2. Transition Systems

José Proença

System Verification (CC4084) 2024/2025

CISTER – U.Porto, Porto, Portugal

https://fm-dcc.github.io/sv2425

Syllabus

- Introduction to model-checking
- CCS: a simple language for concurrency
  - Syntax
  - Semantics
  - Equivalence
  - mCRL2: modelling
- Dynamic logic
  - Syntax
  - Semantics
  - Relation with equivalence
  - mCRL2: verification
- Timed Automata
  - Syntax
  - Semantics (composition, Zeno)
  - Equivalence
  - UPPAAL: modelling
- Temporal logics (LTL/CTL)
  - Syntax
  - Semantics
  - UPPAAL: verification
- Probabilistic and stochastic systems
  - Going probabilistic
  - UPPAAL: monte-carlo

Why transition systems?

A Sprinkle of Linguistics

During the module we will encounter two linguistic concepts that every programmer should know:

- syntax - the rules used for determining whether a sentence is valid (in a language) or not
- semantics - the meaning of valid sentences

Ex. 2.1: Syntax

The sentence/program $\mathtt{x} := \mathtt{p}\,;\mathtt{q}$ is forbidden by the syntactic rules of most programming languages

Ex. 2.2: Semantics

The sentence/program $\mathtt{x} := 1$ has the meaning "writes 1 in the memory address corresponding to $\mathtt{x}$"

The need for Semantics in Formal Analysis

How can one prove that a program does what is supposed to do if its semantics (i.e. its meaning) is not established *a priori* ?

Ex. 2.3:

What is the end result of running $x := 2\ ; (x := x + 1 \parallel x := 0)$?

parallelism operator

Ex. 2.4: Value of y?

$int\ x = 0\ ;\ \ int\ f()\{x{+}{+}; return\ x; \}\ \ int\ g()\{x{-}{-}; return\ x; \}\ \ int\ y = f() + g();$

Widely used programming languages still lacks a formal semantics

Defining Transition System with Functors

Preliminaries pt. I

Definition (Functor)

A functor F sends a set X into a new set FX and a function $f : X \to Y$ into a new function $Ff : FX \to FY$ such that

$$F(\mathrm{id}) = \mathrm{id} \qquad F(g \cdot f) = Fg \cdot Ff$$

Fix a set A. The following two functors then naturally arise

- product - $X \mapsto A \times X, \;\; f \mapsto id \times f$
- exponential - $X \mapsto X^A, \;\; f \mapsto (g \mapsto f \cdot g)$

Preliminaries pt. II - the List and Powerset functors

The list functor - $[X] \mapsto X^*, \;\; [f] \mapsto \mathtt{map}\ f$

applies f to every element of a given list

The powerset functor - almost like the list functor; the difference is that we do not look at the order in which elements appear and how many times they repeat. Formally,

$$\mathtt{P}(X) \mapsto \;?\qquad\qquad ,\qquad \mathtt{P}(f) \mapsto \;?$$

Preliminaries pt. II - the List and Powerset functors

The list functor - $[X] \mapsto X^*$, $[f] \mapsto \texttt{map}\ f$

applies f to every element of a given list

The powerset functor - almost like the list functor; the difference is that we do not look at the order in which elements appear and how many times they repeat. Formally,

$$\mathtt{P}(X) \mapsto \{A \mid A \subseteq X\}, \qquad \mathtt{P}(f) \mapsto \ ?$$

Preliminaries pt. II - the List and Powerset functors

The list functor - $[X] \mapsto X^*, \; [f] \mapsto \texttt{map}\ f$

applies f to every element of a given list

The powerset functor - almost like the list functor; the difference is that we do not look at the order in which elements appear and how many times they repeat. Formally,

$$\mathtt{P}(X) \mapsto \{A \mid A \subseteq X\}, \qquad \mathtt{P}(f) \mapsto (A \mapsto \{f(a) \mid a \in A\})$$

The list functor - $[X] \mapsto X^*$, $[f] \mapsto \mathtt{map}\ f$

applies f to every element of a given list

The powerset functor - almost like the list functor; the difference is that we do not look at the order in which elements appear and how many times they repeat. Formally,

$$\mathtt{P}(X) \mapsto \{A \mid A \subseteq X\}, \qquad \mathtt{P}(f) \mapsto (A \mapsto \{f(a) \mid a \in A\})$$

Ex. 2.5: Powerset on Booleans

$\mathtt{P}(\mathtt{Bool}) \mapsto$

$\mathtt{P}(\mathtt{not}) \mapsto$

The list functor - $[X] \mapsto X^*$, $[f] \mapsto \mathtt{map}\ f$

applies f to every element of a given list

The powerset functor - almost like the list functor; the difference is that we do not look at the order in which elements appear and how many times they repeat. Formally,

$$\mathtt{P}(X) \mapsto \{A \mid A \subseteq X\}, \qquad \mathtt{P}(f) \mapsto (A \mapsto \{f(a) \mid a \in A\})$$

Ex. 2.5: Powerset on Booleans

$\mathtt{P}(\mathtt{Bool}) \mapsto \{\emptyset, \{\top\}, \{\bot\}, \{\top, \bot\}\}$

$\mathtt{P}(\mathtt{not}) \mapsto \mathit{Bools} \mapsto \{\mathit{not}(b) \mid b \in \mathit{Bools}\}$

A (Generalised) Notion of a Transition System

Definition (Transition system)

Let F be a functor. An F-transition system is a map $X \to FX$

Some famous examples of F-transition systems

- Moore machine - $X \to N \times X$
- Deterministic automata - $X \to \texttt{Bool} \times X^N$
- Non-deterministic automata - $X \to \texttt{Bool} \times \mathrm{P}(X)^N$
- Markov chain - $X \to \mathrm{D}(X)$

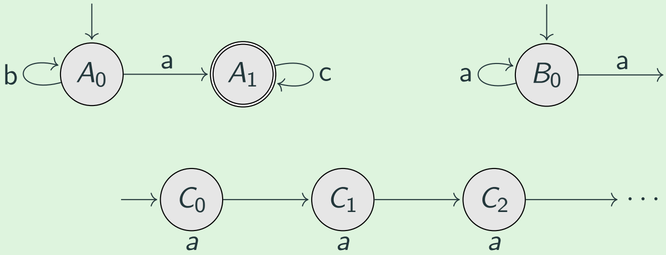
Powerset functor

Distribution functor

Recall functors

$X \mapsto A \times X \qquad X \mapsto \mathrm{P}(X) \qquad X \mapsto X^A \qquad X \mapsto \mathrm{D}(X)$

Ex. 2.6: Formalise as an F-transition system



Our First encounter with Coalgebra

Indeed the idea of working at the level of

Functors as Transition Types

is a very fruitful one; and which we only barely grasped —

in essence, it provides a universal theory of transition systems that can be instantiated to most kinds of transition system we will encounter in our life

CCS Process algebra

Sequential CCS - Syntax

$$\mathcal{P} \ni P, Q ::= K \mid \alpha.P \mid P + Q \mid \mathbf{0} \mid P[f] \mid P \backslash L \mid P|Q$$

where

- $\alpha \in N \cup \{\tau\}$ is an action
- K s a collection of process names or process constants
- $L \subseteq N$ is a set of labels
- f is a function that renames actions s.t. $f(\tau) = \tau$
- notation:

  $[f] = [a_1 \mapsto b_1, \ldots, a_n \mapsto b_n]$

Syntax

$$\mathcal{P} \ni P, Q ::= K \mid \alpha.P \mid P + Q \mid \mathbf{0} \mid P[f] \mid P\backslash L \mid P|Q$$

Ex. 2.7: Which are NOT syntactically correct? Why?

$$a.b.A + B \quad (1)$$
$$(a.\mathbf{0} + b.A)\backslash\{a, b, c\} \quad (2)$$
$$(a.\mathbf{0} + b.A)\backslash\{a, \tau\} \quad (3)$$
$$a.B + [b \mapsto a] \quad (4)$$
$$\tau.\tau.B + \mathbf{0} \quad (5)$$
$$a.(a + b).A \quad (6)$$
$$(a.B + b.B)[a \mapsto a, \tau \mapsto b] \quad (7)$$
$$(a.B + \tau.B)[b \mapsto a, a \mapsto a] \quad (8)$$
$$(a.b.A + b.\mathbf{0}).B \quad (9)$$
$$(a.b.A + b.\mathbf{0}) + B \quad (10)$$

CCS semantics - building a transition system

Every P yields a transition system $X \rightarrow ???$ with transitions prescribed by the rules below.

$$\text{(act)}\ \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \text{(sum-1)}\ \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'} \qquad \text{(sum-2)}\ \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 + P_2 \xrightarrow{\alpha} P_2'}$$

$$\text{(res)}\ \frac{P \xrightarrow{\alpha} P'}{P \backslash L \xrightarrow{\alpha} P' \backslash L}\ \alpha \notin L \qquad \text{(rel)}\ \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

- Initial states: the process being translated
- Final states: all states are final
- Language: possible sequences of actions of a process

CCS semantics - building a transition system

Every P yields a transition system $X \to$??? with transitions prescribed by the rules below.

$$\text{(act)}\ \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \text{(sum-1)}\ \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'} \qquad \text{(sum-2)}\ \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 + P_2 \xrightarrow{\alpha} P_2'}$$

$$\text{(res)}\ \frac{P \xrightarrow{\alpha} P'}{P\backslash L \xrightarrow{\alpha} P'\backslash L}\ \alpha \notin L \qquad \text{(rel)}\ \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

Ex. 2.8: Build a derivation tree to prove the transitions below

1. $(a.A + b.B) \xrightarrow{b} B$
2. $(a.b.A + (b.a.B + c.a.C)) \xrightarrow{b} a.B$
3. $((a.B + b.A)[a \mapsto c])\backslash\{a, b\} \xrightarrow{c} (B[a \mapsto c])\backslash\{a, b\}$

Exercise

Ex. 2.9: Draw the automata

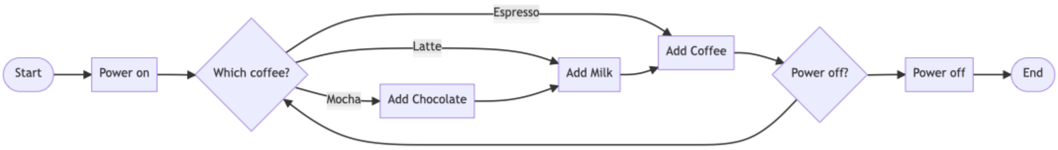
$$\begin{aligned} CM &= \mathrm{coin.coffee.}CM \\ CS &= \mathrm{pub.(coin.coffee.CS + coin.tea.CS)} \end{aligned}$$

Ex. 2.10: What is the language of the process A?

$$\begin{aligned} A &= \mathrm{goLeft.A + goRight.B} \\ B &= \mathrm{rest}.\mathbf{0} \end{aligned}$$

Check result online: `http://lmf.di.uminho.pt/ccs-caos`

Exercise



Ex. 2.11: Write the process of the flowchart above

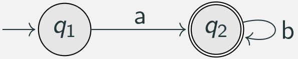
$$
\begin{aligned}
P &= \mathsf{powerOn}.Q \\
Q &= \mathsf{selMocha}.\mathsf{addChocolate}.Mk + \mathsf{selLatte}.Mk + \ldots \\
Mk &= \mathsf{addMilk}\ldots
\end{aligned}
$$

Concurrent Process algebra

Overview

Recall

1. Non-deterministic Finite Automata ($X \to \texttt{Bool} \times \mathcal{P}(X)^N$):



2. (Sequential) Process algebra: $P = a.Q \qquad Q = b.Q$
3. Meaning of (2) using (1)

Still missing

- **Interaction** between processes
- Enrich (2) and (3)

CCS - Updated Syntax

$$\mathcal{P} \ni P, Q ::= K \mid \alpha.P \mid P + Q \mid \mathbf{0} \mid P[f] \mid P\backslash L \mid P|Q$$

where

- $\alpha \in N \cup \overline{N} \cup \{\tau\}$ is an action
- K s a collection of process names or process constants
- $L \subseteq N$ is a set of labels
- f is a function that renames actions s.t. $f(\tau) = \tau$ and $f(\bar{a}) = \overline{f(a)}$
- notation:

  $[f] = [a_1 \mapsto b_1, \ldots, a_n \mapsto b_n]$ where $a_i, b_i \in N \cup \{\tau\}$

Syntax

$$\mathcal{P} \ni P, Q ::= K \mid \alpha.P \mid P + Q \mid \mathbf{0} \mid P[f] \mid P \backslash L \mid P|Q$$

Ex. 2.12: Which are syntactically correct?

$$a.\overline{b}.A + B \tag{11}$$
$$(a.\mathbf{0} + \overline{a}.A) \backslash \{\overline{a}, b\} \tag{12}$$
$$(a.\mathbf{0} + \overline{a}.A) \backslash \{a, \tau\} \tag{13}$$
$$(a.\mathbf{0} + \overline{\tau}.A) \backslash \{a\} \tag{14}$$
$$\tau.\tau.B + \overline{a}.\mathbf{0} \tag{15}$$
$$(\mathbf{0}|\mathbf{0}) + \mathbf{0} \tag{16}$$
$$(a.B + b.B)[a \mapsto a, \tau \mapsto b] \tag{17}$$
$$(a.B + \tau.B)[b \mapsto a, b \mapsto a] \tag{18}$$
$$(a.B + b.B)[a \mapsto b, b \mapsto \overline{a}] \tag{19}$$
$$(a.b.A + \overline{a}.\mathbf{0})|B \tag{20}$$
$$(a.b.A + \overline{a}.\mathbf{0}).B \tag{21}$$
$$(a.b.A + \overline{a}.\mathbf{0}) + B \tag{22}$$

CCS semantics - building an NFA

(act)
$$\frac{}{\alpha.P \xrightarrow{\alpha} P}$$

(sum-1)
$$\frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'}$$

(sum-2)
$$\frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 + P_2 \xrightarrow{\alpha} P_2'}$$

(res)
$$\frac{P \xrightarrow{\alpha} P'}{P\backslash L \xrightarrow{\alpha} P'\backslash L} \quad \alpha, \overline{\alpha} \notin L$$

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

(com1)
$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$$

(com2)
$$\frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

(com3)
$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

CCS semantics - building an NFA

$$\text{(act)}\quad \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \text{(sum-1)}\quad \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'} \qquad \text{(sum-2)}\quad \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1 + P_2 \xrightarrow{\alpha} P_2'}$$

$$\text{(res)}\quad \frac{P \xrightarrow{\alpha} P'}{P\backslash L \xrightarrow{\alpha} P'\backslash L}\ \alpha, \overline{\alpha} \notin L \qquad \text{(rel)}\quad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$\text{(com1)}\quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \text{(com2)}\quad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'} \qquad \text{(com3)}\quad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

Ex. 2.13: Draw the transition systems

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

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

$$SmUni = (CM|CS)\backslash\{\text{coin}, \text{coffee}\}$$

Ex. 2.14: Let $A = b.a.B$. Show that:

1. $(A \mid \overline{b}.\mathbf{0})\backslash\{b\} \xrightarrow{\tau} (a.B \mid \mathbf{0})\backslash\{b\}$
2. $(A \mid b.a.B) + ((b.A)[b \mapsto a]) \xrightarrow{a} A[b \mapsto a]$

Ex. 2.15: Draw the NFAs A and D

$$A = x.B + x.x.C \qquad D = x.x.x.D + x.E$$

$$B = x.x.A + y.C \qquad E = x.F + y.F$$

$$C = x.A \qquad F = x.D$$

mCRL2 Tools – generate automata

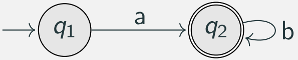
Slides 3:

`https://fm-dcc.github.io/sv2425/slides/3-mcrl2.pdf`

Observational Equivalence

Recall

1. F-transition systems, e.g., Non-deterministic Finite Automata:



2. Process algebra: $P = a.Q \quad Q = b.Q \quad P|Q$
3. Interaction between processes
4. Meaning of CCS using transition systems

Still missing

- When is a process P equivalent to a process Q?
- When can a process P be safely replaced by a process Q?

Observational Equivalence Informally

Two programs are observationally equivalent if it is impossible to observe any difference in their behaviour

Here behaviour is described in terms of transition systems

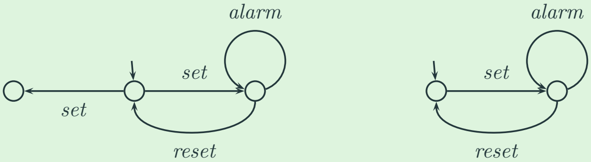
. . . and therefore behaviour/equivalence needs to be pinned down to them

EQ1 – Language equivalence

Definition

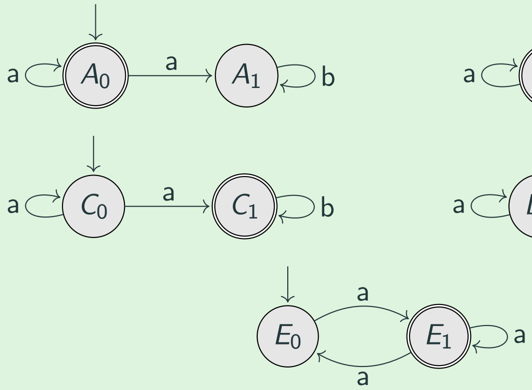
Two automata A, B are language equivalent iff $L_A = L_B$
(i.e. if they can perform the same finite sequences of transitions)

Example



Language equivalence applies when one can neither interact with a system, nor distinguish a slow system from one that has come to a stand still.

Ex. 2.16: Find pairs of automata with the same language



Exercise

Ex. 2.17: Check if the processes are language equivalent

$$P = coin.(\overline{coffee}.P + \overline{tea}.P) \qquad Q = coin.\overline{coffee}.Q + coin.\overline{tea}.Q$$

EQ2 – Similarity

the quest for a behavioural equality:
able to identify states that cannot be distinguished by any realistic form of observation

Simulation

A state q simulates another state p if
every transition from q is corresponded by a transition from p and
this capacity is kept along the whole life of the system to which state space q belongs to.

Simulation of NFA $(X \to \mathcal{P}(X)^N)$

Definition

Given NFA A_1 and A_2 over N with states S_1 and S_2 respectively, a relation $R \subseteq S_1 \times S_2$ is a simulation iff, for all $\langle p, q \rangle \in R$ and $a \in N$,

$$(1) \quad p \xrightarrow{a}_1 p' \;\Rightarrow\; \langle \exists\, q' \;:\; q' \in S_2 :\; q \xrightarrow{a}_2 q' \wedge \langle p', q' \rangle \in R \rangle$$

$$\begin{tikzcd}
p \arrow[d, "a"] & R & q \\
p' & &
\end{tikzcd}
\qquad \Longrightarrow \qquad
\begin{tikzcd}
& & q \arrow[d, "a"] \\
p' & R & q'
\end{tikzcd}$$

Ex. 2.18: Find simulations

$$\begin{tikzcd}
 & q_1 \arrow[r, "d"] & q_2 \\
q_0 \arrow[ur, "a"] \arrow[dr, "a"'] & & \\
 & q_4 \arrow[r, "e"'] & q_3
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & p_2 \\
p_0 \arrow[r, "a"] & p_1 \arrow[ur, "d"] \arrow[dr, "e"'] & \\
 & & p_3
\end{tikzcd}$$

Ex. 2.18: Find simulations

$$
\begin{tikzcd}
 & q_1 \arrow[r, "d"] & q_2 & & & p_2 \\
q_0 \arrow[ur, "a"] \arrow[dr, "a"'] & & & p_0 \arrow[r, "a"] & p_1 \arrow[ur, "d"] \arrow[dr, "e"'] & \\
 & q_4 \arrow[r, "e"'] & q_3 & & & p_3
\end{tikzcd}
$$

$$q_0 \lesssim p_0 \qquad \text{cf.} \qquad \{\langle q_0, p_0\rangle, \langle q_1, p_1\rangle, \langle q_4, p_1\rangle, \ldots\}$$

Definition

$$p \lesssim q \;\equiv\; \langle \exists\, R \;::\; R \text{ is a simulation and } \langle p, q\rangle \in R\rangle$$

We say p is simulated by q.

Lemma

The similarity relation is a preorder
(ie, reflexive and transitive)

EQ3 – Bisimilarity

Bisimulation

Definition

Given NFA A_1 and A_2 over N with states S_1 and S_2 respectively, relation $R \subseteq S_1 \times S_2$ is a bisimulation iff both R and its converse R° are simulations.

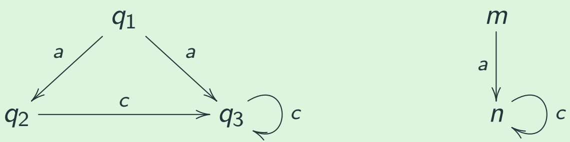
I.e., whenever $\langle p, q \rangle \in R$ and $a \in N$,

$$
\begin{array}{ll}
(1) & p \xrightarrow{a}_1 p' \;\Rightarrow\; \langle \exists\, q' \;:\; q' \in S_2 :\; q \xrightarrow{a}_2 q' \wedge \langle p', q' \rangle \in R \rangle \\
(2) & q \xrightarrow{a}_2 q' \;\Rightarrow\; \langle \exists\, p' \;:\; p' \in S_1 :\; p \xrightarrow{a}_1 p' \wedge \langle p', q' \rangle \in R \rangle
\end{array}
$$

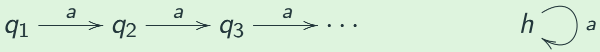
$$
\begin{tikzcd}
p \arrow[d, "a"] & R & q & & & q \arrow[d, "a"] \\
p' & & & \Rightarrow & p' & R & q'
\end{tikzcd}
\qquad
\begin{tikzcd}
p \arrow[d, "a"] & & & & p & R & q \arrow[d, "a"] \\
p' & R & q' & \Leftarrow & & & q'
\end{tikzcd}
$$

Examples

Ex. 2.19: Find bisimulations that include $\langle q_1, m \rangle$



Ex. 2.20: Find bisimulations that include $\langle q_1, h \rangle$



Bisimilarity

Definition

$$p \sim q \;\equiv\; \langle \exists\, R \;::\; R \text{ is a bisimulation and } \langle p, q \rangle \in R \rangle$$

We say p is bisimilar to q.

Lemma

Two processes P and Q are bisimilar if there is a bisimulation that includes $\langle P, Q \rangle$.

Lemma

The bisimilarity relation is an equivalence relation
(ie, symmetric, reflexive and transitive)

Exercises

Ex. 2.21: Check if there is a bisimulation that include $\langle q_1, p_1 \rangle$

$$
\begin{tikzcd}
& q_1 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
q_2 \arrow[d, "c"] & & q_3 \arrow[d, "c"] \\
q_4 & & q_5
\end{tikzcd}
\qquad
\begin{tikzcd}
& p_1 \arrow[d, "a"] & \\
& p_2 \arrow[dl, "c"'] \arrow[dr, "c"] & \\
p_4 & & p_5
\end{tikzcd}
$$

Ex. 2.22: Check if there is a bisimulation that include $\langle P, Q \rangle$

$$P = coin.(\overline{coffee}.P + \overline{tea}.P) \qquad Q = coin.\overline{coffee}.Q + coin.\overline{tea}.Q$$

Ex. 2.23: Check if there is a bisimulation that include $\langle q_1, p_1 \rangle$

$$
\begin{tikzcd}
 & q_1 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
q_2 \arrow[d, "b"] & & q_3 \arrow[d, "c"] \\
q_4 & & q_5
\end{tikzcd}
\qquad
\begin{tikzcd}
 & p_1 \arrow[d, "a"] & \\
 & p_2 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
p_4 & & p_5
\end{tikzcd}
$$

Ex. 2.24: Check if, for any process P

$$P \sim P + \mathbf{0}$$

mCRL2 Tools – check bisimilarity

Slides 3:

`https://fm-dcc.github.io/sv2425/slides/3-mcrl2.pdf`

Generalising Observational Equivalences

Definition

Fix a functor F and consider two transition systems $f : X \longrightarrow FX$ and $g : Y \longrightarrow FY$. Two states $x \in X$, $y \in Y$ are observationally equivalent if

- there exists a relation $R \subseteq X \times Y$ with $(x, y) \in R$ and
- there exists a transition system $b : R \longrightarrow FR$ such that the diagram below commutes

$$\begin{tikzcd} X \arrow[d, "f"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "b"'] & Y \arrow[d, "g"] \\ FX & FR \arrow[l, "F\pi_1"] \arrow[r, "F\pi_2"'] & FY \end{tikzcd}$$

If such is the case we write $x \sim y$

Observational Equivalence for Moore Machine

Given $\langle o_1, n_1 \rangle : X \to A \times X$ and $\langle o_2, n_2 \rangle : Y \to A \times Y$ we obtain from the previous slide that $x \sim y$ iff

- $o_1(x) = o_2(y)$
- $n_1(x) \sim n_2(y)$

Observational Equivalence for Labelled Transition Systems

Recall that we used systems of type $X \to \mathrm{P}(X)^N$ for establishing the semantics of CCS processes. This means that ...

notions of observational behaviour/equivalence for such transition systems directly impact our concurrent language

Given $\overline{t_1} : X \to \mathrm{P}(X)^N$ and $\overline{t_2} : Y \to \mathrm{P}(Y)^N$, $x \sim y$ iff for all $l \in N$

- $\forall x' \in t_1(x, n).\ \exists y' \in t_2(y, n).\ x' \sim y'$
- $\forall y' \in t_2(y, n).\ \exists x' \in t_1(x, n).\ x' \sim y'$