

2. Transition Systems

José Proença

System Verification (CC4084) 2024/2025

CISTER – U.Porto, Porto, Portugal

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



CISTER - Research Centre in
Real-Time & Embedded
Computing Systems

- 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?

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 $x := p ; q$ is forbidden by the syntactic rules of most programming languages

Ex. 2.2: Semantics

The sentence/program $x := 1$ has the meaning “writes 1 in the memory address corresponding to x ”

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

Definition (Functor)

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

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

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

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

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



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,

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

Ex. 2.5: Powerset on Booleans

$$P(\text{Bool}) \mapsto \{\emptyset, \{\top\}, \{\perp\}, \{\top, \perp\}\}$$

Definition (Transition system)

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

Some famous examples of F -transition systems

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

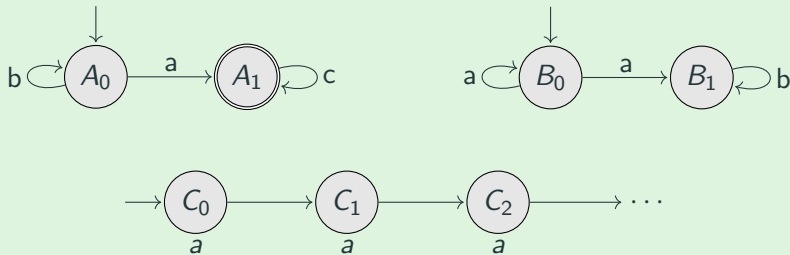


Distribution functor



Powerset functor

Ex. 2.6: Formalise as an F-transition system



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

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

where

- $\alpha \in N \cup \{\tau\}$ is an **action**
- K is 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:**

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

Syntax

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

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

$$a.b.A + B \quad (1)$$

$$(a.\mathbf{0} + b.A) \setminus \{a, b, c\} \quad (2)$$

$$(a.\mathbf{0} + b.A) \setminus \{a, \tau\} \quad (3)$$

$$a.B + [b \mapsto a] \quad (4)$$

$$\tau.\tau.B + \mathbf{0} \quad (5)$$

$$a.(a + b).A \quad (6)$$

$$(a.B + b.B)[a \mapsto a, \tau \mapsto b] \quad (7)$$

$$(a.B + \tau.B)[b \mapsto a, a \mapsto a] \quad (8)$$

$$(a.b.A + b.\mathbf{0}).B \quad (9)$$

$$(a.b.A + b.\mathbf{0}) + B \quad (10)$$

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

$$\begin{array}{c}
 \text{(act)} \\
 \hline
 \alpha.P \xrightarrow{\alpha} P
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(sum-1)} \\
 \frac{P_1 \xrightarrow{\alpha} P'_1}{P_1 + P_2 \xrightarrow{\alpha} P'_1}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(sum-2)} \\
 \frac{P_2 \xrightarrow{\alpha} P'_2}{P_1 + P_2 \xrightarrow{\alpha} P'_2}
 \end{array}$$

$$\begin{array}{c}
 \text{(res)} \\
 \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad \alpha \notin L
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(rel)} \\
 \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}
 \end{array}$$

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

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

$$\begin{array}{c}
 \text{(act)} \\
 \hline
 \alpha.P \xrightarrow{\alpha} P
 \end{array}
 \quad
 \begin{array}{c}
 \text{(sum-1)} \\
 \frac{P_1 \xrightarrow{\alpha} P'_1}{P_1 + P_2 \xrightarrow{\alpha} P'_1}
 \end{array}
 \quad
 \begin{array}{c}
 \text{(sum-2)} \\
 \frac{P_2 \xrightarrow{\alpha} P'_2}{P_1 + P_2 \xrightarrow{\alpha} P'_2}
 \end{array}$$

$$\begin{array}{c}
 \text{(res)} \\
 \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad \alpha \notin L
 \end{array}
 \quad
 \begin{array}{c}
 \text{(rel)} \\
 \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}
 \end{array}$$

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. $((a.B + b.A)[a \mapsto c]) \setminus \{a, b\} \xrightarrow{c} (B[a \mapsto c]) \setminus \{a, b\}$

Ex. 2.9: Draw the automata

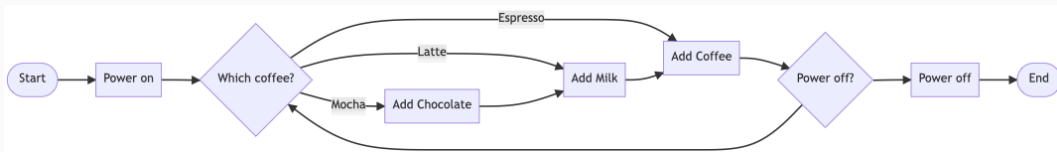
$$CM = \text{coin.coffee}.CM$$

$$CS = \text{pub.}(\text{coin.coffee}.CS + \text{coin.tea}.CS)$$

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

$$A = \text{goLeft}.A + \text{goRight}.B$$

$$B = \text{rest}.\mathbf{0}$$



Ex. 2.11: Write the process of the flowchart above

$P = \text{powerOn}.Q$

$Q = \text{selMocha.addChocolate}.Mk + \text{selLatte}.Mk + \dots$

$Mk = \text{addMilk} \dots$

Concurrent Process algebra

Recall

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



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

Still missing

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

CCS - Updated Syntax

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

where

- $\alpha \in N \cup \overline{N} \cup \{\tau\}$ is an action
- K is 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 $f(\overline{a}) = \overline{f(a)}$
- notation:

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

Syntax

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

Ex. 2.12: Which are syntactically correct?

$$a.\bar{b}.A + B \quad (11)$$

$$(a.\mathbf{0} + \bar{a}.A) \setminus \{\bar{a}, b\} \quad (12)$$

$$(a.\mathbf{0} + \bar{a}.A) \setminus \{a, \tau\} \quad (13)$$

$$(a.\mathbf{0} + \bar{\tau}.A) \setminus \{a\} \quad (14)$$

$$\tau.\tau.B + \bar{a}.\mathbf{0} \quad (15)$$

$$(\mathbf{0}|\mathbf{0}) + \mathbf{0} \quad (16)$$

$$(a.B + b.B)[a \mapsto a, \tau \mapsto b] \quad (17)$$

$$(a.B + \tau.B)[b \mapsto a, b \mapsto a] \quad (18)$$

$$(a.B + b.B)[a \mapsto b, b \mapsto \bar{a}] \quad (19)$$

$$(a.b.A + \bar{a}.\mathbf{0})|B \quad (20)$$

$$(a.b.A + \bar{a}.\mathbf{0}).B \quad (21)$$

$$(a.b.A + \bar{a}.\mathbf{0}) + B \quad (22)$$

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

$$\frac{(\text{sum-1}) \quad P_1 \xrightarrow{\alpha} P'_1}{P_1 + P_2 \xrightarrow{\alpha} P'_1}$$

$$\frac{(\text{sum-2}) \quad P_2 \xrightarrow{\alpha} P'_2}{P_1 + P_2 \xrightarrow{\alpha} P'_2}$$

$$\frac{(\text{res}) \quad P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad \alpha, \bar{\alpha} \notin L$$

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

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

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

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

$$\begin{array}{c}
 \text{(act)} \\
 \hline
 \alpha.P \xrightarrow{\alpha} P
 \end{array}
 \quad
 \begin{array}{c}
 \text{(sum-1)} \\
 \hline
 \frac{P_1 \xrightarrow{\alpha} P'_1}{P_1 + P_2 \xrightarrow{\alpha} P'_1}
 \end{array}
 \quad
 \begin{array}{c}
 \text{(sum-2)} \\
 \hline
 \frac{P_2 \xrightarrow{\alpha} P'_2}{P_1 + P_2 \xrightarrow{\alpha} P'_2}
 \end{array}$$

$$\begin{array}{c}
 \text{(res)} \\
 \hline
 \frac{P \xrightarrow{\alpha} P'}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \quad \alpha, \bar{\alpha} \notin L
 \end{array}
 \quad
 \begin{array}{c}
 \text{(rel)} \\
 \hline
 \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}
 \end{array}$$

$$\begin{array}{c}
 \text{(com1)} \\
 \hline
 \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}
 \end{array}
 \quad
 \begin{array}{c}
 \text{(com2)} \\
 \hline
 \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}
 \end{array}
 \quad
 \begin{array}{c}
 \text{(com3)} \\
 \hline
 \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}
 \end{array}$$

Ex. 2.13: Draw the transition systems

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

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

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

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

1. $(A \mid \bar{b}.0) \setminus \{b\} \xrightarrow{\tau} (a.B \mid 0) \setminus \{b\}$
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

$$A = x.B + x.x.C$$

$$B = x.x.A + y.C$$

$$C = x.A$$

$$D = x.x.x.D + x.E$$

$$E = x.F + y.F$$

$$F = x.D$$

mCRL2 Tools – generate automata

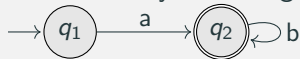
Slides 3:

<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: $P = a.Q$ $Q = b.Q$ $P|Q$
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 ?

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

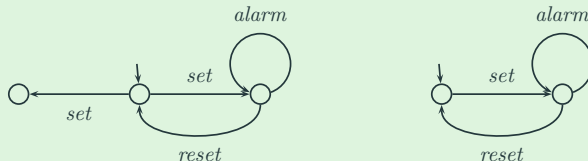
... and therefore behaviour/equivalence needs to be pinned down to them

EQ1 – Language equivalence

Definition

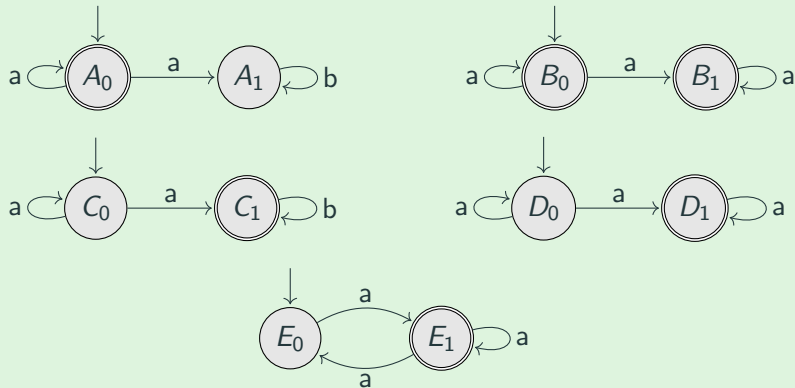
Two automata A, B are **language equivalent** iff $L_A = L_B$
(i.e. if they can perform the same finite sequences of transitions)

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



Ex. 2.17: Check if the processes are language equivalent

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

$$Q = \text{coin} . \overline{\text{coffee}} . Q + \text{coin} . \overline{\text{tea}} . Q$$

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.

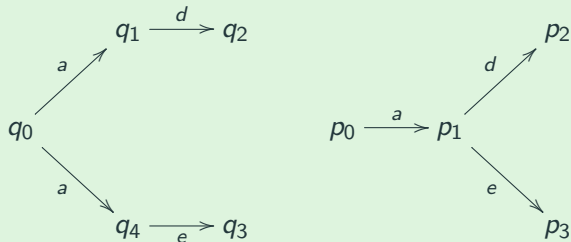
Definition

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 $a \in N$,

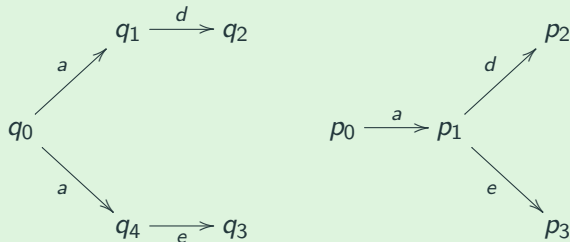
$$(1) \quad p \xrightarrow{a}_1 p' \Rightarrow \langle \exists q' : q' \in S_2 : q \xrightarrow{a}_2 q' \wedge \langle p', q' \rangle \in R \rangle$$



Ex. 2.18: Find simulations



Ex. 2.18: Find simulations



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

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

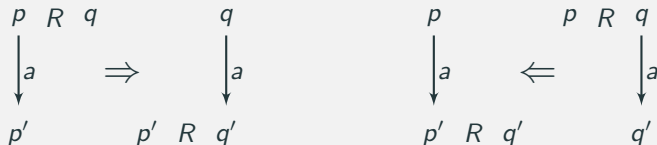
Definition

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.

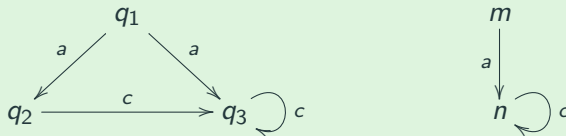
I.e., whenever $\langle p, q \rangle \in R$ and $a \in N$,

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

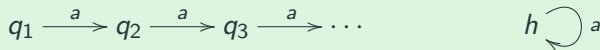
$$(2) \quad q \xrightarrow{a}_2 q' \Rightarrow \langle \exists p' : p' \in S_1 : p \xrightarrow{a}_1 p' \wedge \langle p', q' \rangle \in R \rangle$$



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



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



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*.

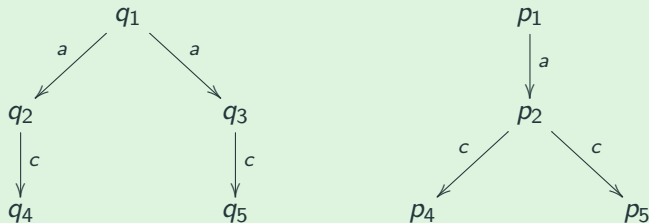
Lemma

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)

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

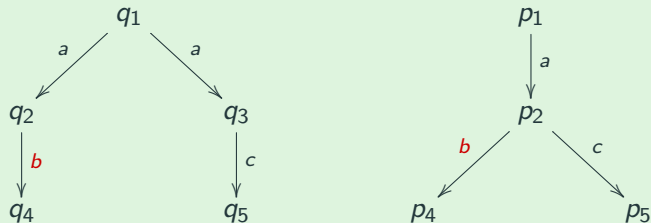


Ex. 2.22: Check if there is a bisimulation that include $\langle P, Q \rangle$

$$P = \text{coin}.\overline{\text{coffee}}.P + \overline{\text{tea}}.P$$

$$Q = \text{coin}.\overline{\text{coffee}}.Q + \text{coin}.\overline{\text{tea}}.Q$$

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



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 $f : X \rightarrow FX$ and $g : Y \rightarrow FY$.

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** $b : R \rightarrow FR$ such that the diagram below commutes

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Y \\ f \downarrow & & b \downarrow & & \downarrow g \\ FX & \xleftarrow{F\pi_1} & FR & \xrightarrow{F\pi_2} & FY \end{array}$$

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

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

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

Recall that we used systems of type $X \rightarrow P(X)^N$ 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 $\bar{t}_1 : X \rightarrow P(X)^N$ and $\bar{t}_2 : Y \rightarrow P(Y)^N$, $x \sim y$ iff for all $l \in N$

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