Labelled Transition Systems

Luís Soares Barbosa

Interaction & Concurrency Course Unit (Lcc)

Universidade do Minho

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 $(S, N, \longrightarrow)$ where

- $S = \{s_0, s_1, s_2, ...\}$ is a set of states
- $\longrightarrow \subseteq S \times N \times S$ is the transition relation, often given as an N-indexed family of binary relations

$$s \xrightarrow{a} s' \;\equiv\; (s, a, s') \in \longrightarrow$$

In some contexts the definition is extended with a set $\downarrow \subseteq S$ of terminating or final states and a characteristic predicate

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

Labelled Transition System

Morphism

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

$$s \xrightarrow{a} s' \quad \Rightarrow \quad h(s) \xrightarrow{a}' h(s')$$

i.e.

$$\boxed{\text{morphisms preserve transitions}}$$

... and termination, whenever applicable:

$$s \downarrow \quad \Rightarrow \quad h(s) \downarrow'$$

Labelled Transition System

System

Given a LTS $(S, N, \longrightarrow)$, each state $s \in S$ determines a system over all states reachable from s and the corresponding restrictions upon $\longrightarrow$.

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

- $s \xrightarrow{\epsilon}{}^* s$ 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 $s \xrightarrow{\sigma}{}^* t$

Automata

Back to old friends?

$$\boxed{\text{automaton behaviour} \quad \equiv \quad \text{accepted language}}$$

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 anomalous termination (eg deadlock)
- Nondeterminisim should be taken seriously: the notion of equivalence based on accepted language is blind wrt nondeterminism
- 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.

Looking for suitable notions of equivalence of behaviours

Intuition

Two LTS should be equivalent if they cannot be distinguished by interacting with them.

Equality of functional behaviour

is not preserved by parallel composition: non compositional semantics, cf,

`x:=4; x := x+1` and `x:=5`

Graph isomorphism

is too strong (why?)

Trace

Definition

Let $T = (S, N, \longrightarrow)$ be a labelled transition system. The set of traces $\mathrm{Tr}(s)$, for $s \in S$ is the minimal set satisfying

(1) $\epsilon \in \mathrm{Tr}(s)$

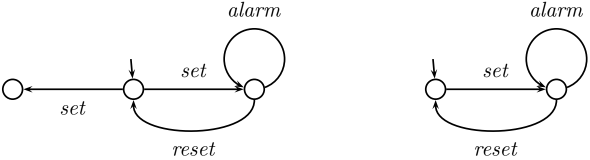
(2) $a\sigma \in \mathrm{Tr}(s) \Rightarrow (\exists\, s' \;:\; s' \in S :\; s \xrightarrow{a} s' \wedge \sigma \in \mathrm{Tr}(s'))$

Trace equivalence

Definition

Two states s, r are trace equivalent iff $\mathsf{Tr}(s) = \mathsf{Tr}(r)$
(i.e. they can perform the same finite sequences of transitions)

Example



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

Simulation

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

Simulation

A state q is simulated 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

Definition

Given $(S_1, N, \longrightarrow_1)$ and $(S_2, N, \longrightarrow_2)$ over N, relation $R \subseteq S_1 \times S_2$ is a simulation iff, for all $(p, q) \in R$ and $a \in N$,

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

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

Example

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

$$q_0 \lesssim p_0 \qquad \text{cf.} \qquad \{(q_0, p_0), (q_1, p_1), (q_4, p_1), (q_2, p_2), (q_3, p_3)\}$$

Similarity

Definition

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

Lemma

The similarity relation is a preorder
(i.e. reflexive and transitive)

Bisimulation

Definition

Given $(S_1, N, \longrightarrow_1)$ and $(S_2, N, \longrightarrow_2)$ over N, relation $R \subseteq S_1 \times S_2$ is a bisimulation iff both R and its converse R° are simulations.
I.e. whenever $(p, q) \in R$ and $a \in N$,

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

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

Bisimulation

The Game characterization

Two players R and I discuss whether the transition structures are mutually corresponding

- R starts by chosing a transition
- I replies trying to match it
- if I succeeds, R plays again
- R wins if I fails to find a corresponding match
- I wins if it replies to all moves from R and the game is in a configuration where all states have been visited or R can't move further. In this case is said that I has a wining strategy

Examples

$$\begin{tikzcd}
& q_1 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
q_2 \arrow[rr, "c"] & & q_3 \arrow[loop right, "c"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
m \arrow[d, "a"'] \\
n \arrow[loop right, "c"]
\end{tikzcd}$$

$$q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3 \xrightarrow{a} \cdots
\qquad\qquad
\begin{tikzcd}
h \arrow[loop right, "a"]
\end{tikzcd}$$

Examples

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

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

After thoughts

- Follows a $\forall, \exists$ pattern: p in all its transitions challenge q which is called to find a matchh to each of those (and conversely)
- Tighter correspondence with transitions
- Based on the information that the transitions convey, rather than on the shape of the LTS
- Local checks on states
- Lack of hierarchy on the pairs of the bisimulation (no temporal order on the checks is required)

which means bisimilarity can be used to reason about infinite or circular behaviours.

After thoughts

Compare the definition of bisimilarity with

$p == q$ if, for all $a \in N$

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

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

After thoughts

$p == q$ if, for all $a \in N$

(1) $p \downarrow_1 \; \Leftrightarrow q \downarrow_2$

(2.1) $p \xrightarrow{a}_1 p' \;\Rightarrow\; (\exists\, q' \; : \; q' \in S_2 : \; q \xrightarrow{a}_2 q' \wedge p' == q')$
(2.1) $q \xrightarrow{a}_2 q' \;\Rightarrow\; (\exists\, p' \; : \; p' \in S_1 : \; p \xrightarrow{a}_1 p' \wedge p' == q')$

- The meaning of $==$ on the pair (p, q) requires having already established the meaning of $==$ on the derivatives
- ... therefore the definition is ill-founded if the state space reachable from (p, q) is infinite or contain loops
- ... this is a local but inherently inductive definition (to revisit later)

After thoughts

Proof method

To prove that two behaviours are bisimilar, find a bisimulation containing them ...

- ... impredicative character
- coinductive vs inductive definition

Properties

Definition

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

Lemma

1. The identity relation *id* is a bisimulation
2. The empty relation $\perp$ is a bisimulation
3. The converse R° of a bisimulation is a bisimulation
4. The composition $S \cdot R$ of two bisimulations S and R is a bisimulation
5. The $\bigcup_{i \in I} R_i$ of a family of bisimulations $\{R_i \mid i \in I\}$ is a bisimulation

Properties

Lemma

The bisimilarity relation is an equivalence relation
(i.e. reflexive, symmetric and transitive)

Lemma

The class of all bisimulations between two LTS has the structure of a complete lattice, ordered by set inclusion, whose top is the bisimilarity relation $\sim$.

Properties

Lemma

In a deterministic labelled transition system, two states are bisimilar iff they are trace equivalent, i.e.,

$$s \sim s' \ \Leftrightarrow \ \mathsf{Tr}(s) = \mathsf{Tr}(s')$$

Hint: define a relation R as

$$(x, y) \in R \Leftrightarrow \mathsf{Tr}(x) = \mathsf{Tr}(y)$$

and show R is a bisimulation.

Properties

Warning

The bisimilarity relation $\sim$ is not the symmetric closure of $\lesssim$

Example

$$q_0 \lesssim p_0, \; p_0 \lesssim q_0 \quad \text{but} \quad p_0 \not\sim q_0$$

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

Notes

Similarity as the greatest simulation

$$\lesssim \;\; \widehat{=} \;\; \bigcup\{S \mid S \text{ is a simulation}\}$$

Bisimilarity as the greatest bisimulation

$$\sim \;\; \widehat{=} \;\; \bigcup\{S \mid S \text{ is a bisimulation}\}$$