Labelled Transition Systems

Luís Soares Barbosa

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

UNITED NATIONS
UNIVERSITY
UNU-EGOV

Interaction & Concurrency Course Unit (Lcc)

Universidade do Minho, 04.II.2019

Reactive systems

Reactive system

system that computes by reacting to stimuli from its environment along its overall computation

- in contrast to sequential systems whose meaning is defined by the results of finite computations, the behaviour of reactive systems is mainly determined by interaction and mobility of non-terminating processes, evolving concurrently.
- observation $\equiv$ interaction
- behaviour $\equiv$ a structured record of interactions

Reactive systems

Concurrency vs interaction

$$x := 0;$$
$$x := x + 1 \mid x := x + 2$$

- both statements in parallel could read x before it is written
- which values can x take?
- which is the program outcome if exclusive access to memory and atomic execution of assignments is guaranteed?

Labelled Transition System

Definition

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

- $S = \{s_0, s_1, s_2, ...\}$ is a set of 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

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

Labelled Transition System

Morphism

A morphism relating two LTS over N, $\langle S, N, \downarrow, \longrightarrow \rangle$ and $\langle S', N, \downarrow', \longrightarrow' \rangle$, is a function $h : S \longrightarrow S'$ st

$$\begin{array}{rcl} s \xrightarrow{a} s' & \Rightarrow & h\ s \xrightarrow{a}' h\ s' \\ s \downarrow & \Rightarrow & h\ s \downarrow' \end{array}$$

morphisms preserve transitions and termination

Labelled Transition System

System

Given a LTS $\langle S, N, \downarrow, \longrightarrow \rangle$, each state $s \in S$ determines a system over all states reachable from s and the corresponding restrictions of $\longrightarrow$ and $\downarrow$.

LTS classification

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

Reachability

Definition

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

- $s \stackrel{\epsilon}{\longrightarrow}^* s$ for each $s \in S$, where $\epsilon \in N^*$ denotes the empty word;
- if $s \stackrel{a}{\longrightarrow} s''$ and $s'' \stackrel{\sigma}{\longrightarrow}^* s'$ then $s \stackrel{a\sigma}{\longrightarrow}^* s'$, 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 $s \stackrel{\sigma}{\longrightarrow}^* t$

Labelled Transition System

Alternative characterization (coalgebraic)

A morphism $h : \langle S, \text{next}\rangle \longrightarrow \langle S', \text{next}'\rangle$ is a function $h : S \longrightarrow S'$ st the following diagram commutes

$$\begin{array}{ccc} S \times N & \xrightarrow{\text{next}} & \mathcal{P}S \\ {\scriptstyle h\times id}\Big\downarrow & & \Big\downarrow{\scriptstyle \mathcal{P}h} \\ S' \times N & \xrightarrow{\text{next}'} & \mathcal{P}S' \end{array}$$

i.e.,

$$\mathcal{P}h \cdot \text{next} \quad = \quad \text{next}' \cdot (h \times id)$$

or, going pointwise,

$$\{h\ x \mid x \in \text{next}\ \langle s, a\rangle\} \quad = \quad \text{next}'\ \langle h\ s, a\rangle$$

Labelled Transition System

Alternative characterization (coalgebraic)

A morphism $h : \langle S, \text{next} \rangle \longrightarrow \langle S', \text{next}' \rangle$

- preseves transitions:

$$s' \in \text{next} \ \langle s, a \rangle \Rightarrow h \ s' \in \text{next}' \ \langle h \ s, a \rangle$$

- reflects transitions:

$$r' \in \text{next}' \ \langle h \ s, a \rangle \Rightarrow \langle \exists \ s' \in S \ : \ s' \in \text{next} \ \langle s, a \rangle : \ r' = h \ s' \rangle$$

(why?)

Comparison

- Both definitions coincide at the object level:

$$\langle s, a, s' \rangle \in T \quad \equiv \quad s' \in \text{next}\ \langle s, a \rangle$$

- Wrt morphisms, the relational definition is more general, corresponding, in coalgebraic terms to

$$\mathcal{P}h \cdot \text{next} \quad \subseteq \quad \text{next}' \cdot (h \times id)$$

Automata

Back to old friends?

automaton behaviour $\equiv$ accepted language

Recall that finite automata recognize regular languages, i.e. generated by

- $L_1 + L_2 \;\widehat{=}\; L_1 \cup L_2$ (union)
- $L_1 \cdot L_2 \;\widehat{=}\; \{st \mid s \in L_1, t \in L_2\}$ (concatenation)
- $L^* \;\widehat{=}\; \{\epsilon\} \cup L \cup (L \cdot L) \cup (L \cdot L \cdot L) \cup \ldots$ (iteration)

Automata

There is a syntax to specify such languages:

$$E \quad ::= \quad \epsilon \mid a \mid E + E \mid E\,E \mid E^*$$

where $a \in \Sigma$.

- which regular expression specifies $\{a, bc\}$?
- and $\{ca, cb\}$?

and an algebra of regular expressions:

$$\begin{aligned} (E_1 + E_2) + E_3 &= E_1 + (E_2 + E_3) \\ (E_1 + E_2)\,E_3 &= E_1\,E_3 + E_2\,E_3 \\ E_1\,(E_2\,E_1)^* &= (E_1\,E_2)^*\,E_1 \end{aligned}$$

Automata

There is a syntax to specify such languages:

$$E \quad ::= \quad \epsilon \mid a \mid E + E \mid E\,E \mid E^*$$

where $a \in \Sigma$.

- which regular expression specifies $\{a, bc\}$?
- and $\{ca, cb\}$?

and an algebra of regular expressions:

$$\begin{aligned}(E_1 + E_2) + E_3 &= E_1 + (E_2 + E_3) \\ (E_1 + E_2)\,E_3 &= E_1\,E_3 + E_2\,E_3 \\ E_1\,(E_2\,E_1)^* &= (E_1\,E_2)^*\,E_1\end{aligned}$$

After thoughts

... need more general models and theories:

- Several interaction points ($\neq$ functions)
- Need to distinguish normal from anomolous termination (eg deadlock)
- Non determinisim should be taken seriously: the notion of equivalence based on accepted language is blind wrt non determinism
- Moreover: the reactive characters of systems entail that not only the generated language is important, but also the states traversed during an execution of the automata.

The course

Aims

- To become familiar with reactive systems, emphasising their concurrent composition and continuous interaction with their environment
- To introduce techniques for (formal) specification, analysis and verification of reactive systems

The course

1. Basic models for reactive systems
   (state, behaviour, interaction, concurrency)
   1.1 Labelled transition systems
   1.2 Similarity and bisimilarity
2. Process algebra
   2.1 Processes and behaviour
   2.2 CCS and mCRL2
3. Logics for reactive systems
   3.1 Hennessy-Milner logic and its extensions
   3.2 Modal, hybrid and temporal logics
   3.3 Specification and verification of logic constraints
4. Quantum processes
   4.1 Introduction to the quantum computation model
   4.2 Quantum processes and algorithms

Bibliography

Basic

1. Luca Aceto, Anna Inglfsdttir, Kim G. Larsen, Jiri Srba. *Reactive Systems: Modelling, Specification and Verification*. CUP, 2007.
2. Jan Friso Groote, Mohammad Reza Mousavi. *Modeling and Analysis of Communicating Systems*. MIT Press, 2008.
3. Noson Yanofsky, Mirco Mannucci. *Quantum Computing for Computer Scientists*. CUP, 2008.

Complementary

1. J. C. M. Baeten, T. Basten, M. A. Reniers. *Process Algebra: Equational Theories of Communicating Processes*. CUP, 2010.
2. Robin Milner. *Communicating and Mobile Systems: The Pi Calculus*. CUP, 1999.
3. Christel Baier, Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008

Pragmatics

Assessment

- Training assignments (20% each): 4 June
- Written test (80%): 28 May

Interaction

- web: `arca.di.uminho.pt/ic-1819/`
- contact: `lsb@di.uminho.pt`

...

- Week 25 Feb $\longrightarrow$ 3 Jun
- Week 8 April $\longrightarrow$ tba