[\![tt]\!] = \mathsf{Proc}$
2. $[\![ff]\!] = \emptyset$
3. $[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!]$
4. $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$
5. $[\![\langle a \rangle F]\!] = \langle \cdot a \cdot \rangle [\![F]\!]$
6. $[\![[a]F]\!] = [\cdot a \cdot][\![F]\!]$,

where we use the set operators $\langle \cdot a \cdot \rangle, [\cdot a \cdot] : 2^{\mathsf{Proc}} \to 2^{\mathsf{Proc}}$ defined by

$$\begin{aligned} \langle \cdot a \cdot \rangle S &= \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ and } p' \in S, \text{ for some } p'\} \quad \text{and} \\ [\cdot a \cdot] S &= \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ implies } p' \in S, \text{ for each } p'\} \ . \end{aligned}$$

We write $p \models F$, read 'p satisfies F', iff $p \in [\![F]\!]$.

Two formulae are *equivalent* if, and only if, they are satisfied by the same processes in every transition system.

Hennessy-Milner Logic (HML)

Example: Safety Requirement - “Something bad never happens”

Definition The set $\mathcal{M}$ of *Hennessy-Milner formulae* over a set of actions Act is given by the following abstract syntax:

$$F, G ::= tt \mid ff \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a]F \ ,$$

where $a \in \mathsf{Act}$, and we use tt and ff to denote ‘true’ and ‘false’, respectively. If

$$\mathrm{CM} \stackrel{\text{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$$

$$\mathrm{CS} \stackrel{\text{def}}{=} \overline{\mathrm{pub}}.\overline{\mathrm{coin}}.\mathrm{coffee}.\mathrm{CS}$$

$$\mathrm{SmUni} \stackrel{\text{def}}{=} (\mathrm{CM} \mid \mathrm{CS}) \setminus \mathrm{coin} \setminus \mathrm{coffee}$$

(Relevant) Safety Requirement in HML?

Definition [Denotational semantics] We define $[\![F]\!] \subseteq \mathsf{Proc}$ for $F \in \mathcal{M}$ by

1. $[\![tt]\!] = \mathsf{Proc}$
2. $[\![ff]\!] = \emptyset$
3. $[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!]$
4. $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$
5. $[\![\langle a \rangle F]\!] = \langle \cdot a \cdot \rangle [\![F]\!]$
6. $[\![[a]F]\!] = [\cdot a \cdot][\![F]\!]$,

where we use the set operators $\langle \cdot a \cdot \rangle, [\cdot a \cdot] : 2^{\mathsf{Proc}} \to 2^{\mathsf{Proc}}$ defined by

$$\langle \cdot a \cdot \rangle S = \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ and } p' \in S, \text{ for some } p'\} \quad \text{and}$$
$$[\cdot a \cdot] S = \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ implies } p' \in S, \text{ for each } p'\} \ .$$

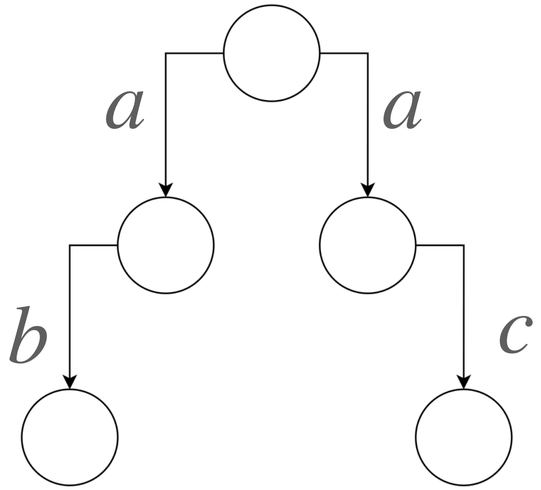
We write $p \models F$, read ‘p satisfies F’, iff $p \in [\![F]\!]$.

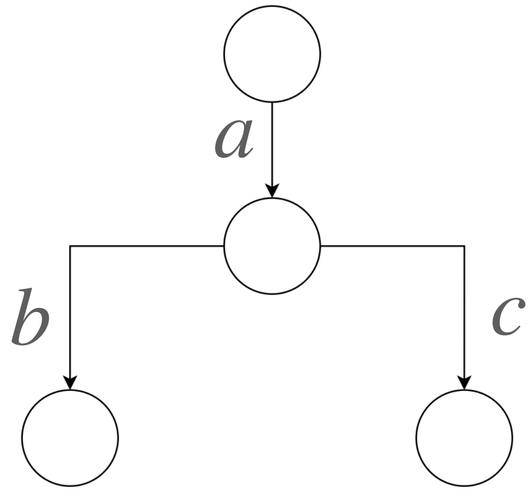
Two formulae are *equivalent* if, and only if, they are satisfied by the same processes in every transition system.

HML & Behavioural Equivalence

Exercise

- Find an HML formula that satisfies $a\,.\,b.0 + a\,.\,c.0$ but not $a\,.\,(b.0 + c.0)$





Recall: $\nsim$

Definition 5.2 [Denotational semantics] We define $[\![F]\!] \subseteq \mathsf{Proc}$ for $F \in \mathcal{M}$ by

1. $[\![tt]\!] = \mathsf{Proc}$
2. $[\![ff]\!] = \emptyset$
3. $[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!]$
4. $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$
5. $[\![\langle a \rangle F]\!] = \langle \cdot a \cdot \rangle [\![F]\!]$
6. $[\![[a]F]\!] = [\cdot a \cdot][\![F]\!]$,

where we use the set operators $\langle \cdot a \cdot \rangle, [\cdot a \cdot] : 2^{\mathsf{Proc}} \to 2^{\mathsf{Proc}}$ defined by

$$\langle \cdot a \cdot \rangle S = \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ and } p' \in S, \text{ for some } p'\} \quad \text{and}$$
$$[\cdot a \cdot] S = \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ implies } p' \in S, \text{ for each } p'\} \ .$$

HML & Behavioural Equivalence

Definition [Image finite process] A process P is *image finite* iff the collection $\{P' \mid P \xrightarrow{a} P'\}$ is finite for each action a.

An LTS is image finite if so is each of its states.

Example of not image finite process?

Theorem [Hennessy and Milner (Hennessy and Milner, 1985)] Let

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

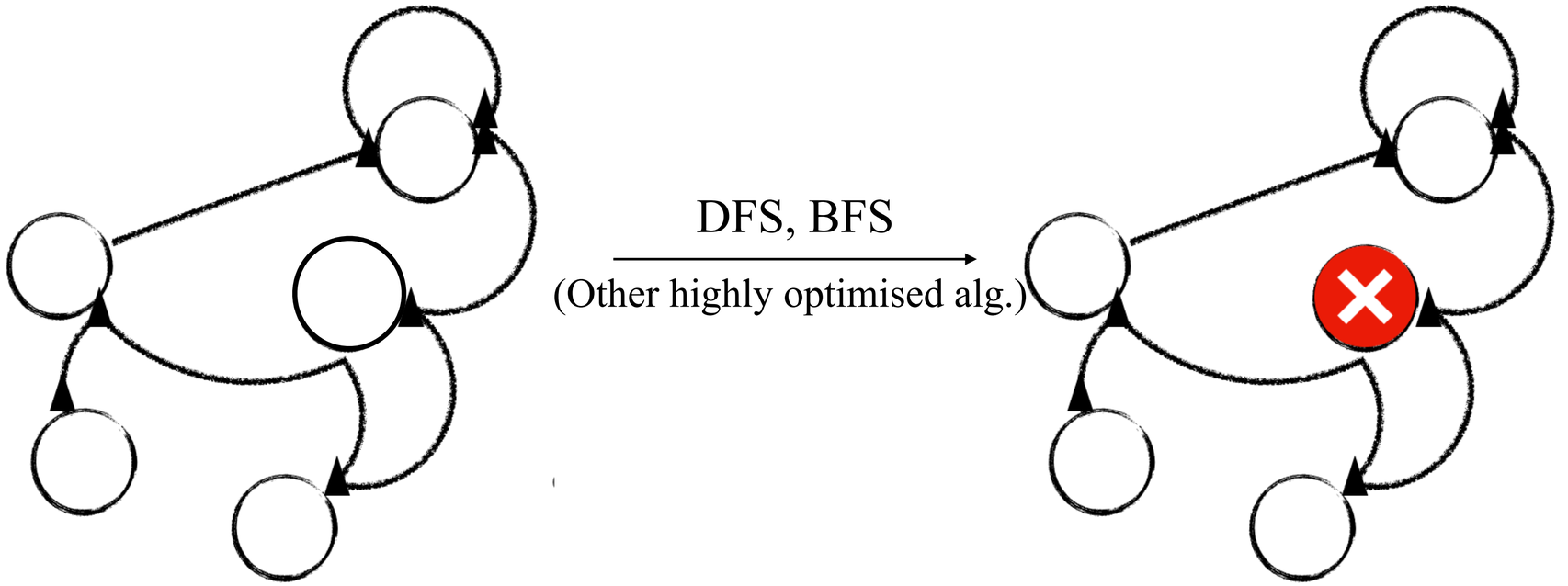
be an image finite LTS. Assume that P, Q are states in Proc. Then $P \sim Q$ iff P and Q satisfy exactly the same formulae in Hennessy-Milner logic.

Model Checking

LTS behaviours against HML specifications (*Impl* ⊨ *Spec*)

Impl - CCS / LTS

Spec - HML formula *F*; e.g., [[< *a* > *ff*]]



Many Other Logics
(Selection)

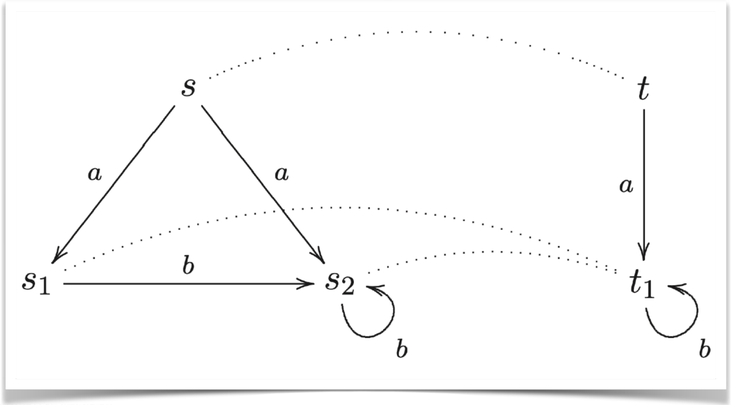
- HML with recursion, data and time
  - Nice read (& mCRL2 tooling): J.F. Groote, M.R. Mousavi. *Modeling and Analysis of Communicating Systems*. MIT Press 2014
- Linear Temporal Logic (LTL)
  - A. Pnueli. *The Temporal Logic of Programs.* FOCS 1977
- Computation Tree Logic (CTL)
  - Nice read: M. Y. Vardi. *Branching vs. Linear Time: Semantical Perspective*. CSL 2011

Questions?

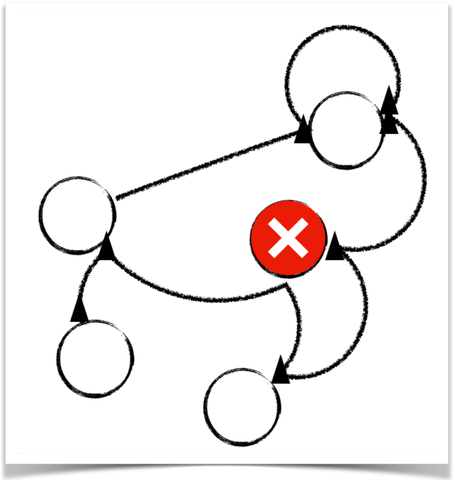
Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - $Impl$, $Spec$
  - $Impl \;\; R \;\; Spec$?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - $Impl, Spec$ — CCS process terms / LTSs
  - $E \vdash Impl = Spec$?
    - Syntactic equivalence / axiomatisation E
    - $Impl, Spec$ — CCS process terms
  - $Impl \models Spec$?
    - Model checking
    - $Impl$ — CCS process term / LTS, $Spec$ — logical formula (e.g., Hennessy-Milner logic)



$$x + y = y + x$$
$$x + (y + z) = (x + y) + z$$
$$x + x = x$$
$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$
$$x \,|\, y = y \,|\, x$$
$$x \,|\, (y \,|\, z) = (x \,|\, y) \,|\, z$$



Verification via Equational Reasoning

- $Impl, Spec$ – CCS process terms
- $E \vdash Impl = Spec\,?$
  - Syntactic equivalence / axiomatisation E
  - Typical axioms / equations E (selection):

$$x + y = y + x$$

$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$x + 0 = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$0.x = 0$$

$$x \,|\, y = y \,|\, x$$

$$x \,|\, (y \,|\, z) = (x \,|\, y) \,|\, z \qquad \ldots$$

Verification via Equational Reasoning

- $Impl, Spec$ – CCS process terms
- $E \vdash Impl = Spec\,?$
  - Syntactic equivalence / axiomatisation E
  - Typical axioms / equations *E* (selection):
- How do I know *E* is “correct”?

Fix a notion of behavioural equivalence (e.g., ~)

Show that *E* is sound & complete:

$\forall p, q \,.\, E \vdash p = q$ iff $p \sim q$

$$x + y = y + x$$

$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$x + 0 = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$0.x = 0$$

$$x \,|\, y = y \,|\, x$$

$$x \,|\, (y \,|\, z) = (x \,|\, y) \,|\, z \qquad \ldots$$

Verification via Equational Reasoning

- $Impl, Spec$ – CCS process terms
- $E \vdash Impl = Spec\,?$
  - Syntactic equivalence / axiomatisation E
  - Typical axioms / equations *E* for ~ (selection):

$$x + y = y + x$$

$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$x + 0 = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$0.x = 0$$

$$x \,|\, y = y \,|\, x$$

$$x \,|\, (y \,|\, z) = (x \,|\, y) \,|\, z \qquad \ldots$$

Verification via Equational Reasoning

- $Impl, Spec$ – CCS process terms
- $E \vdash Impl = Spec\,?$
  - Syntactic equivalence / axiomatisation E
  - Typical axioms / equations *E* for Trace equiv:

$$x + y = y + x$$

$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$x + 0 = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$x \,.\, (y + z) = x \,.\, y + x \,.\, z$$

$$0.x = 0$$

...

Verification via Equational Reasoning

Example

- $Impl, Spec$ – CCS process terms
- $E \vdash Impl = Spec\,?$
  - Syntactic equivalence / axiomatisation E
  - Typical axioms / equations *E* for Trace equiv:

$$\mathrm{CTM}' \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CTM}' + \mathrm{coin}.\overline{\mathrm{tea}}.\mathrm{CTM}'$$

$$\mathrm{CTM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$$

$$E \vdash CTM' = CTM$$

$$x + y = y + x$$

$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$x + 0 = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$x \,.\, (y + z) = x \,.\, y + x \,.\, z$$

$$0.x = 0$$

$$\ldots$$

Sound & Complete Axiomatisation for CCS ~

- For CCS processes that can perform only finite sets of actions
- a finite ground-complete axiomatisation modulo bisimilarity can be obtained
  - J.A. Bergstra, J.W. Klop. *Process algebra for synchronous communication*. Inf. Control 60, 1984

Axiomatisations of Behavioural Equivalence

Challenge

- New behavioural equivalence, or
- New syntax / operator

=>

- derive a new *E*
- prove *E* sound & complete

Axiomatisations of Behavioural Equivalence

- L. Aceto, B. Bloom, F.W. Vaandrager. *Turning SOS Rules into Equations*. Inf. Comput.,1994.
  - Algorithm for generation of sound & complete axiomatisations, for a large class of systems specified in the GSOS format:

DEFINITION 2.1. Suppose Σ is a signature. A *GSOS rule* ρ over Σ is a pair containing a set of transition formulas and a single transition formula, written

$$\frac{\bigcup_{i=1}^{l} \{x_i \xrightarrow{a_{ij}} y_{ij} \mid 1 \leqslant j \leqslant m_i\} \cup \bigcup_{i=1}^{l} \{x_i \xrightarrow{b_{ik}} \mid 1 \leqslant k \leqslant n_i\}}{f(x_1, \ldots, x_l) \xrightarrow{c} C[\vec{x}, \vec{y}]}, \qquad (7)$$

where all the variables are distinct, $m_i, n_i \geqslant 0$, f is an operation symbol from Σ with arity l, and $C[\vec{x}, \vec{y}]$ is a Σ-context with variables including at most the x_i's and y_{ij}'s. (It need not contain all these variables.) Note that a_{ij}, b_{ij}, and c are actions and not, as for instance in [47], variables ranging over actions.

Rule Formats for “Well-behaved” Behavioural Equivalences

(Selection)

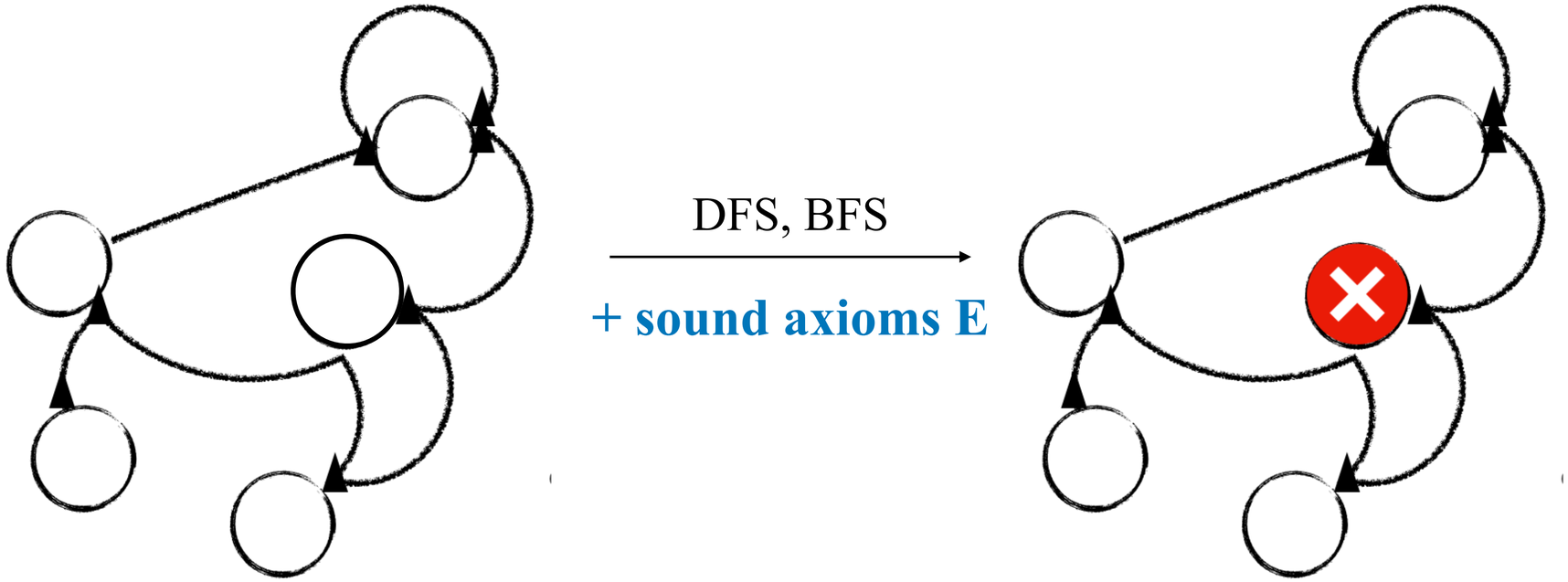
- “Well-behaved”
  - e.g., congruence: $P \sim Q \implies C[P] \sim C[Q]$
- **GSOS**
  - B. Bloom, S. Istrail, A. R. Meyer. *Bisimulation can’t be traced*. JACM, 1995.
- **NTYFT / NTYXT**
  - J.F. Groote. *Transition System Specifications with Negative Premises.* TCS, 1993.

Model Checking Modulo Sound Axioms

Potential state-space reduction —> Speedup

Impl - CCS / LTS

Spec - HML formula *F*; e.g., [[$< a > ff$]]



Questions?

CCS Process Terms

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

Examples

$\mathrm{CM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

$\mathrm{CTM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$

$\mathrm{CS} \stackrel{\mathrm{def}}{=} \overline{\mathrm{pub}}.\overline{\mathrm{coin}}.\mathrm{coffee}.\mathrm{CS}$

$\mathrm{SmUni} \stackrel{\mathrm{def}}{=} (\mathrm{CM} \mid \mathrm{CS}) \setminus \mathrm{coin} \setminus \mathrm{coffee}$

$\mathrm{CHM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{choc}}.\mathrm{CHM}$

$\mathrm{DFM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{figs}}.\mathrm{DFM}$

$\mathrm{CRM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{crisps}}.\mathrm{CRM}$

$\mathrm{VM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{item}}.\mathrm{VM}$

$\mathrm{CHM} \stackrel{\mathrm{def}}{=} \mathrm{VM}[\mathrm{choc}/\mathrm{item}]$

CCS Process Terms

Examples

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

$\mathrm{CM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

$\mathrm{CTM} \stackrel{\mathrm{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$

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

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

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

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

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

Behaviour of CCS Process Terms

Example

ACT $\dfrac{}{\alpha.P \xrightarrow{\alpha} P}$

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

COM1 $\dfrac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$

COM2 $\dfrac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}$

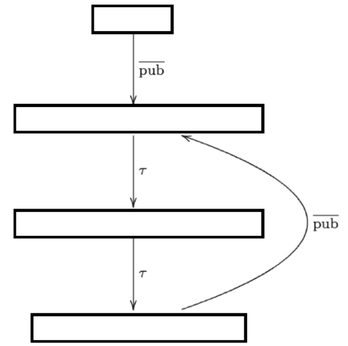
COM3 $\dfrac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$

RES $\dfrac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$ where $\alpha, \bar{\alpha} \notin L$

REL $\dfrac{P \xrightarrow{\alpha}$

$P[f] \xrightarrow{f($

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



CCS Process Terms

Examples

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

$\mathrm{CM} \stackrel{\text{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

$\mathrm{CTM} \stackrel{\text{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$

$\mathrm{CHM} \stackrel{\text{def}}{=}$

$\mathrm{DFM} \stackrel{\text{def}}{=}$

$\mathrm{CRM} \stackrel{\text{def}}{=}$

$\mathrm{VM} \stackrel{\text{def}}{=}$

$\mathrm{CHM} \stackrel{\text{def}}{=}$

Behaviour of CCS Process Terms

Example

$$\text{ACT} \quad \frac{}{\alpha.P \xrightarrow{\alpha} P}$$

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

$$\text{COM1} \quad \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

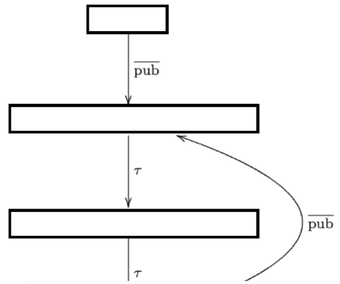
$$\text{COM2} \quad \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}$$

$$\text{COM3} \quad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\text{RES} \quad \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \text{ where } \alpha, \bar{\alpha} \notin L$$

$$\text{REL} \quad \frac{P \xrightarrow{\alpha}}{P[f] \xrightarrow{f(}}$$

$$\text{CON} \quad \frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'} \text{ v}$$

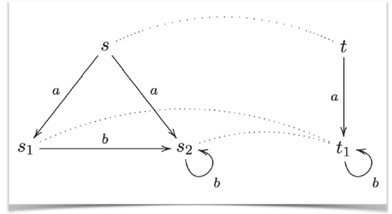


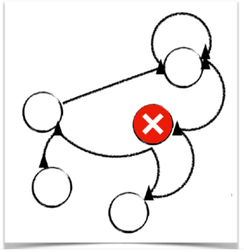
Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - *Impl*, *Spec*
  - *Impl* R *Spec*?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - *Impl*, *Spec* — CCS process terms / LTSs
  - $E \vdash \mathit{Impl} = \mathit{Spec}$?
    - Syntactic equivalence / axiomatisation E
    - *Impl*, *Spec* — CCS process terms
  - *Impl* $\models$ *Spec*?
    - Model checking
    - *Impl* — CCS process term / LTS, *Spec* — logical formula (e.g., Hennessy-Milner logic)

$$x + y = y + x$$
$$x + (y + z) = (x + y) + z$$
$$x + x = x$$
$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$
$$x \mid y = y \mid x$$
$$x \mid (y \mid z) = (x \mid y) \mid z$$





CCS Process Terms

Examples

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

$\mathrm{CM} \stackrel{\text{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

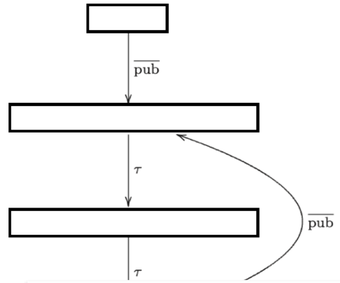
$\mathrm{CTM} \stackrel{\text{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$

$\mathrm{CHM} \stackrel{\text{def}}{=}$

Behaviour of CCS Process Terms

Example

$$\text{ACT}\ \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \text{RES}\ \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}\ \text{where}\ \alpha, \bar{\alpha} \notin L$$



Semantic Lattice

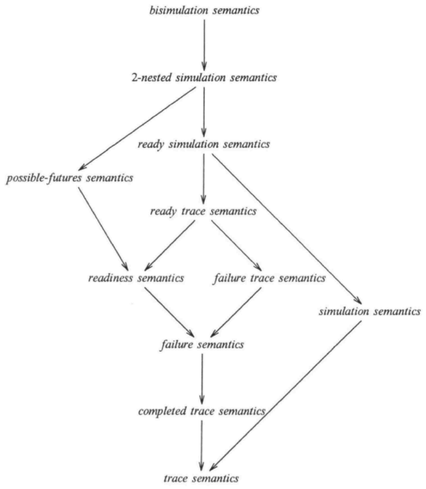


FIGURE 1. *The linear time - branching time spectrum*

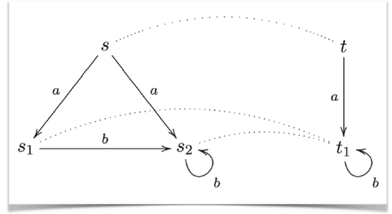
R.J. van Glabbeek: *The Linear Time - Branching Time Spectrum I.* Handbook of Process Algebra 2001

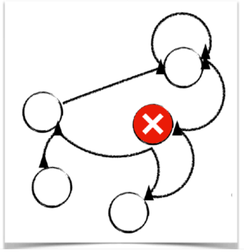
Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - *Impl*, *Spec*
  - *Impl* *R* *Spec* ?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - *Impl*, *Spec* – CCS process terms / LTSs
  - $E \vdash \mathit{Impl} = \mathit{Spec}$?
    - Syntactic equivalence / axiomatisation E
    - *Impl*, *Spec* – CCS process terms
  - *Impl* ⊨ *Spec* ?
    - Model checking
    - *Impl* – CCS process term / LTS, *Spec* – logical formula (e.g., Hennessy-Milner logic)

$$x + y = y + x$$
$$x + (y + z) = (x + y) + z$$
$$x + x = x$$
$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$
$$x \mid y = y \mid x$$
$$x \mid (y \mid z) = (x \mid y) \mid z$$





CCS Process Terms

Examples

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

$\mathrm{CM} \stackrel{\text{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

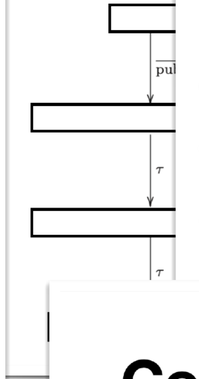
$\mathrm{CTM} \stackrel{\text{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$

$\mathrm{CHM} \stackrel{\text{def}}{=}$

Behaviour of CCS Process Terms

Example

$$\text{ACT}\ \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \text{RES}\ \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}\ \text{where } \alpha, \bar{\alpha} \notin L$$



$\mathrm{CM} \stackrel{\text{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

$\mathrm{CS} \stackrel{\text{def}}{=} \overline{\mathrm{pub}}.\overline{\mathrm{coin}}.\mathrm{coffee}.\mathrm{CS}$

$\mathrm{SmUni} \stackrel{\text{def}}{=} (\mathrm{CM} \mid \mathrm{CS}) \setminus \mathrm{coin} \setminus \mathrm{coffee}$

(Relevant) Safety HML specification?

Hennessy-Milner Logic (HML)

Example: Safety - "Something bad never happens"

Definition The set $\mathcal{M}$ of *Hennessy-Milner formulae* over a set of actions Act is given by the following abstract syntax:

$$F, G ::= tt \mid ff \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a] F \ ,$$

where $a \in \mathsf{Act}$, and we use tt and ff to denote 'true' and 'false', respectively. If

Definition [Denotational semantics] We define $[\![F]\!] \subseteq \mathsf{Proc}$ for $F \in \mathcal{M}$ by

1. $[\![tt]\!] = \mathsf{Proc}$
2. $[\![ff]\!] = \emptyset$
3. $[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!]$
4. $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$
5. $[\![\langle a \rangle F]\!] = \langle \cdot a \cdot \rangle [\![F]\!]$
6. $[\![[a] F]\!] = [\cdot a \cdot][\![F]\!]$,

where we use the set operators $\langle \cdot a \cdot \rangle, [\cdot a \cdot] : 2^{\mathsf{Proc}} \to 2^{\mathsf{Proc}}$ defined by

$$\begin{aligned} \langle \cdot a \cdot \rangle S &= \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ and } p' \in S, \text{ for some } p'\} \quad \text{and} \\ [\cdot a \cdot] S &= \{p \in \mathsf{Proc} \mid p \xrightarrow{a} p' \text{ implies } p' \in S, \text{ for each } p'\} \ . \end{aligned}$$

We write $p \models F$, read 'p satisfies F', iff $p \in [\![F]\!]$.

Two formulae are *equivalent* if, and only if, they are satisfied by the same processes in every transition system.

Semantic Lattice

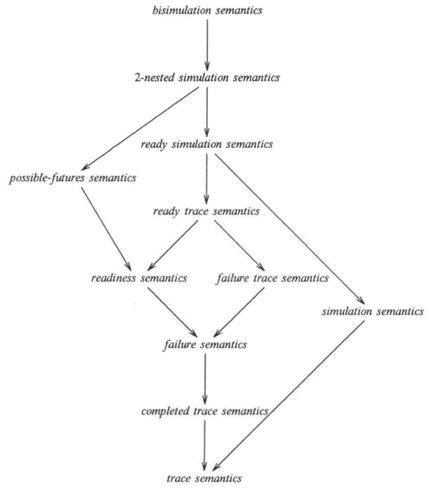


FIGURE 1. *The linear time - branching time spectrum*

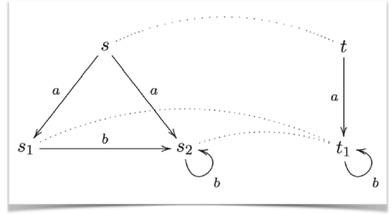
R.J. van Glabbeek: *The Linear Time - Branching Time Spectrum I.* Handbook of Process Algebra 2001

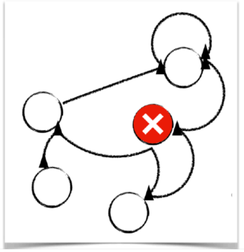
Correctness of CCS Models

Verification

- How do I know my CCS model behaves as expected?
  - *Impl*, *Spec*
  - *Impl* *R* *Spec*?
    - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
    - *Impl*, *Spec* – CCS process terms / LTSs
  - $E \vdash \mathit{Impl} = \mathit{Spec}$?
    - Syntactic equivalence / axiomatisation E
    - *Impl*, *Spec* – CCS process terms
  - *Impl* ⊨ *Spec*?
    - Model checking
    - *Impl* – CCS process term / LTS, *Spec* – logical formula (e.g., Hennessy-Milner logic)

$$\begin{aligned} x + y &= y + x \\ x + (y + z) &= (x + y) + z \\ x + x &= x \\ (x + y) \,.\, z &= x \,.\, z + y \,.\, z \\ x \mid y &= y \mid x \\ x \mid (y \mid z) &= (x \mid y) \mid z \end{aligned}$$





CCS Process Terms

Examples

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

$\mathrm{CM} \stackrel{\text{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

$\mathrm{CTM} \stackrel{\text{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$

$\mathrm{CHM} \stackrel{\text{def}}{=}$

Behaviour of CCS Process Terms

Example

ACT $\dfrac{}{\alpha.P \xrightarrow{\alpha} P}$

RES $\dfrac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$

Semantic Lattice

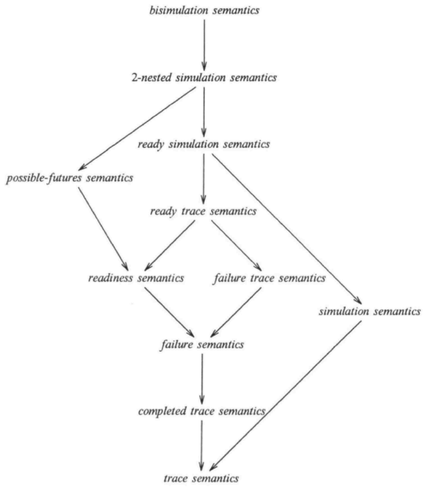


FIGURE 1. *The linear time - branching time spectrum*

R.J. van Glabbeek: *The Linear Time - Branching Time Spectrum I.* Handbook of Process Algebra 2001

Hennessy-Milner Logic (HML)

Example: Safety - "Something bad never happens"

Definition The set $\mathcal{M}$ of *Hennessy-Milner formulae* over a set of actions Act is given by the following abstract syntax:

Definition [Denotational semantics] We define $[\![F]\!] \subseteq$ Proc for $F \in \mathcal{M}$ by

1. $[\![tt]\!] =$ Proc
4. $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$

$= \langle \cdot a \cdot \rangle [\![F]\!]$

$= [\cdot a \cdot][\![F]\!]$,

pc defined by

some $p'\}$ and

for each $p'\}$.

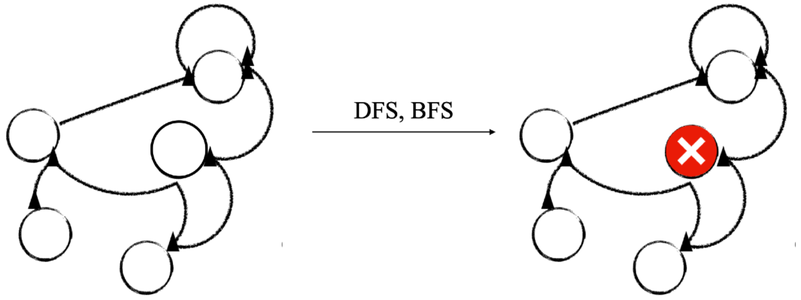
e satisfied by the same

Model Checking

LTS behaviours against HML specifications (*Impl* ⊨ *Spec*)

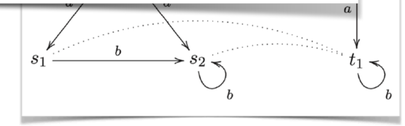
Impl - CCS / LTS

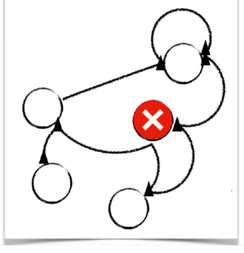
Spec - HML formula *F*; e.g., [[< *a* > *ff*]]



- *Impl R Spec* ?
  - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
  - *Impl*, *Spec* – CCS process terms / LTSs
- $E \vdash Impl = Spec$?
  - Syntactic equivalence / axiomatisation E
  - *Impl*, *Spec* – CCS process terms
- *Impl* ⊨ *Spec* ?
  - Model checking
  - *Impl* – CCS process term / LTS, *Spec* – logical formula (e.g., Hennessy-Milner logic)

$$x + y = y + x$$
$$x + (y + z) = (x + y) + z$$
$$x + x = x$$
$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$
$$x \mid y = y \mid x$$
$$x \mid (y \mid z) = (x \mid y) \mid z$$





CCS Process Terms

Examples

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

$\mathrm{CM} \stackrel{\text{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

$\mathrm{CTM} \stackrel{\text{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$

$\mathrm{CHM} \stackrel{\text{def}}{=}$

Behaviour of CCS Process Terms

Example

ACT $\dfrac{}{\alpha.P \xrightarrow{\alpha} P}$

RES $\dfrac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$

Hennessy-Milner Logic (HML)

Example: Safety - “Something bad never happens”

Definition The set $\mathcal{M}$ of *Hennessy-Milner formulae* over a set of actions Act is given by the following abstract syntax:

Definition [Denotational semantics] We define $[\![F]\!] \subseteq$ Proc for $F \in \mathcal{M}$ by

1. $[\![tt]\!] =$ Proc
4. $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$

$= \langle \cdot a \cdot \rangle [\![F]\!]$

$= [\cdot a \cdot][\![F]\!]$,

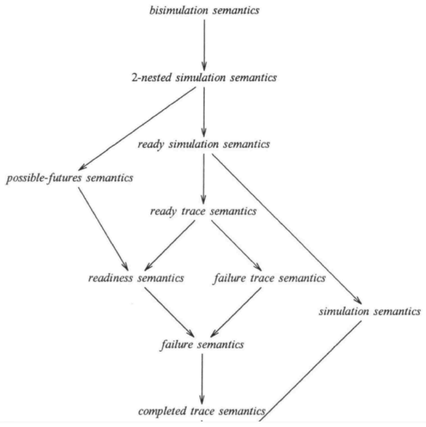
pc defined by

some $p'\}$ and

for each $p'\}$.

e satisfied by the same

Semantic Lattice

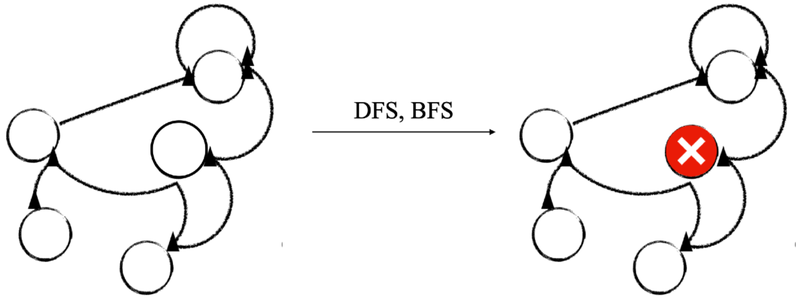


Model Checking

LTS behaviours against HML specifications (*Impl* ⊨ *Spec*)

Impl - CCS / LTS

Spec - HML formula *F*; e.g., [[$< a > ff$]]



31

Verification via Equational Reasoning

- $Impl, Spec$ – CCS process terms
- $E \vdash Impl = Spec$?
  - Syntactic equivalence / axiomatisation E
  - Typical axioms / equations E (selection):
- How do I know E is “correct”?

Fix a notion of behavioural equivalence (e.g., ~)

Show that E is sound & complete:

$\forall p, q \,.\, E \vdash p = q$ iff $p \sim q$

$$x + y = y + x$$

$$x + (y + z) = (x + y) + z$$

$$x + x = x$$

$$x + 0 = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$0.x = 0$$

$$x \mid y = y \mid x$$

$$x \mid (y \mid z) = (x \mid y) \mid z$$

36

- $Impl \;\; R \;\; Spec$?
  - trace equivalence, (weak) bisimilarity, ready/failure equivalence, etc.
  - $Impl, Spec$ – CCS process terms / LTSs
- $E \vdash Impl = Spec$?
  - Syntactic equivalence / axiomatisation E
  - $Impl, Spec$ – CCS process terms
- $Impl \vDash Spec$?
  - Model checking
  - $Impl$ – CCS process term / LTS, $Spec$ – logical formula (e.g., Hennessy-Milner logic)

$$x + y = y + x$$

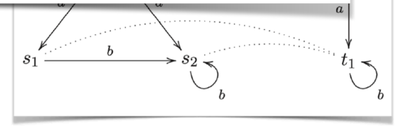
$$x + (y + z) = (x + y) + z$$

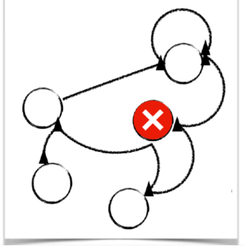
$$x + x = x$$

$$(x + y) \,.\, z = x \,.\, z + y \,.\, z$$

$$x \mid y = y \mid x$$

$$x \mid (y \mid z) = (x \mid y) \mid z$$





12

CCS Process Terms

Examples

$$P, Q ::= K \quad | \quad \alpha.P \quad | \quad \sum_{i \in I} P_i \quad | \quad P \mid Q \quad | \quad P[f] \quad | \quad P \setminus L$$

$\mathrm{CM} \stackrel{\text{def}}{=} \mathrm{coin}.\overline{\mathrm{coffee}}.\mathrm{CM}$

$\mathrm{CTM} \stackrel{\text{def}}{=} \mathrm{coin}.(\overline{\mathrm{coffee}}.\mathrm{CTM} + \overline{\mathrm{tea}}.\mathrm{CTM})$

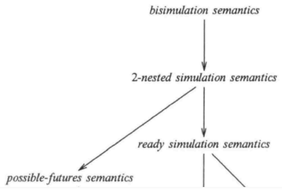
$\mathrm{CHM} \stackrel{\text{def}}{=}$

Behaviour of CCS Process Terms

Example

$$\mathrm{ACT}\ \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \mathrm{RES}\ \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L}$$

Semantic Lattice



Hennessy-Milner Logic (HML)

Example: Safety - "Something bad never happens"

Definition The set $\mathcal{M}$ of *Hennessy-Milner formulae* over a set of actions Act is given by the following abstract syntax:

Definition [Denotational semantics] We define $[\![F]\!] \subseteq \mathsf{Proc}$ for $F \in \mathcal{M}$ by

1. $[\![tt]\!] = \mathsf{Proc}$
4. $[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$

$= \langle \cdot a \cdot \rangle [\![F]\!]$

$= [\cdot a \cdot][\![F]\!]$,

pc defined by

some $p'\}$ and

for each $p'\}$.

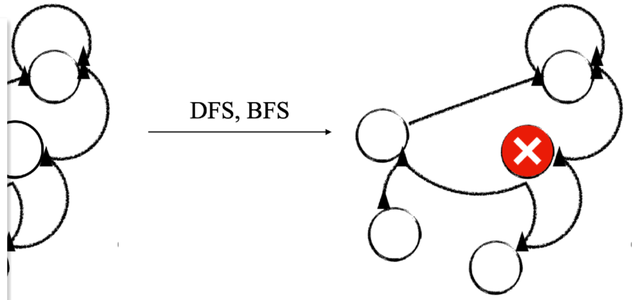
e satisfied by the same

Model Checking

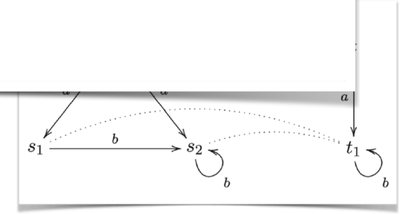
LTS behaviours against HML specifications ($Impl \vDash Spec$)

Impl - CCS / LTS

Spec - HML formula F; e.g., $[[< a > ff]]$



31



?

valence, (weak) bisimilarity, ready/failure equivalence, etc.

– CCS process terms / LTSs

pec ?

equivalence / axiomatisation E

- *Impl*, *Spec* – CCS process terms

$x + y = y + x$

$x + (y + z) = (x + y) + z$

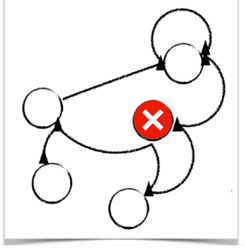
$x + x = x$

$(x + y) \,.\, z = x \,.\, z + y \,.\, z$

$x \mid y = y \mid x$

$x \mid (y \mid z) = (x \mid y) \mid z$

- *Impl* ⊨ *Spec* ?
  - Model checking
  - *Impl* – CCS process term / LTS, *Spec* – logical formula (e.g., Hennessy-Milner logic)



12

Rule Formats for "Well-behaved" Behavioural Equivalences

(Selection)

- "Well-behaved"
  - e.g., congruence: $P \sim Q \implies C[P] \sim C[Q]$
- **GSOS**
  - B. Bloom, S. Istrail, A. R. Meyer. *Bisimulation can't be traced*. JACM, 1995.
- **NTYFT / NTYXT**
  - J.F. Groote. Transition System Specifications with Negative Premises. TCS, 1993.

43

Verification v

- *Impl*, *Spec* – CCS proce
- $E \vdash Impl = Spec$?
  - Syntactic equivalence
  - Typical axioms / equat
- How do I know *E* is "correct"?

Fix a notion of behavioural equivalence (e.g., ~)

Show that *E* is sound & complete:

$\forall p, q \,.\, E \vdash p = q$ iff $p \sim q$

$(x + y) \,.\, z = x \,.\, z + y \,.\, z$

$0.x = 0$

$x \mid y = y \mid x$

$x \mid (y \mid z) = (x \mid y) \mid z$

36