

7. Behavioural Modelling

David Pereira José Proença

RAMDE 2021/2022

Requirements and Model-driven Engineering

CISTER – ISEP

Porto, Portugal

<https://cister-labs.github.io/ramde2122>

Overview

So far

- Models and properties for **structures**: boolean and 1st order logic, ...
- Useful, e.g., for UML class diagrams

Next

- Look at UML **behaviour** diagrams
- Use a domain with a **precise semantics**
 - Non-deterministic finite **automata** (NFA)
 - Simple language for **processes**
 - Encode processes \rightarrow NFA
 - Equivalence of processes



What are formal methods?

Formal methods are **techniques** to
model **complex systems** using
rigorous mathematical models

Specification

Define part of the system
using a modelling
language

Verification

Prove properties.
Show correctness.
Find bugs.

Implementation

Generate correct code.

All formal models are wrong

All formal models are wrong
... but some of them are usefull!

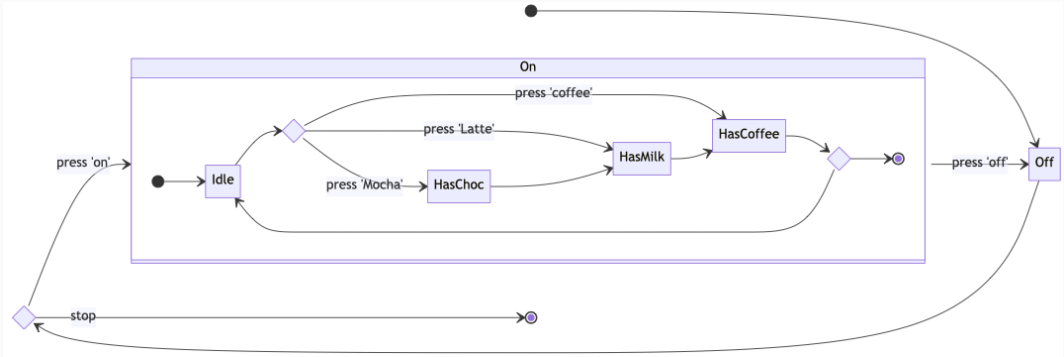
- High-level overview or requirements and associated processes
- Mathematical Preliminaries
 - Basic mathematical notations
 - Set theory
 - Propositional Logic
 - First Order Logic
 - The Z3 automatic theorem prover
- Behavioural modelling
 - Single component
 - State diagrams and Flow charts
 - Formal modelling: Automata, Process Algebra in mCRL2
 - Many components
 - Communication diagrams and Sequence diagrams
 - Formal modelling: Process algebra with interactions
 - Equivalences
 - Verification

UML behaviour diagrams

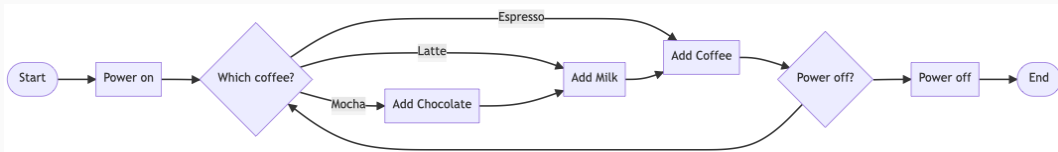
Describe the **state** of a component, what **actions** it can do, and how it **evolves** during its life cycle.

- **State Diagram** focus on states
- **Flowchart** focus on actions (also known as *activity diagrams*)

Coffee State Diagram



Coffee Flowchart



Used symbols: *processes, decisions, and start/end*

Other symbols include: data (or input/output), documents, connectors, comments

Automata – Basic definitions

Sequential and Reactive systems

Sequential systems

Meaning is defined by the results of finite computations

We start here. . .

Reactive systems

Meaning is determined by **interaction** and **mobility** of **non-terminating** processes, evolving **concurrently**

then we go reactive

Non-Deterministic Finite Automata (NFA)

Definition

A NFA over a set N of **names** is a tuple $\langle S, I, \downarrow, N, \longrightarrow \rangle$ where

- $S = \{s_0, s_1, s_2, \dots\}$ is a set of **states**
- $I \subseteq S$ is the set of **initial** states
- $\downarrow \subseteq S$ is the set of terminating or **final** states

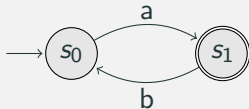
$$\downarrow s \equiv s \in \downarrow$$

- $\longrightarrow \subseteq S \times N \times S$ is the **transition** relation, often given as an N -indexed family of binary relations

$$s \xrightarrow{a} s' \equiv \langle s, a, s' \rangle \in \longrightarrow$$

Example

Example of an automaton

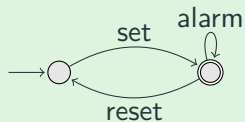
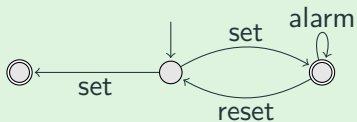


s_0 is an initial state

s_1 is a final state

(Formalise this automata)

Ex. 7.1: Formalise these automata as $\langle S, I, \downarrow, N, \longrightarrow \rangle$



A note on Homework

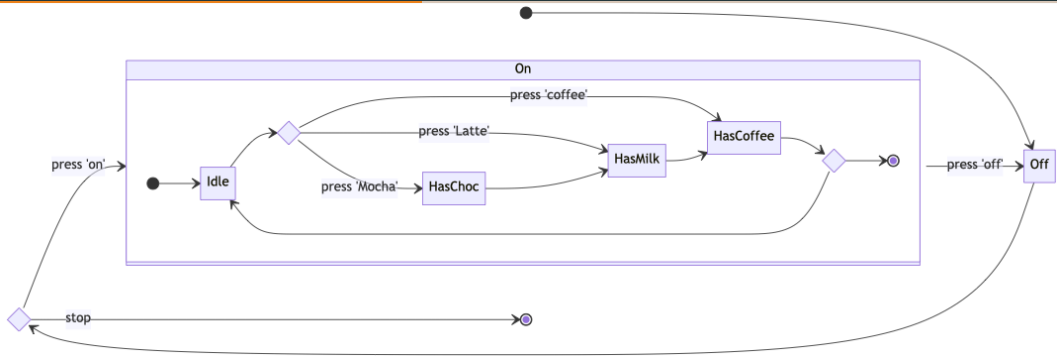
- 10% of the final mark
- focus on effort – doing badly is better than not doing
- submission: a PDF by email to the teacher who provided the exercises; here pro@isep.ipp.pt.

Deadlines

Exercises presented in a given week must be submitted by the end of the following week, Sunday @ 23h59.

Website/Teams will be kept up-to-date with ongoing open submissions.

Exercise



Ex. 7.2: Draw LTS

(suggestion: by hand on a paper, and take a photo of it.)

Labelled Transition System

More generally, a NFA $\langle S, I, \downarrow, N, \longrightarrow \rangle$ is a **labelled transition system** (LTS) $\langle S, N, \longrightarrow \rangle$, where each state $s \in S$ determines a **system** over all states reachable from s and the corresponding restriction of \longrightarrow .

LTS classification

- deterministic
- non deterministic
- finite
- finitely branching
- image finite
- ...

Definition

The **reachability relation**, $\longrightarrow^* \subseteq S \times N^* \times S$, is defined inductively

- $s \xrightarrow{\epsilon}^* s$ for each $s \in S$, where $\epsilon \in N^*$ denotes the empty **word**;
- if $s \xrightarrow{a} s''$ and $s'' \xrightarrow{\sigma}^* s'$ then $s \xrightarrow{a\sigma}^* s'$, for $a \in N, \sigma \in N^*$

Reachable state

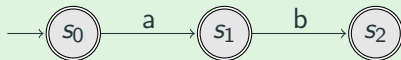
$t \in S$ is **reachable** from $s \in S$ iff there is a **word** $\sigma \in N^*$ st $s \xrightarrow{\sigma}^* t$

Language

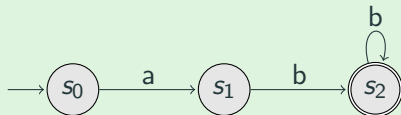
A word σ is in the language L_A of an automata $A = \langle S, I, \downarrow, N, \longrightarrow \rangle$
iff
there are states $s \in I, s' \in \downarrow$ such that $s \xrightarrow{\sigma^*} s'$.

Exercises

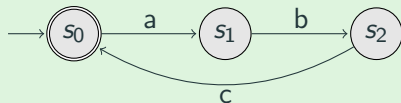
Ex. 7.3: What is the language of this automata?



Ex. 7.4: What is the language of this automata?



Ex. 7.5: What is the language of this automata?



Regular Expressions – syntax

- $w_1 w_2$: word w_1 followed by word w_2
- $w_1 + w_2$: word w_1 or word w_2
- a^* : 0 or more a 's
- a^+ : 1 or more a 's
- ϵ : empty word

Examples

- $ab + c$: (a followed by b) or c
- $(ab)^* b$: b or abb or $ababb$ or ...
- $c((ab)^* b)^+$: cb or $cabb$ or $cababb$ or ...

Extra: Regular Expressions

Regular Expressions – syntax

- $w_1 w_2$: word w_1 followed by word w_2
- $w_1 + w_2$: word w_1 or word w_2
- a^* : 0 or more a 's
- a^+ : 1 or more a 's
- ϵ : empty word

Examples

- $ab + c$: (a followed by b) or c
- $(ab)^* b$: b or abb or $ababb$ or ...
- $c((ab)^* b)^+$: cb or $cabb$ or $cababb$ or ...

NFA vs. Reg. Expr.

Word w expressible by a NFA $\Leftrightarrow w$ expressible by a Reg. Expr.

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
- I is an indexing set
- $L \subseteq N$ 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. 7.6: Which are NOT syntactically correct? Why?

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

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

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

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

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

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

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

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

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

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

CCS semantics - building a NFA

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

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

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

CCS semantics - building a NFA

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

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

Ex. 7.7: 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. $((a.B + b.A)[a \mapsto c]) \setminus \{a, b\} \xrightarrow{c} B$

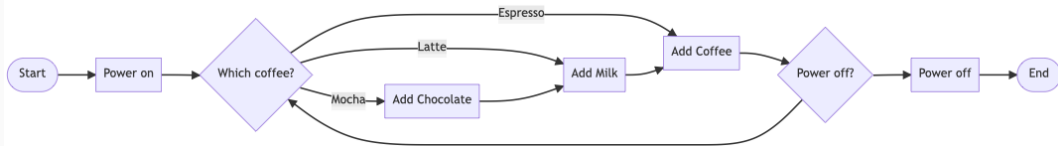
Ex. 7.8: Draw the automata

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

Ex. 7.9: What is the language of this process?

$$A = \text{goLeft}.A + \text{goRight}.B$$
$$B = \text{rest}.\mathbf{0}$$

Exercise



Ex. 7.10: Write the process of the flowchart above

$P = \text{powerOn}.Q$

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

$Mk = \text{addMilk} \dots$

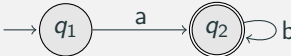
mCRL2 Tools

Slides 10:

<https://cister-labs.github.io/ramde2122/slides/10-mcrl2.pdf>

Concurrent Process algebra

Recall

1. Non-deterministic Finite Automata: 
2. (Sequential) Process algebra: $P = a.Q$ $Q = b.Q$
3. Meaning of (2) using (1)

Still missing

- **Interaction** between processes
- *Interaction diagrams* vs. *interacting 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
- I is an indexing set
- $L \subseteq N$ 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. 7.11: Which are syntactically correct?

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

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

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

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

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

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

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

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

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

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

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

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

CCS semantics - building an NFA

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

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

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

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

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

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

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

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

CCS semantics - building an NFA

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

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

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

Ex. 7.12: Draw the NFAs

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

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

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

Ex. 7.13: Let $A = b.a.B$. Show that:

1. $(A \mid \bar{b}.0) \setminus \{b\} \xrightarrow{\tau} (a.B \mid 0) \setminus \{b\}$
2. $(A \mid b.a.B) + (b.A)[a/b] \xrightarrow{a} A[a/b]$

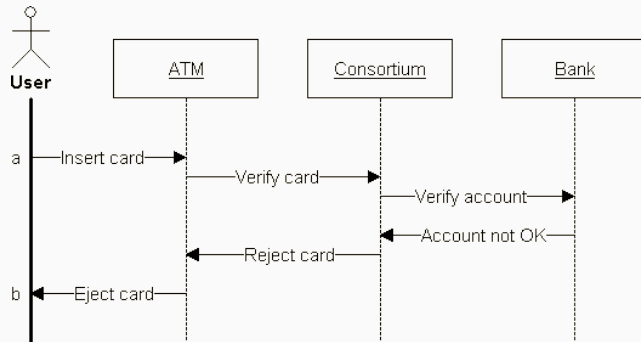
mCRL2 Tools

Slides 10:

<https://cister-labs.github.io/ramde2122/slides/10-mcrl2.pdf>

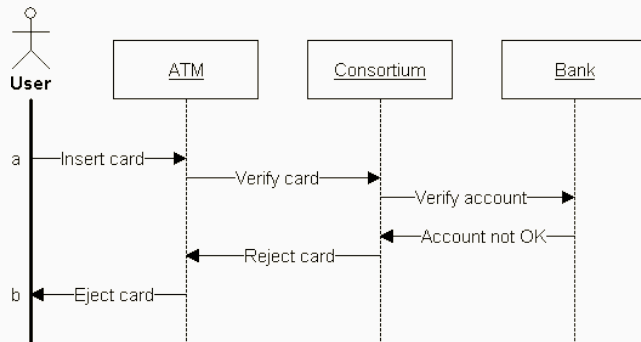
Sequence Diagrams vs. Interactive Processes

Sequence Diagrams as Interactive Processes



- **Objects** as **Processes**
(e.g., processes U , A , C , B)
- **Send** actions (e.g., *insertCard*)
- **Receive** actions (e.g., *insertCard*)
- Unique action for each object pair
- Do not write $(\dots + \mathbf{0})$

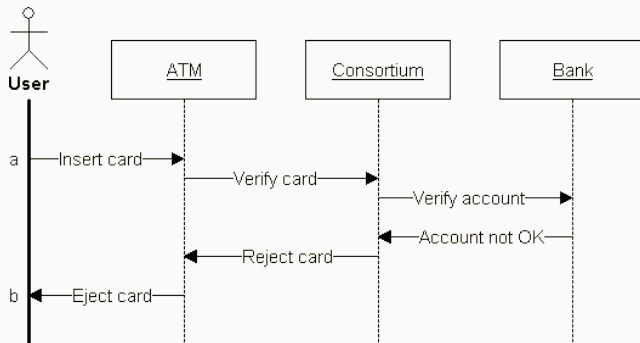
Language of Sequence Diagrams, Informally



This example has only 1 word and its prefixes

$\text{Tr}(sd) = \text{insertCard} \cdot \text{verifyCard} \cdot \text{verifyAccount} \cdot \text{accountNotOK} \cdot$
 $\text{rejectedCard} \cdot \text{ejectCard}$

Sequence Diagrams as Interactive Processes



Ex. 7.14: Write an interactive processes that acts as above

$Sys = (U|A|C|E)$
 $U = \text{insertCard}.\overline{\text{ejectCard}}.0$
 $A = \dots$
 $C = \dots$
 $E = \dots$