Course "Modelling of Concurrent Systems"
Summer Semester 2016
University of Duisburg-Essen

Harsh Beohar
LF 265,
harsh.beohar@uni-due.de

Course handler

Harsh Beohar

- Room LF 265
- E-Mail: harsh.beohar@uni-due.de
- Meeting by appointment.
- Please send mail only by your official student mail id's.

http://www.uni-due.de/zim/services/e-mail/

Task: Lecturer + Exercise Tutor.

Web-Seite:
http://www.ti.inf.uni-due.de/en/teaching/ss2016/mod-ns/

Lecture schedule

Schedule:

- Monday, 10:00-12:00, in Room LE 120
- Thursday, 12:00-14:00, in Room LE 120

Exercises

Schedule:

(Roughly, every fourth lecture kept for exercises modulo holidays.)

- Thursday, 21/04, 12:00-14:00, in Room LE 120.
- Monday, 09/05, 10:00-12:00, in Room LE 120.
- Monday, 30/05, 10:00-12:00, in Room LE 120.
- Monday, 13/06, 10:00-12:00, in Room LE 120.
- Monday, 27/06, 10:00-12:00, in Room LE 120.
- Monday, 18/06, 10:00-12:00, in Room LE 120.

Idea:

- Problem sheet will be announced in the class, whenever it is published.
- At the same time, also the deadline to submit the exercises will also be involved.
- Please hand in your solutions at the start of the lecture.

Exercises

Scheme:

- In the three best scored exercise sheets, if the sum is more than 50% and once a solution is presented on board then you get a bonus point.
- Effect is improvement by one grade level. E.g. 2.3 to 2.0
- Group solutions are not allowed.

Target audience

MAI

Master "Applied computer science" ("Angewandte Informatik") - focuss engineering or media computer science:

- In the brochure you can find the field of application: "Distributed Reliable Systems" ("Verteilte, Verlässliche Systeme")
  Concurrent systems (Nebenläufige Systeme)

Stundenzahl: *4 SWS (3V + 1Ü), 6 Credits*

Target audience

Master ISE/CE – Verteilte, Verlässliche Systeme

In Master "ISE – Computer Engineering", this lecture is classified as follows:

- Elective "Verteilte, Verlässliche Systeme" (Reliable Systems)
  Stundenzahl: *4 SWS (3V + 1Ü)*

Requirement

Prerequisites:

- Automata and Formal languages.

For the past teaching content, see (although material is considerably different from the last time)
http://www.ti.inf.uni-due.de/teaching/ss2014/mod_ns/

Examination

The exam will be held as a viva voce (oral examination).
Current planned dates:
9th August (Tuesday) and 10th August (Wednesday).

Literature

- Jos Baeten, Twan Basten, and Michel Reniers.

  Process algebra: Equational theories of communicating processes Cambridge University Press, 2010.

  Contents: **(Probabilistic) Process algebra**.

- Luca Aceto, Anna Ingólfsdóttir, Kim G. Larsen, Jiri Srba.

  Reactive Systems: Modelling, Specification and Verification. Cambridge University Press, 2007.

  Contents: **Strong and weak bisimulation, Hennessy-Milner logic, Timed automata**

- Grzegorz Rozenberg.

  Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations World Scientific, 1997.

  Contents: **Graph transformations System**

Literature in Process algebra

- Robin Milner.

  Communication and Concurrency. Prentice Hall, 1989.

  Contents: **Process calculus (CCS), Strong and weak bisimulation, Hennessy-Milner logic.**

- Tony Hoare.

  Communicating sequential processes 2004. Available at `http://www.usingcsp.com/cspbook.pdf`

  Contents: **Process calculus CSP, Failure equivalence**

- Bill Roscoe.

  The Theory and Practice of Concurrency 1997. Available at `http://www.cs.ox.ac.uk/people/bill.roscoe/publications/68b.pdf`.

  Contents: **A reference book on CSP**

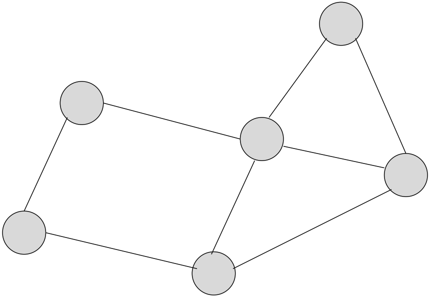
Lecture style

- We will follow the process algebra book. Also, which sections are to be read for the next lecture will be announced in the current one.
- Very few materials will be presented using slides and mostly on blackboard. So please make your own notes!

Motivation

What are concurrent systems?

In general: systems in which several components/processes run concurrently and typically communicate via message passing.



Motivation

Concurrency versus parallelism:

Parallelism

Two events take place in parallel if they are executed at the same moment in time.

Concurrency

Two events are concurrent if they *could potentially* be executed in parallel, but they do not have to. This means there is no causal dependency between them.

Motivation

Concurrency versus parallelism:

Parallelism

Two events take place in parallel if they are executed at the same moment in time.

Concurrency

Two events are concurrent if they *could potentially* be executed in parallel, but they do not have to. This means there is no causal dependency between them.

Hence: concurrency is the more general term.
Examples?

Motivation

(Potential) characteristics of concurrent systems

- Concurrency/parallelism
- Openness (extendability, interaction with the environment)
- Modularity
- Non-terminating behaviour (infinite runs)
- Non-determinism
- Temporal properties (e.g. "an event will occur eventually")

Motivation

Problems with concurrent systems

- Deadlocks
- Guaranteeing mutual exclusion
- Infinite respectively huge state space
- Strongly dynamic behaviour/changing number of processes
- Variable topology/mobility

Motivation

Problems with concurrent systems

- Deadlocks
- Guaranteeing mutual exclusion
- Infinite respectively huge state space
- Strongly dynamic behaviour/changing number of processes
- Variable topology/mobility

Hence: We need methods to model, analyze and verify such systems.

Change in view

Classic view

- Program is a function that transform an input into output.
- Two programs are equivalent if and only if they compute the same output for every input.

Change in view

Classic view

- Program is a function that transform an input into output.
- Two programs are equivalent if and only if they compute the same output for every input.

1. $x = 1;$
2. $x = x + 1;$
3. print x;

Change in view

Classic view

- Program is a function that transform an input into output.
- Two programs are equivalent if and only if they compute the same output for every input.

1. $x = 1;$
2. $x = x + 1;$
3. print $x;$

1. $x = 1;$
2. $x = x \times 2;$
3. print $x;$

Change in view

Classic view

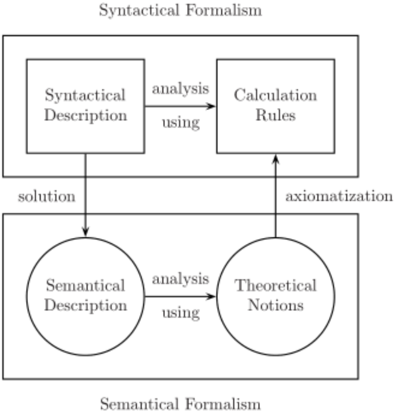
- Program is a function that transform an input into output.
- Two programs are equivalent if and only if they compute the same output for every input.

1. $x = 1;$ 2. $x = x + 1;$ 3. print $x;$	$\|$	1. $x = 1;$ 2. $x = x \times 2;$ 3. print $x;$

Mathematical modelling

- Inspired from traditional engineering disciplines.
- Make system models in formal way.
- Analyse them.
- Then build the ‘real’ system and test it against the models.

Mathematical modelling (Cuijpers 2004.)



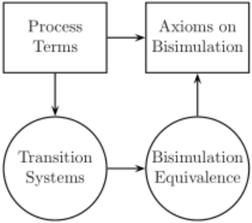


Table of contents

We will introduce the following models for concurrent systems:

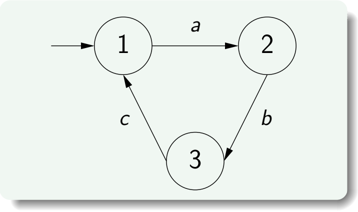
- Transition systems
- Models which are closer to realistic programming languages (for instance process calculi)
- Additional models: Timed automata, graph transformation systems

Furthermore (in order to investigate/analyze systems):

- Specification of properties of concurrent systems (Hennessy-Milner logics)
- Behavioural equivalences: When do two systems behave the same (from the point of view of an external observer)?

Transition systems

- Transition systems represent states and transitions between states.
- True parallelism is not directly represented.
- Strong similarity to automata, however we are here not so much interested in the accepted language.



Reviews of some notions

Definitions

- For a set X, we write X^* for the set of all finite words including the empty one ε.
- For a set X, we write X^ω the set of all infinite words. Also, we write $X^\infty = X^* \cup X^\omega$.
- A binary relation R between the sets X and Y is a subset of $X \times Y$, i.e., $R \subseteq X \times Y$.
- Often, we write xRy iff $(x, y) \in R$.
- A preorder $R \subseteq X \times Y$ is a

Reviews of some notions

Definitions

- For a set X, we write X^* for the set of all finite words including the empty one ε.
- For a set X, we write X^ω the set of all infinite words. Also, we write $X^\infty = X^* \cup X^\omega$.
- A binary relation R between the sets X and Y is a subset of $X \times Y$, i.e., $R \subseteq X \times Y$.
- Often, we write xRy iff $(x, y) \in R$.
- A preorder $R \subseteq X \times Y$ is a reflexive and transitive relation.

Reviews of some notions

Definitions

- For a set X, we write X^* for the set of all finite words including the empty one ε.
- For a set X, we write X^ω the set of all infinite words. Also, we write $X^\infty = X^* \cup X^\omega$.
- A binary relation R between the sets X and Y is a subset of $X \times Y$, i.e., $R \subseteq X \times Y$.
- Often, we write xRy iff $(x, y) \in R$.
- A preorder $R \subseteq X \times Y$ is a reflexive and transitive relation.
- A partial order (poset) R is a

Reviews of some notions

Definitions

- For a set X, we write X^* for the set of all finite words including the empty one ε.
- For a set X, we write X^ω the set of all infinite words. Also, we write $X^\infty = X^* \cup X^\omega$.
- A binary relation R between the sets X and Y is a subset of $X \times Y$, i.e., $R \subseteq X \times Y$.
- Often, we write xRy iff $(x, y) \in R$.
- A preorder $R \subseteq X \times Y$ is a reflexive and transitive relation.
- A partial order (poset) R is a preorder that is antisymmetric.

Reviews of some notions

Definitions

- For a set X, we write X^* for the set of all finite words including the empty one ε.
- For a set X, we write X^ω the set of all infinite words. Also, we write $X^\infty = X^* \cup X^\omega$.
- A binary relation R between the sets X and Y is a subset of $X \times Y$, i.e., $R \subseteq X \times Y$.
- Often, we write xRy iff $(x, y) \in R$.
- A preorder $R \subseteq X \times Y$ is a reflexive and transitive relation.
- A partial order (poset) R is a preorder that is antisymmetric.
- An equivalence relation R is a

Reviews of some notions

Definitions

- For a set X, we write X^* for the set of all finite words including the empty one ε.
- For a set X, we write X^ω the set of all infinite words. Also, we write $X^\infty = X^* \cup X^\omega$.
- A binary relation R between the sets X and Y is a subset of $X \times Y$, i.e., $R \subseteq X \times Y$.
- Often, we write xRy iff $(x, y) \in R$.
- A preorder $R \subseteq X \times Y$ is a reflexive and transitive relation.
- A partial order (poset) R is a preorder that is antisymmetric.
- An equivalence relation R is a partial order that is symmetric.

Transition system space

Formal definition

A transition system space is a triple $(S, L, \rightarrow)$ of

1. a set of states S;
2. a set of labels L;
3. a transition relation $\rightarrow \subseteq S \times L \times S$;

Notations:

- $s \xrightarrow{a} t \iff (s, a, t) \in \rightarrow$
- $s \stackrel{a}{\nrightarrow} \iff \nexists_{t \in S}\ s \xrightarrow{a} t$

Basics

Example on board.

Definition

The reachability relation $\rightarrow^* \subseteq S \times L^* \times S$ is inductively defined as follows:

$$\frac{}{s \overset{\varepsilon}{\twoheadrightarrow} s} \qquad \frac{s \overset{w}{\twoheadrightarrow} s' \quad s' \overset{a}{\rightarrow} s''}{s \overset{wa}{\twoheadrightarrow} s''}$$

The transition system induced by state s consists of all states reachable from s, and it has the transitions and final states induced by the transition system space.

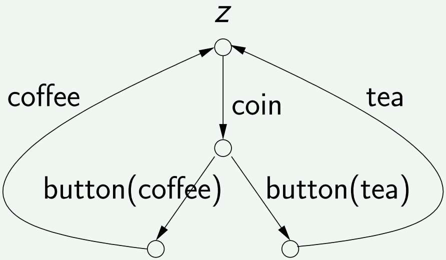
Transition systems (examples)

A classical example: the tea/coffee-machine

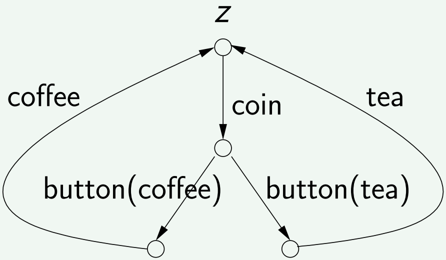
We want to model a very simple machine that

- outputs tea or coffee after a coin has been inserted and a button has been pressed,
- can show faulty behaviour *and*
- may potentially behave non-deterministically.

Transition systems (examples)

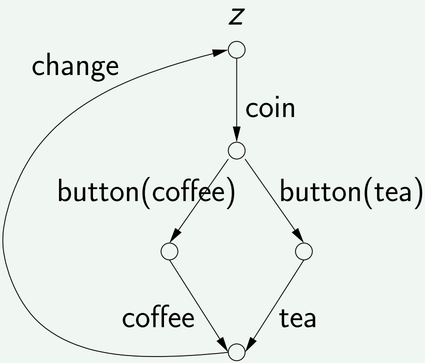


Transition systems (examples)

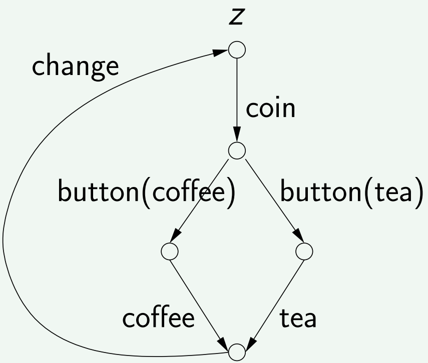


A tea/coffee-machine.

Transition systems (examples)

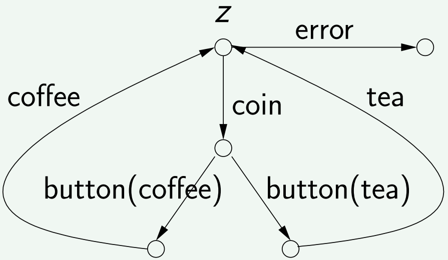


Transition systems (examples)

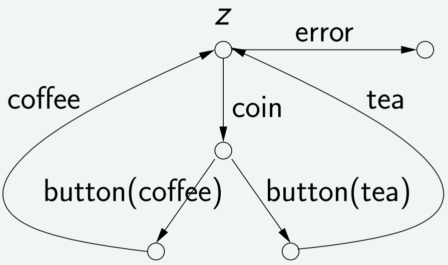


A machine that gives back change.

Transition systems (examples)

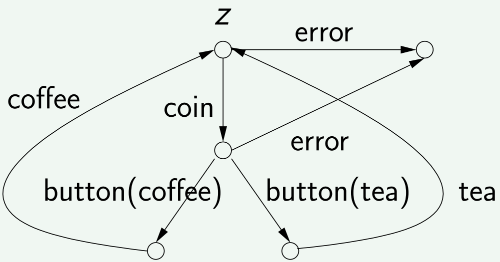


Transition systems (examples)

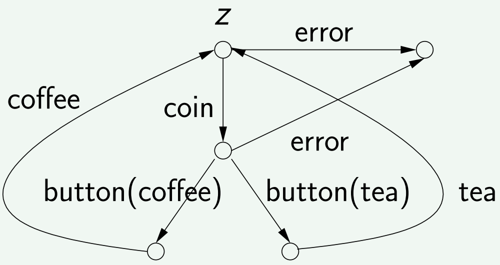


A machine with an error. The occurrence of an error is actually rather an internal action and could alternatively be modelled with a τ.

Transition systems (examples)

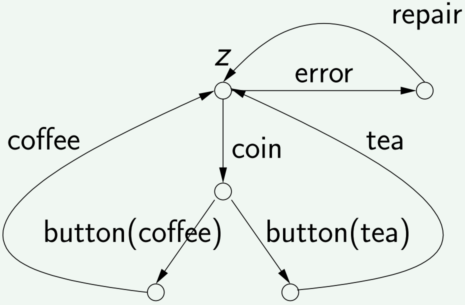


Transition systems (examples)

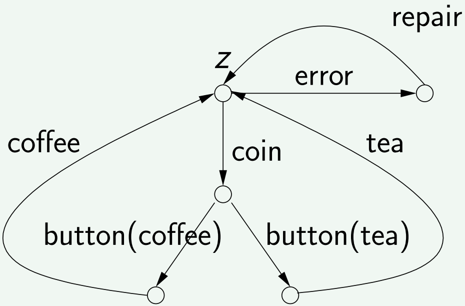


An (unfair) machine with faulty behaviour which may enter the error state after a coin has been inserted.

Transition systems (examples)

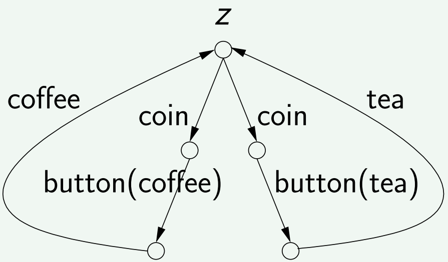


Transition systems (examples)

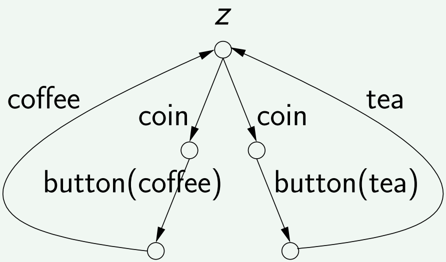


A machine with an error state that can be repaired.

Transition systems (examples)



Transition systems (examples)



A machine with non-deterministic behaviour that makes a choice of beverages for the user.

Deterministic transition systems

Deterministic transition system (definition)

A state s of a transition system is *deterministic*

$$\forall_{s',s''\in S,a\in L}\ (s \xrightarrow{a} s' \wedge s \xrightarrow{a} s'') \implies s' = s''$$

A transition system induced by state s is deterministic if every reachable states from s is deterministic.

Deterministic transition systems

Deterministic transition system (definition)

A state s of a transition system is *deterministic*

$$\forall_{s',s''\in S, a\in L}\ (s \xrightarrow{a} s' \wedge s \xrightarrow{a} s'') \implies s' = s''$$

A transition system induced by state s is deterministic if every reachable states from s is deterministic.

Remarks:

- All tea/coffee-machines, apart from the last, are deterministic.

Deterministic transition systems

Deterministic transition system (definition)

A state s of a transition system is *deterministic*

$$\forall_{s',s''\in S,a\in L}\ (s \xrightarrow{a} s' \wedge s \xrightarrow{a} s'') \implies s' = s''$$

A transition system induced by state s is deterministic if every reachable states from s is deterministic.

Remarks:

- All tea/coffee-machines, apart from the last, are deterministic.
- Opposed to deterministic finite automata we do not require for deterministic transition systems that every action is feasible in every state.

Some more definitions

- A state s of a transition system is a *deadlock* state iff

$$\forall_{a \in L} \nexists_t \; s \xrightarrow{a} t.$$

  A transition system starting from s has a deadlock iff a deadlock state is reachable from s.

- A transition system is *regular* iff both its set of states and transitions are finite.
- A transition system is *image finite* iff each of its states has only finitely many outgoing transitions.

Behavioural equivalences (bisimilarity)

Similar to the minimization procedure for (deterministic) finite automata, there exists a method for determining bisimilar pairs of states in a transition system.

Behavioural equivalences (bisimilarity)

Similar to the minimization procedure for (deterministic) finite automata, there exists a method for determining bisimilar pairs of states in a transition system.

Idea:

- Start with a very coarse relation $\sim_0$ that relates all possible states.
- Refine this relation step by step and construct relations $\sim_1$, $\sim_2$, ...
- As soon as two subsequent relations coincide ($\sim_n = \sim_{n+1}$) we have found the bisimilarity (at least for finite transition systems). That is, we have $\leftrightarroweq = \sim_n$.

Behavioural equivalences (bisimilarity)

Method for determining bisimilar pairs of states

Input: A transition system $T = (S, L, \rightarrow)$

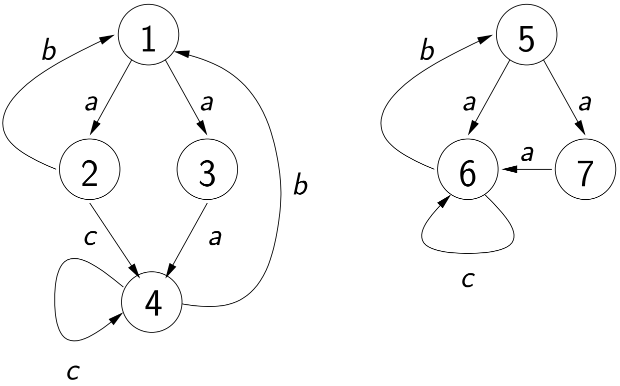
- Define $\sim_0 = S \times S$.
- $\sim_{n+1} \subseteq S \times S$, where $s \sim_{n+1} s'$ if and only if for all $a \in L$:
  1. For every t with $s \xrightarrow{a} t$ there exists t' such that $s' \xrightarrow{a} t'$ and $t \sim_n t'$.
  2. For every t' with $s' \xrightarrow{a} t'$ there exists t such that $s \xrightarrow{a} t$ and $t \sim_n t'$.

The method terminates as soon as $\sim_n = \sim_{n+1}$.

Output: $\sim_n$

Behavioural equivalences (bisimilarity)

Example: determine the bisimilar pairs of states of the following transition system



Behavioural equivalences (bisimilarity)

If we represent the equivalence relations $\sim_i$ via equivalence classes, then we obtain the following sequence $\sim_0, \sim_1, \sim_2 {=} \sim_3$.

$\sim_0$

1	3	2	6
5	7	4	

$\sim_1$

1 3		2 6	
5 7		4	

$\sim_2$ = $\sim_3$

1	3	2 6
5	7	4

1	3	2 6
5	7	4

Behavioural equivalences (bisimilarity)

Lemma

It holds that:

1. $\sim_n$ is an equivalence relation for all $n \in \mathbb{N}$.
2. $s \sim_n s'$ implies $s \sim_m s'$ for all $m \leq n$.
3. $s \leftrightarrow s'$ implies $s \sim_n s'$ for all $n \in \mathbb{N}$.
4. $\sim_n = \sim_{n+1}$ implies $\sim_n = \sim_m$ for all $m \geq n$.

Behavioural equivalences (bisimilarity)

Proposition

Let $T = (S, L, \rightarrow)$ be an image finite transition system space, i.e., for every state s the set

$$\{t \mid \exists a \in L \colon s \xrightarrow{a} t\}$$

is finite.

Then we have $s \leftrightarroweq t$ if and only if $s \sim_n t$ for all $n \in \mathbb{N}$.

In other words: $\leftrightarroweq = \bigcap_{n \in \mathbb{N}} \sim_n$.

Behavioural equivalences (bisimilarity)

Proposition

Let $T = (S, L, \rightarrow)$ be an image finite transition system space, i.e., for every state s the set

$$\{t \mid \exists a \in L \colon s \xrightarrow{a} t\}$$

is finite.

Then we have $s \leftrightarrow t$ if and only if $s \sim_n t$ for all $n \in \mathbb{N}$.

In other words: $\leftrightarrow = \bigcap_{n \in \mathbb{N}} \sim_n$.

This proposition does not hold for transition systems which are not finitely branching.

Recap on recursion

Definition

Given a signature Σ and a set of recursive variables V_R, a recursive equation is an equation of the form

$$X = t,$$

where $X \in V_R, t \in \mathcal{T}(\Sigma), \mathrm{var}(t) \subseteq V_R$, and $\mathrm{var}(t) \cap V_R = \emptyset$.
A recursive specification E over Σ and V_R is a set of recusrive specifications that contains precisely one recursive equation for each recursive variable.

Added the axiom (viewing recursive variables as constant):

$$(X = t)_{(X=t)\in E}.$$

Solutions of recursive specification

Given:

1. a process theory T with signature Σ,
2. a recursive specification E over Σ and V_R,
3. a model $\mathbb{M}$ (with domain M) of T with interpretation ι.

Then, an *extended* interpretation $\kappa : \Sigma \cup V_R \to M$ is a *solution* of E in $\mathbb{M}$ iff

$$\mathbb{M}, \kappa \models X = t \qquad \text{for every } (X = t) \in E.$$

Intuition

Associate a transition system to a recursive variable and verify that the transition systems of both L.H.S. and R.H.S. are bisimilar.

Solutions of recursive specification

Give how many solutions the following recursive specification has in the theory $\mathrm{BSP}(A)$ and $(\mathrm{BSP} + E)(A)$?

1. $E = \{X = a.1\}$.
2. $E = \{X = a.X\}$.
3. $E = \{X = X\}$.

Term model

Definition

Let Rec denote the set of all recursive specifications of interest and $V_R(E)$ denote the set of recursive variables in a recursive specification E.

Then, $\mathcal{C}(\mathrm{BSP}_{\mathrm{rec}})(A)$ is the set of all closed terms over the $BSP(A)$ signature extended with all recursion variables.

The term algebra $\mathbb{P}(\mathrm{BSP}_{\mathrm{rec}}(A))$ is defined as follows:

$$\Big(\mathcal{C}(\mathrm{BSP}_{\mathrm{rec}})(A), +, (a._)_{a\in A}, (\mu X.E)_{E\in \mathrm{Rec}, X\in V_R(E)}, 0, 1\Big).$$

Term model

$$\frac{(X = t) \in E \wedge \mu t.E \downarrow}{\mu X.E \downarrow} \qquad \frac{(X = t) \in E \wedge \mu t.E \xrightarrow{a} y}{\mu X.E \xrightarrow{a} y},$$

where

1. $\mu t.E = t$ if $t \in \{0, 1\}$,
2. $\mu(\mu X.E).E = \mu X.E$ (for any recursive variable $X \in V_R(E)$),
3. $\mu(a.t).E = a.(\mu t.E)$ (for any $a \in A, t \in \mathcal{C}((\mathrm{BSP} + E)(A))$),
4. $\mu(s + t).E = \mu s.E + \mu t.E$ (for any $s, t \in \mathcal{C}((\mathrm{BSP} + E)(A))$.

Theorem

Bisimilarity is a congruence on term algebra $\mathbb{P}(BSP_{rec}(A))$.

Expressiveness

Definition

A (computable) transition system is *expressible* in a process theory T if it is bisimilar to the transition system induced by a closed term in T.

A transition system is *countable* iff both the sets of states and transitions are countable.

A process is *image-finite* (*regular*) iff the equivalence class under bisimilarity contains at-least one image-finite (regular) transition system.

Expressiveness

Theorem

Every countable transition system is expressible in $BSP_{rec}(A)$.

Warning

This doesn't imply that every transition system induced by recursive specifications over BSP(A) is image finite (nor regular).

Example: Consider $E = \{X_n = X_{n+1} + a^n.1\}$ (for $n \geq 0$).

Definability

Another notion (however model-independent) of expressiveness.

Definition

Let T be a process theory *without recursion*. A process is (finitely) *definable* over T if and only if it is the unique solution of some designated recursive variable of a (finite) guarded recursive specification over the signature of T.

Theorem

A process is (finitely) definable over $BSP(A)$ *iff it is image-finite (regular).*

Stack process

Consider the following set of recursive equations over a finite set of data elements.

$$
\begin{aligned}
\mathsf{Stack} &= S_{\varepsilon}, \\
S_{\varepsilon} &= 1 + \sum_{d \in D} \mathsf{push}(d).S_d, \\
S_{d\sigma} &= \mathsf{pop}(d).S_{\sigma} + \sum_{e \in D} \mathsf{push}(e).S_{ed\sigma} \quad (\text{for every } d \in D, \sigma \in D^*)
\end{aligned}
$$

Stack process

Consider the following set of recursive equations over a finite set of data elements.

$$\begin{aligned} \text{Stack} &= S_\varepsilon, \\ S_\varepsilon &= 1 + \sum_{d \in D} \text{push}(d).S_d, \\ S_{d\sigma} &= \text{pop}(d).S_\sigma + \sum_{e \in D} \text{push}(e).S_{ed\sigma} \quad (\text{for every } d \in D, \sigma \in D^*) \end{aligned}$$

Is Stack a regular process?

Recap on BCP

$$\frac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y} \qquad \frac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y' \quad \gamma(a,b) \text{ is defined}}{x \parallel y \xrightarrow{\gamma(a,b)} x' \parallel y'}$$

$$\frac{x \xrightarrow{a} x'}{x \mathbin{⌊\!⌊} y \xrightarrow{a} x' \parallel y} \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y' \quad \gamma(a,b) \text{ is defined}}{x | y \xrightarrow{\gamma(a,b)} x' \parallel y'}$$

$$\frac{x\downarrow \quad y\downarrow}{x \parallel y\downarrow} \qquad \frac{x\downarrow \quad y\downarrow}{x|y\downarrow}$$

Recap on BCP

Key axiom

$$x \parallel y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x | y$$

Axioms for $\mathbin{\lfloor\!\lfloor}$

$$\begin{aligned} 0 \mathbin{\lfloor\!\lfloor} x &= 0 \\ 1 \mathbin{\lfloor\!\lfloor} x &= 0 \\ a.x \mathbin{\lfloor\!\lfloor} y &= a.(x \parallel y) \\ (x + y) \mathbin{\lfloor\!\lfloor} z &= x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z \end{aligned}$$

Recap on BCP

Key axiom

$$x \parallel y = x \llfloor y + y \llfloor x + x | y$$

Axioms for |

$$\begin{aligned} 0|x &= 0 \\ 1|1 &= 1 \\ a.x|1 &= 0 \\ a.x|b.y &= \begin{cases} \gamma(a,b).x \parallel y & \text{if} \gamma(a,b) \text{ is defined} \\ 0 & \text{otherwise} \end{cases} \\ (x+y)|z &= x|z + y|z \end{aligned}$$

How to enforce interaction

Encapsulation (restriction) operator

$$\frac{x \xrightarrow{a} x' \quad a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(x')} \qquad \frac{x\downarrow}{\partial_H(x)\downarrow}$$

Example

Let $\gamma(a, b) = c$. Then, $\partial_{\{a,b\}}(a.0 \parallel b.0) =$

How to enforce interaction

Encapsulation (restriction) operator

$$\frac{x \xrightarrow{a} x' \quad a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(x')} \qquad \frac{x\downarrow}{\partial_H(x)\downarrow}$$

Example

Let $\gamma(a, b) = c$. Then, $\partial_{\{a,b\}}(a.0 \parallel b.0) = c.(0 \parallel 0)$.

How to enforce interaction

Encapsulation (restriction) operator

$$\frac{x \xrightarrow{a} x' \quad a \notin H}{\partial_H(x) \xrightarrow{a} \partial_H(x')} \qquad \frac{x\downarrow}{\partial_H(x)\downarrow}$$

Example

Let $\gamma(a, b) = c$. Then, $\partial_{\{a,b\}}(a.0 \parallel b.0) = c.(0 \parallel 0)$.
Practical example: Construction of asynchronous processes

Axioms of standard concurrency

The following ones (except A1) are derivable for closed terms using the old axioms.

$$x|y = y|x \qquad (A1)$$
$$x \parallel 1 = x$$
$$1|x + 1 = 1$$
$$(x \parallel y) \parallel z = x \parallel (y \parallel z)$$
$$(x|y)|z = x|(y|z)$$
$$(x \lfloor\!\lfloor y) \lfloor\!\lfloor z = x \lfloor\!\lfloor (y \parallel z)$$
$$(x|y) \lfloor\!\lfloor z = x|(y \lfloor\!\lfloor z)$$

Term algebra and term model

Term algebra $\mathbb{P}(\mathrm{BCP}(A,\gamma))$

$$\left(\mathcal{C}(\mathrm{BCP}(A,\gamma)), 0, 1, (a.\_)_{a\in A}, +, (\partial_H(\_))_{H\subseteq A}, \|, \mathbin{⫫}, \mid\right).$$

Term model $\mathbb{P}(\mathrm{BCP}(A,\gamma))_{/\leftrightarrow}$.

Results

1. Bisimilarity is a congruence on term algebra $\mathbb{P}(\mathrm{BCP})(A,\gamma)$.
2. For every closed $\mathrm{BCP}(A,\gamma)$-term t, there is a closed $\mathrm{BSP}(A)$-term t' such that $\mathrm{BCP}(A,\gamma) \vdash p = q$.
3. $\mathrm{BCP}(A,\gamma)$ is a sound axiomatization of $\mathbb{P}(\mathrm{BCP}(A,\gamma))_{/\leftrightarrow}$.
4. $\mathrm{BCP}(A,\gamma)$ is a ground-complete axiomatization of $\mathbb{P}(\mathrm{BCP}(A,\gamma))_{/\leftrightarrow}$.

Expressivity of BCP

Consider the following specification of bag over $D = \{0, 1\}$.

$$
\begin{aligned}
B_{0,0} &= 1 + i?0.B_{1,0} + i?1.B_{0,1},\\
B_{0,m+1} &= o!1.B_{0,m} + i?0.B_{1,m+1} + i?1.B_{0,m+2},\\
B_{n+1,0} &= o!0.B_{n,0} + i?0.B_{+2,0} + i?1.B_{n+1,1},\\
B_{n+1,m+1} &= o!0.B_{n,m+1} + o!1.B_{n+1,m} + i?0.B_{n+2,m+1} + i?1.B_{n+1,m+2}.
\end{aligned}
$$

Theorem

The behaviour of the above bag is finitely definable in BCP$(A, \emptyset)$.

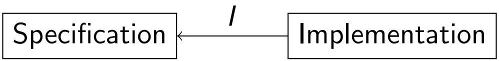
Proof.

See Proposition 7.6.4, where the following recursive equation:

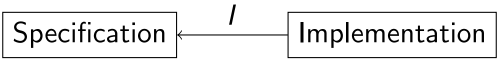
$$\mathrm{Bag} = 1 + i?0.(\mathrm{Bag} \parallel o!0.1) + i?1.(\mathrm{Bag} \parallel o!1.1)$$

is shown to have the same solution as $B_{0,0}$. □

The need of abstraction (τ-steps)



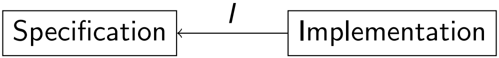
The need of abstraction (τ-steps)



Idea

To render implementation at the same abstraction level of the specification.

The need of abstraction (τ-steps)



Idea

To render implementation at the same abstraction level of the specification.

Formally,

$$\text{Specification} \leftrightarrow_b \tau_I(\text{Implementation})$$

Example:

1. Specification: One 2-place buffer
2. Implementation: Two 1-place buffers running in parallel.

$\mathrm{BSP}_\tau(A)$

The signature is extended with $\tau.x$.
The following *branching* axiom is added:

$$a.(\tau.(x+y)+x) = a.(x+y), \quad \text{where } a \in A_\tau.$$

$\mathrm{BSP}_\tau(A)$

The signature is extended with $\tau.x$.
The following *branching* axiom is added:

$$a.(\tau.(x+y)+x) = a.(x+y), \quad \text{where } a \in A_\tau.$$

Term algebra : $\mathbb{P}(\mathrm{BSP}_\tau)(A) = (\mathcal{C}(\mathrm{BSP}_\tau(A)), 0, 1, (a._)_{a\in A_\tau}, +)$

Results

1. $\leftrightarroweq_{rb}$ is a congruence on the term algebra $\mathbb{P}(\mathrm{BSP}_\tau)(A)$.
2. Soundness: The branching axiom is valid in the term model $\mathbb{P}(\mathrm{BSP}_\tau)(A)_{/\leftrightarroweq_{rb}}$.
3. Theory $\mathrm{BSP}_\tau(A)$ is a ground-complete axiomatisation of the term model $\mathbb{P}(\mathrm{BSP}_\tau)(A)_{/\leftrightarroweq_{rb}}$.

$\mathrm{TCP}_\tau(A)$

Signature consists of:

1. deadlock 0,
2. empty process 1,
3. Action prefix $a.\_$ $(a \in A_\tau)$,
4. Choice operator $+$,
5. Sequential composition $\cdot$,
6. Merge $\parallel$, left merge $⌊⌊$, and communication merge $|$,
7. Encapsulation operator $\partial_H(\_)$ $(H \subseteq A)$,
8. Hiding operator $\tau_I(\_)$ $(I \subseteq A)$.

Key axioms

Key axioms to be added with respect to the old ones.

$$
\begin{aligned}
a.(\tau.(x+y)+x) &= a.(x+y) & \\
x \lfloor\!\lfloor 0 &= x \cdot 0 & \\
x \lfloor\!\lfloor \tau.y &= x \lfloor\!\lfloor y & \\
x | \tau.y &= 0 & \\
\tau_I(\square) &= \square & \square \in \{0,1\} \\
\tau_I(a.x) &= a.\tau_I(x) & \text{if } a \notin I \\
\tau_I(a.x) &= \tau.\tau_I(x) & \text{if } a \in I \\
\tau_I(x+y) &= \tau_I(x) + \tau_I(y) &
\end{aligned}
$$

Today's focus

Modelling of alternating bit protocol using *mCRL2*.

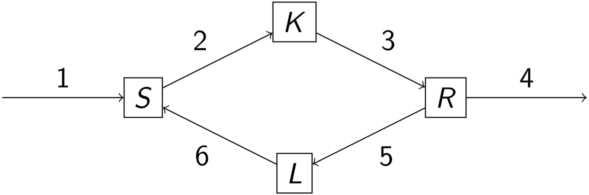
1. *mCRL2* is collection of tools aimed at formally analysing behaviour of distributed and concurrent systems.
2. extends process algebra ACP with various features data and time.
3. Downloadable from: `www.mcrl2.org`

My intention is not to introduce mCRL2, but rather to give a small demo on establishing the correctness of alternating bit protocol. (Different from the book where manual proof is given!)

Case-study: Alternating bit protocol (ABP)

Objective

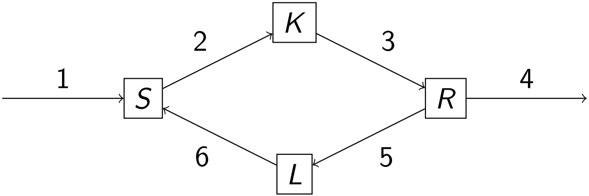
How to transmit data in a reliable way through an unreliable channel?



Correctness w.r.t. one-place buffer:

$$Buf_{14} = 1 + \sum_{d \in D} 1?d.4!d.Buf_{14}.$$

ABP contd.

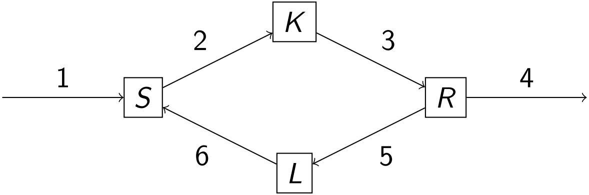


Protocol description:

Sender S

- Sender reads a datum d and passes on a sequence $d0, d0, \cdots$ to K until an acknowledgement 0 is received.
- Upon receiving correct acknowledgement, S reads the next datum e, appends 1, and send the sequence $e1, e1, \cdots$ to K until an acknowledgement 1 is received.

ABP contd.

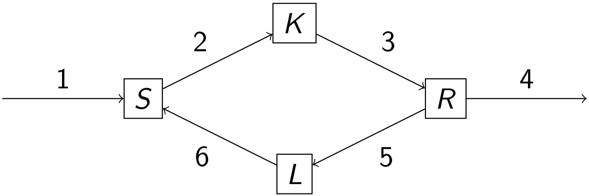


Protocol description:

Unreliable channel K

- Process K models transmission of data of the form ($d0$ or $d1$).
- However, K may also corrupt data.
- Assumption: Incorrect transmission of data d is recognized by a checksum.

ABP contd.

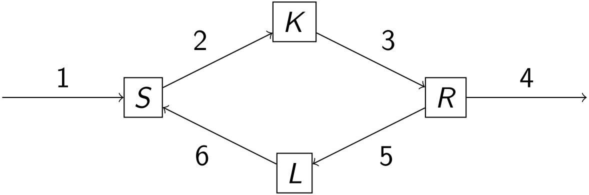


Protocol description:

Receiver R

- Receiver R receives data of the form $d0$ ($d1$).
- Later, R sends data d through port 4.

ABP contd.



Protocol description:

Unreliable channel L

- L models the transmission of acknowledgement from R to S.
- Like K, it may also corrupt the acknowledgement.

Recursion in $\mathrm{TCP}_\tau(A, \gamma)$

Does the operator $\tau.\_$ bodes well with guardedness?

Example

Consider the recursive equation $X = \tau.X$. How many distinct solutions the above equations have?

Recursion in $\mathrm{TCP}_\tau(A, \gamma)$

Does the operator $\tau.\_$ bodes well with guardedness?

Example

Consider the recursive equation $X = \tau.X$. How many distinct solutions the above equations have?

In particular, for any $\mathrm{TCP}_\tau(A, \gamma)$ closed term p, we have $\tau.p \leftrightarrow_{rb} \tau.\tau.p$.

Observation

Thus, the silent step should not be considered as a guard!

Recursion in $\mathrm{TCP}_\tau(A, \gamma)$

Direct abstraction from a guard

Consider the recursive specification $E = \{X = \tau_I(i.X)\}$ with $i \in I$.
How many distinct solutions the above equations have?

Recursion in $\mathrm{TCP}_\tau(A, \gamma)$

Direct abstraction from a guard

Consider the recursive specification $E = \{X = \tau_I(i.X)\}$ with $i \in I$. How many distinct solutions the above equations have?

For any $a, b \notin I$ and $a \neq b$, we have $[\tau.a.1]_{\leftrightarrow_{rb}}$ and $[\tau.b.1]_{\leftrightarrow_{rb}}$ are solutions of X.

Indirect abstraction from a guard

Consider the recursive specification $E = \{X = i.\tau_I(X)\}$ with $i \in I$. How many distinct solutions the above equations have?

Recursion in $\mathrm{TCP}_\tau(A, \gamma)$

Direct abstraction from a guard

Consider the recursive specification $E = \{X = \tau_I(i.X)\}$ with $i \in I$. How many distinct solutions the above equations have?

For any $a, b \notin I$ and $a \neq b$, we have $[\tau.a.1]_{\leftrightarrow_{rb}}$ and $[\tau.b.1]_{\leftrightarrow_{rb}}$ are solutions of X.

Indirect abstraction from a guard

Consider the recursive specification $E = \{X = i.\tau_I(X)\}$ with $i \in I$. How many distinct solutions the above equations have?

For any $a, b \notin I$ and $a \neq b$, we have $[i.a.1]_{\leftrightarrow_{rb}}$ and $[i.b.1]_{\leftrightarrow_{rb}}$ are solutions of X.

Guardedness in $\mathrm{TCP}_\tau(A, \gamma)$

Thus, necessary to restrict the notion of guardedness.

Modified definition

An occurrence of a variable x in a $\mathrm{TCP}_\tau(A, \gamma)$-term s is *guarded* iff the abstraction operator does not occur in s and x occurs in a subterm of the form $a.t$, for some action $a \in A$ and $\mathrm{TCP}_\tau(A, \gamma)$-term t.

Note that the restriction is on the occurrence of a variable, while the old definitions of when a term is (completely) guarded carries over without any change!

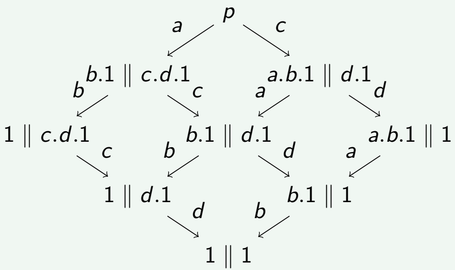
Result

Both the recursion principles RDP and RSP are valid in $\mathbb{P}(\mathrm{TCP}_\tau(A, \gamma))_{/\leftrightarrow_{rb}}$.

State explosion problem

Network of two processes

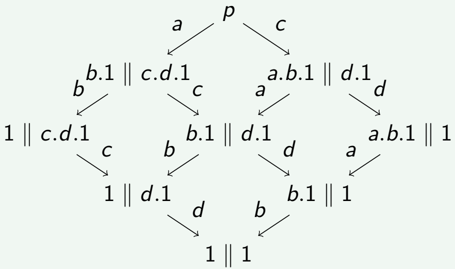
Assume $p \equiv a.b.1 \parallel c.d.1$ with $\gamma = \emptyset$. Then,



State explosion problem

Network of two processes

Assume $p \equiv a.b.1 \parallel c.d.1$ with $\gamma = \emptyset$. Then,



Trivia

In a network of n processes, if each of them have k states then how many number of states are generated?

Motivation

State explosion problem

In our previous example, the number of state grows exponentially with the number of communicating components.

Some ways to circumvent the size (i.e., the sum of the number of states and the number of transitions) of a transition system:

- Partial order reduction.
- Abstraction.
- Confluence and τ-prioritisation.

Motivation

In modelling, we are often solving this design equation:

$$\text{Spec} \leftrightarrow_b \tau_I(\partial_H(Imp))$$

Some transitions of the 'big' transition system are made invisible!

Key observations

1. Some observable transitions are removed in favour of confluent τ-steps while preserving branching bisimilarity.
2. All confluent τ-steps are inert.

Material: Section 11.1 and 11.2 from the book "Modelling and analysis of communicating systems" by Groote and Mousavi.

τ-Confluence

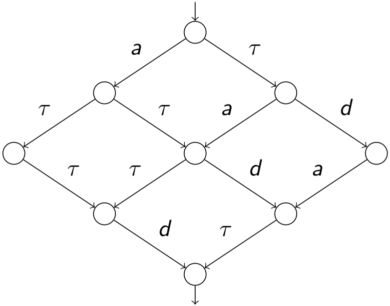
Definition

Let $(S, A_\tau, \to, \downarrow)$ be a transition system space and let $\to_\tau = \{(s, \tau, t) \mid s, t \in S \wedge s \xrightarrow{\tau} t\}$. A set $U \subseteq \to_\tau$ is called *τ-confluent* if for all transitions $s_1 \xrightarrow{a} s_2$ and $(s_1, \tau, s_3) \in U$ we have

- Either $(s_2, \tau, s_4) \in U$ and $s_3 \xrightarrow{a} s_4$, for some $s_4 \in S$.
- Or $a = \tau$ and $s_2 = s_3$.

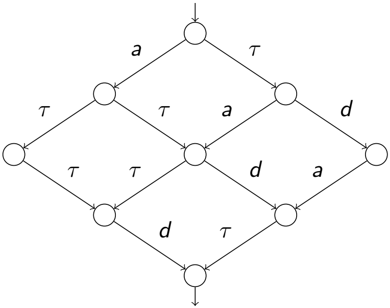
Note that the union of confluent sets of τ-steps is again confluent, so there is a maximal confluent set of τ-transitions.

τ-confluence



Which ones are confluent?

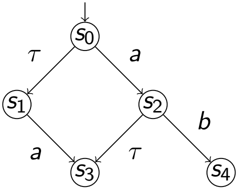
τ-confluence



Which ones are confluent?

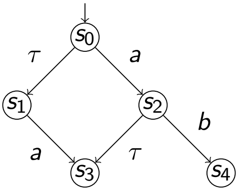
All of them!

τ-confluence



Which ones are confluent?

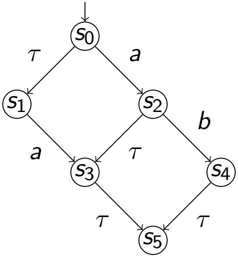
τ-confluence



Which ones are confluent?

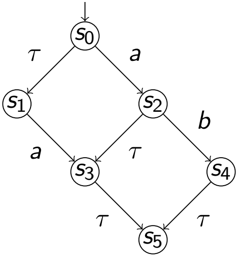
None of them!

τ-confluence



Which ones are confluent?

τ-confluence



Which ones are confluent?

(s_3, τ, s_5) and (s_4, τ, s_5).

τ-Prioritisation

Theorem

All τ-confluent steps are inert modulo $\leftrightarrow_b$.

τ-prioritisation: confluent τ-steps can be taken over other actions.

τ-Prioritisation

Theorem

All τ-confluent steps are inert modulo $\leftrightarrow_b$.

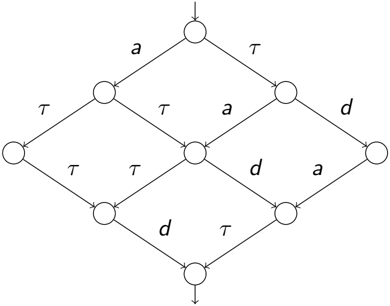
τ-prioritisation: confluent τ-steps can be taken over other actions.

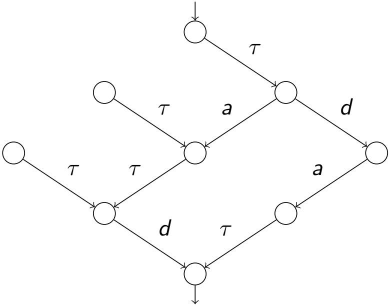
Formally

A transition system $(S, A_\tau, \Rightarrow, s_0, \downarrow)$ is a τ-*priortisation* of a given transition system $(S, A_\tau, \rightarrow, s_0, \downarrow)$ w.r.t. $U \subseteq \rightarrow_\tau$ iff

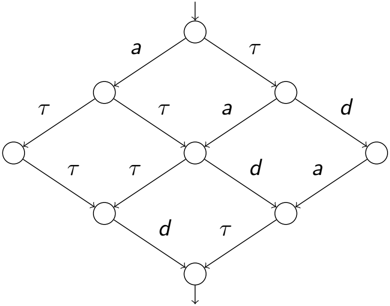
1. $\Rightarrow \subseteq \rightarrow$;
2. $\forall_{s,s' \in S}\ s \xrightarrow{a} s' \implies \left(s \overset{a}{\Rightarrow} s' \vee \exists_{s''}\ s \overset{\tau}{\Rightarrow} s'' \in U\right)$.

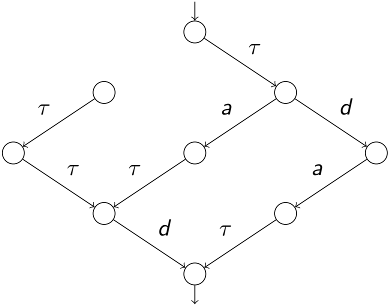
Example



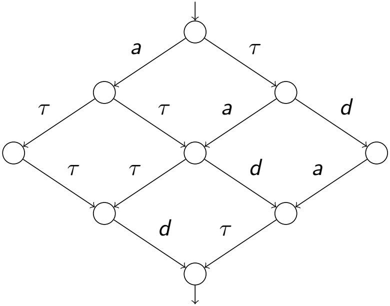


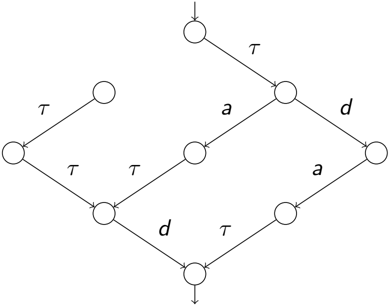
Example





Example





However, reachable state space in both the cases are branching bisimilar to $\tau.(a.1 \parallel d.1)$ with $\gamma = \emptyset$.

Application

τ-prioritisation reduces state space and for divergent free transition systems it preserves branching bisimulation.

Theorem

Given the following data:

1. *a transition system* $(S, A_\tau, \rightarrow, s_0, \downarrow)$,
2. *a* τ*-confluent set of invisible steps* $U \subseteq \rightarrow_\tau$,
3. *a transition system* $(S, A_\tau, \Rightarrow, s_0, \downarrow)$ *which is a* τ*-priortisation of the given transition system w.r.t.* U *and divergent free.*

Then, the two transition systems are branching bisimilar.

Application

τ-prioritisation reduces state space and for divergent free transition systems it preserves branching bisimulation.

Theorem

Given the following data:

1. *a transition system* $(S, A_\tau, \rightarrow, s_0, \downarrow)$,
2. *a* τ*-confluent set of invisible steps* $U \subseteq \rightarrow_\tau$,
3. *a transition system* $(S, A_\tau, \Rightarrow, s_0, \downarrow)$ *which is a* τ*-priortisation of the given transition system w.r.t.* U *and divergent free.*

Then, the two transition systems are branching bisimilar.

Divergent free is essential: $X = \tau.X + a.X$ is not branching bisimilar to $Y = \tau.Y$ after the τ-priortisation.

Quantitative verification

- Need to verify nonfunctional properties.
- Functional property:

Eventually, the message m will be received.

- Nonfunctional property:

The message m will be received with probability 0.98.
Other variations: involve time and even differential equations.

Quantitative verification

- Need to verify nonfunctional properties.
- Functional property:

Eventually, the message m will be received.

- Nonfunctional property:

The message m will be received with probability 0.98.
Other variations: involve time and even differential equations.

- This lecture: modelling uncertainties.

Basic probability theory: in brief

Restrict to classical definition of probability

$$P(A) = \frac{|A|}{|\Omega|}, \quad \text{where}$$

A is the finite set of favourable outcomes and Ω is the finite set of total outcomes.

[1] http://people.cas.uab.edu/~mosya/teaching/485-HW.pdf

Basic probability theory: in brief

Restrict to classical definition of probability

$$P(A) = \frac{|A|}{|\Omega|}, \quad \text{where}$$

A is the finite set of favourable outcomes and Ω is the finite set of total outcomes.

Refresher[1]

Two persons A and B are playing a game of tossing a fair coin alternatively. The first one to get a 'heads' wins the game. Suppose A starts the game, then what is the probability that A wins the game?

[1] http://people.cas.uab.edu/~mosya/teaching/485-HW.pdf

Back to modelling uncertainties

Consider the following one place buffer specification over a unitary domain $\{d\}$ with input port i and output port o.

$$B = 1 + i?d.o!d.B$$

How to model that the data d may be lost and when it's lost then nothing can be taken out or put into B?

Back to modelling uncertainties

Consider the following one place buffer specification over a unitary domain $\{d\}$ with input port i and output port o.

$$B = 1 + i?d.o!d.B$$

How to model that the data d may be lost and when it's lost then nothing can be taken out or put into B?

$$B = 1 + i?d.(o!d.B + \tau.0)$$

Refine: d may be lost with the probability 0.01.

Back to modelling uncertainties

Probabilistic choice $\oplus_p$ ($p \in (0,1)$)

Intuition

$$x \oplus_p y$$

Either x with probability p, or y with probability

Back to modelling uncertainties

Probabilistic choice $\oplus_p$ ($p \in (0,1)$)

Intuition

$$x \oplus_p y$$

Either x with probability p, or y with probability $1 - p$.

Difference between $x + y$ and $x \oplus_p y$.

- $+$: cannot be determined by an experiment.
- $\oplus_p$: can be determined by an experiment. (Law of large numbers!)

BSPprb(A)

Signature$=\{0, 1, +, (a._)_{a \in A}\} \cup \{(\oplus_p)_{p \in (0,1)}\}$

BSPprb(A)

Signature=$\{0, 1, +, (a.\_)_{a \in A}\} \cup \{(\oplus_p)_{p \in (0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x$

BSPprb(A)

Signature= $\{0, 1, +, (a.\_)_{a \in A}\} \cup \{(\oplus_p)_{p \in (0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x$ ✓

BSPprb(A)

Signature= $\{0, 1, +, (a.\_)_{a \in A}\} \cup \{(\oplus_p)_{p \in (0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x$ ✓
2. $(x + y) + z = x + (y + z)$

BSPprb(A)

Signature=$\{0, 1, +, (a.\_)_{a\in A}\} \cup \{(\oplus_p)_{p\in(0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x$ ✓
2. $(x + y) + z = x + (y + z)$ ✓

BSPprb(A)

Signature$=\{0, 1, +, (a._)_{a\in A}\} \cup \{(\oplus_p)_{p\in(0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x$ ✓
2. $(x + y) + z = x + (y + z)$ ✓
3. $x + x = x$

BSPprb(A)

Signature=$\{0, 1, +, (a.\_)_{a \in A}\} \cup \{(\oplus_p)_{p \in (0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x$ ✓
2. $(x + y) + z = x + (y + z)$ ✓
3. $x + x \neq x$

Intuition

Consider x to be tossing a fair coin.
Now $x + x$ means tossing two fair coins at the same time, whereas x means only tossing one. Even the sample space between the two experiments are different!

BSPprb(A)

Signature= $\{0, 1, +, (a._)_{a\in A}\} \cup \{(\oplus_p)_{p\in(0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x$ ✓
2. $(x + y) + z = x + (y + z)$ ✓
3. $x + x \neq x$
4. $a.x + a.x = a.x$ ✓

BSPprb(A)

Signature=$\{0, 1, +, (a.\_)_{a \in A}\} \cup \{(\oplus_p)_{p \in (0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x \quad \checkmark$
2. $(x + y) + z = x + (y + z) \quad \checkmark$
3. $x + x \neq x$
4. $a.x + a.x = a.x \quad \checkmark$
5. $1 + 1 = 1 \quad \checkmark$

BSPprb(A)

Signature=$\{0, 1, +, (a.\_)_{a \in A}\} \cup \{(\oplus_p)_{p \in (0,1)}\}$

What about the axioms?

(Note:The variables can be terms with probabilistic choice.)

1. $x + y = y + x$ ✓
2. $(x + y) + z = x + (y + z)$ ✓
3. $x + x \neq x$
4. $a.x + a.x = a.x$ ✓
5. $1 + 1 = 1$ ✓
6. $x + 0 = x$ ✓

BSPprb(A)

Still remains to consider the axioms of $\oplus_p$.

1. $x \oplus_p y = y \oplus_q x \quad q =$

BSPprb(A)

Still remains to consider the axioms of $\oplus_p$.

1. $x \oplus_p y = y \oplus_q x \quad q = 1 - p.$

BSPprb(A)

Still remains to consider the axioms of $\oplus_p$.

1. $x \oplus_p y = y \oplus_q x \quad q = 1 - p.$
2. $x \oplus_p (y \oplus_r z) = (x \oplus_{p'} y) \oplus_{r'} z$

BSPprb(A)

Still remains to consider the axioms of $\oplus_p$.

1. $x \oplus_p y = y \oplus_q x \quad q = 1 - p.$
2. $x \oplus_p (y \oplus_r z) = (x \oplus_{p'} y) \oplus_{r'} z$

$$p' = \frac{p}{p + r - pr}$$
$$r' = p + r - pr$$

BSPprb(A)

Still remains to consider the axioms of $\oplus_p$.

1. $x \oplus_p y = y \oplus_q x \quad q = 1 - p.$
2. $x \oplus_p (y \oplus_r z) = (x \oplus_{p'} y) \oplus_{r'} z$

$$p' = \frac{p}{p + r - pr}$$
$$r' = p + r - pr$$

3. $x \oplus_p x =$

BSPprb(A)

Still remains to consider the axioms of $\oplus_p$.

1. $x \oplus_p y = y \oplus_q x \quad q = 1 - p.$
2. $x \oplus_p (y \oplus_r z) = (x \oplus_{p'} y) \oplus_{r'} z$

$$p' = \frac{p}{p + r - pr}$$
$$r' = p + r - pr$$

3. $x \oplus_p x = x$

BSPprb(A)

Still remains to consider the axioms of $\oplus_p$.

1. $x \oplus_p y = y \oplus_q x \quad q = 1 - p.$
2. $x \oplus_p (y \oplus_r z) = (x \oplus_{p'} y) \oplus_{r'} z$

$$p' = \frac{p}{p + r - pr}$$
$$r' = p + r - pr$$

3. $x \oplus_p x = x$
4. $(x \oplus_p y) + z =$

BSPprb(A)

Still remains to consider the axioms of $\oplus_p$.

1. $x \oplus_p y = y \oplus_q x \quad q = 1 - p.$
2. $x \oplus_p (y \oplus_r z) = (x \oplus_{p'} y) \oplus_{r'} z$

$$p' = \frac{p}{p + r - pr}$$
$$r' = p + r - pr$$

3. $x \oplus_p x = x$
4. $(x \oplus_p y) + z = (x + z) \oplus_p (y + z)$

BSPprb(A)

Can we see equationally $x + x \neq x$?

On blackboard.

Towards the term model of $\mathrm{BSPprb}(\mathrm{A})$

Identification

1. Two types of states (closed terms): probabilistic terms and dynamic terms.
2. Two types of transitions: action transition relation $\rightarrow$ and probabilistic transition relation $\rightsquigarrow$ with distribution function μ.

Towards the term model of BSPprb(A)

Identification

1. Two types of states (closed terms): probabilistic terms and dynamic terms.
2. Two types of transitions: action transition relation $\rightarrow$ and probabilistic transition relation $\rightsquigarrow$ with distribution function μ.

Probabilistic terms

Terms that have a probabilistic choice as the main operator (e.g., $a.1 \oplus_p b.1$, $a.1 \oplus_p (b.1 + c.1)$) or there are an alternative composition of sub-terms of which at least one has a probabilistic choice as the main opeartor (e.g., $a.1 + (b.1 \oplus c.0)$).

Dynamic terms

Terms of the form $\sum_{i<n} a_i.q_i$ or $\sum_{i<n} a_i.q_i + 1$.

Towards the term model of $\mathrm{BSPprb}(A)$

Identification

1. Two types of states (closed terms): probabilistic terms and dynamic terms.
2. Two types of transitions: action transition relation $\rightarrow$ and probabilistic transition relation $\rightsquigarrow$ with distribution function μ.

Idea

1. Only probabilistic terms can fire probabilistic transitions $\rightsquigarrow$.
2. Target state of any probabilistic transition is a dynamic term if $x \rightsquigarrow x'$ then x' is a dynamic term.
3. To resolve nondeterminism first resolve probabilistic choice (if present).

   E.g., in the term $(a.1 \oplus_p b.1) + c.0$, to fire c one has to resolve a and b probabilistically.

Operational rules of BSPprb(A)

$$a.x \xrightarrow{a} x \qquad 1\downarrow$$

Operational rules of BSPprb(A)

$$a.x \xrightarrow{a} x \qquad 1\downarrow$$

$$\frac{x \rightsquigarrow x'}{x \oplus_p y \rightsquigarrow x'} \qquad \frac{y \rightsquigarrow y'}{x \oplus_p y \rightsquigarrow y'} \qquad \frac{x \not\rightsquigarrow}{x \oplus_p y \rightsquigarrow x} \qquad \frac{y \not\rightsquigarrow}{x \oplus_p y \rightsquigarrow y}$$

Operational rules of $\mathrm{BSPprb}(\mathrm{A})$

$$a.x \xrightarrow{a} x \qquad 1\downarrow$$

$$\frac{x \rightsquigarrow x'}{x \oplus_p y \rightsquigarrow x'} \qquad \frac{y \rightsquigarrow y'}{x \oplus_p y \rightsquigarrow y'} \qquad \frac{x \not\rightsquigarrow}{x \oplus_p y \rightsquigarrow x} \qquad \frac{y \not\rightsquigarrow}{x \oplus_p y \rightsquigarrow y}$$

$$\frac{x \xrightarrow{a} x' \quad y \not\rightsquigarrow}{x + y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y' \quad x \not\rightsquigarrow}{x + y \xrightarrow{a} y'} \qquad \frac{x\downarrow \quad y \not\rightsquigarrow}{(x + y)\downarrow} \qquad \frac{y\downarrow \quad x \not\rightsquigarrow}{(x + y)\downarrow}$$

Operational rules of BSPprb(A)

$$a.x \xrightarrow{a} x \qquad 1\downarrow$$

$$\frac{x \rightsquigarrow x'}{x \oplus_p y \rightsquigarrow x'} \qquad \frac{y \rightsquigarrow y'}{x \oplus_p y \rightsquigarrow y'} \qquad \frac{x \not\rightsquigarrow}{x \oplus_p y \rightsquigarrow x} \qquad \frac{y \not\rightsquigarrow}{x \oplus_p y \rightsquigarrow y}$$

$$\frac{x \xrightarrow{a} x' \quad y \not\rightsquigarrow}{x + y \xrightarrow{a} x'} \qquad \frac{y \xrightarrow{a} y' \quad x \not\rightsquigarrow}{x + y \xrightarrow{a} y'} \qquad \frac{x\downarrow \quad y \not\rightsquigarrow}{(x + y)\downarrow} \qquad \frac{y\downarrow \quad x \not\rightsquigarrow}{(x + y)\downarrow}$$

$$\frac{x \rightsquigarrow x' \quad y \rightsquigarrow y'}{x + y \rightsquigarrow x' + y'} \qquad \frac{x \rightsquigarrow x' \quad y \not\rightsquigarrow}{x + y \rightsquigarrow x' + y} \qquad \frac{x \not\rightsquigarrow \quad y \rightsquigarrow y'}{x + y \rightsquigarrow x + y'}$$

Example

Draw transition systems of the following term

1. $a.1 \oplus_{\frac{1}{2}} b.0$
2. $(a.1 \oplus_{\frac{1}{3}} b.1) + c.0$

Recovering probabilities

What is the probability of performing a in t, t'?

$$t \equiv a.1 \oplus_{\frac{1}{2}} b.1 \quad t' \equiv a.1 \oplus_{\frac{1}{2}} a.1$$

Recovering probabilities

What is the probability of performing a in t, t'?

$$t \equiv a.1 \oplus_{\frac{1}{2}} b.1 \quad t' \equiv a.1 \oplus_{\frac{1}{2}} a.1$$

Such information is captured formally by probability distribution function $\mu : \mathcal{C}(\mathrm{BSPprb(A)}) \times \mathcal{C}(\mathrm{BSPprb(A)}) \to [0, 1]$:

- $\mu(a.q, a.q) = 1$
- $\mu(0, 0) = 1$
- $\mu(1, 1) = 1$
- $\mu(q + r, q' + r') = \mu(q, q') \cdot \mu(q', r')$
- $\mu(q \oplus_p r, s) = p \cdot \mu(q, s) + (1 - p) \cdot \mu(r, s)$, and
- $\mu(q, r) = 0$, in all other cases.

Probabilistic bisimulation

Definition

An equivalence relation R on $\mathcal{C}(\mathrm{BSPprb}(\mathrm{A}))$ is a probabilistic bisimulation iff for all closed terms p, p', q, q' we have

- if $p \xrightarrow{a} p'$ and pRq then $\exists_{q'}\ q \xrightarrow{a} q' \wedge p'Rq'$,
- if pRq and $p\downarrow$, then $q\downarrow$,
- if $p \rightsquigarrow p'$ and pRq, then either
  - $\exists_{q'}\ q \rightsquigarrow q' \wedge p'Rq' \wedge \mu(p, [p']_R) = \mu(q, [q']_R)$, or
  - $p'Rq$ and $\mu(p, [p']_R) = 1$.

Two terms p, q are bisimilar, denoted $p \leftrightarrow q$, iff there is a probabilistic bisimulation relation R such that pRq.

E.g., $(1 \oplus_{\frac{1}{2}} 0) + (1 \oplus_{\frac{1}{2}} 0) \leftrightarrow 1 \oplus_{\frac{3}{4}} 0$,
$a.b.1 \oplus_{\frac{2}{3}} a.1 \leftrightarrow a.b.1 \oplus_{\frac{1}{3}} (a.b.1 \oplus_{\frac{1}{2}} a.1)$.