Labelled Transition Systems

Luís S. Barbosa

HASLab - INESC TEC
Universidade do Minho
Braga, Portugal

1 March, 2012

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 $\Leftrightarrow$ interaction
- behaviour $\Leftrightarrow$ a structured record of interactions

Reactive systems

Concurrency vs interaction

$$\mathrm{x} := 0;$$
$$\mathrm{x} := \mathrm{x} + 1 \mid \mathrm{x} := \mathrm{x} + 2$$

- 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

$$\downarrow s \;\Leftrightarrow\; s \in \downarrow$$

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

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

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
- image finite
- ...

Reachability

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 $s \xrightarrow{\sigma}^* s''$ and $s'' \xrightarrow{a} s'$ then $s \xrightarrow{\sigma a}^* s'$, 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 equivalence

Run

A word $\sigma \in N^*$ is a run (or a complete trace) of a state $s \in S$ iff there is another state $t \in S$, reachable from s such that $\downarrow t$

Language

The language recognized by a state $s \in S$ is the set of runs of s

Language equivalence

Two states are language equivalent if both recognize the same language

Automata

Back to old friends?

automaton behaviour $\Leftrightarrow$ accepted language

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

- $L_1 + L_2 \triangleq L_1 \cup L_2$ (union)
- $L_1 \cdot L_2 \triangleq \{st \mid s \in L_1, t \in L_2\}$ (concatenation)
- $L^* \triangleq \{\epsilon\} \cup L \cup (L \cdot L) \cup (L \cdot L \cdot L) \cup \ldots$ (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 ential 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, emphasizing their concurrent composition and continuous interaction with their environement
- To introduce techniques for (formal) specification, analysis and verification of reactive systems

The course

Syllabus

1. Part 1: Basic models for reactive systems
   1.1 Transition systems, behaviour and bisimilarity
   1.2 Introduction to process algebra
   1.3 Specification and calculus of reactive systems in CCS
   1.4 Logics for processes
2. Part 2: Reactive systems with real-time constraints
   2.1 Timed automata
   2.2 Specification, analysis and verification of reactive systems with real-time constraints
3. Part 3: Reactive systems with mobility requirements
   3.1 Mobility and interaction
   3.2 Introduction to the π-calculus
4. **Laboratory:** Practice in mCRL2