Formale Spezifikation und Verifikation

Mirco Tribastone

Institut für Informatik
Ludwig-Maximilians-Universität München
tribastone@pst.ifi.lmu.de

Preliminaries

Labelled Transitions Systems

Labelled Transition Systems

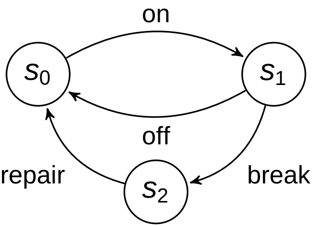


Figure: A reactive system with breakdowns and repairs

Formal Definition

A labelled transition system is a tuple $LTS = (S, A, \rightarrow, s_0)$ where

- S is a set of states
- A is a finite alphabet of actions
- $\rightarrow$ is a ternary relation $\rightarrow \in S \times A \times S$. We often write $s \xrightarrow{a} s'$ instead of $(s, a, s') \in \rightarrow$, for $s, s' \in S$ and $a \in A$
- $s_0 \in S$ is the initial state of the system

Labelled Transition Systems

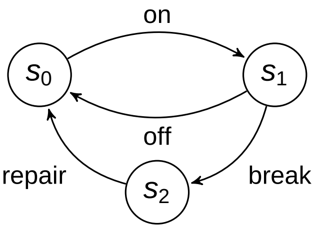


Figure: A reactive system with breakdowns and repairs

For the model in the figure, $LTS = (S, A, \rightarrow, s_0)$, with

$$S = \{s_0, s_1, s_2\},$$
$$A = \{\text{on}, \text{off}, \text{break}, \text{repair}\},$$
$$\rightarrow = \{(s_0, \text{on}, s_1), (s_1, \text{off}, s_0), (s_1, \text{break}, s_2), (s_2, \text{repair}, s_0)\} \; .$$

Labelled Transition Systems

- Sometimes, the initial state is unimportant (or unknown), hence the LTS is characterized only by the triple $(S, A, \rightarrow)$.
- Sometimes, the tuple may be defined as

$$LTS = (S, A, \rightarrow, I) \ ,$$

  where $I \subseteq S$ is a set of initial states.
- States are possible configurations of the system.
- The transition relation may be also expressed as a set of relations, one element for each action, i.e.

$$LTS = \left(S, A, \{ \xrightarrow{a} \ | \ a \in A\}\right)$$

- Transitions with distinct actions are possible between two states, e.g., $s_1 \xrightarrow{\text{off}} s_0$ and $s_1 \xrightarrow{\text{standby}} s_0$.
- Self-loops are allowed, e.g., $s_1 \xrightarrow{\text{nop}} s_1$.

Labelled Transition Systems

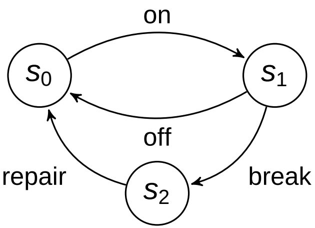


Figure: A reactive system with breakdowns and repairs

Example: breakdown and repair with *memory*. . .

Levels of Abstraction

- Activities are interpreted as being uninterruptible computations that move the system into another configuration.
- This is a very general notion that gives freedom as to which concrete tasks are to be associated with actions in the model.
- For instance, a detailed model of a communication protocol may have $\{\text{send}, \text{receive}, \text{ack}, \ldots\}$.
- A coarse-grained representation may abstract those actions with a single (uninterruptible) action called transmit.
- The former model may be used, for instance, to reason about the possibility of not receiving an acknowledgement after some data is sent.
- The latter may be used if the focus of the model is other than the actual communication mechanism.

Practical Considerations

How can we describe very large labelled transition systems?

As XML?

`<lts><ar><st>q0</st><lab>a</lab><st>q1</st></ar>...</lts>`.

As a table?

Rows and columns are labelled by states, entries are either empty or marked with a set of actions.

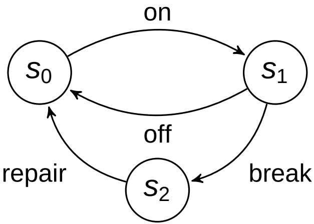
As a listing of triples?

$$\rightarrow = \{(s_0, a, s_1), (s_0, a, s_2), (s_1, b, s_3), (s_1, c, s_4), (s_2, d, s_3), (s_2, d, s_4)\}$$

As a more compact listing of triples?

$$\rightarrow = \{(s_0, a, \{s_1, s_2\}), (s_1, b, s_3),\ (s_1, c, s_4), (s_2, d, \{s_3, s_4\})\}.$$

Some Useful Definitions



Given a labelled transition system $LTS = (S, A, \rightarrow)$, let:

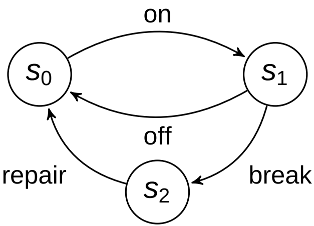
$$\mathrm{Post}(s, a) = \{s' \in S : s \xrightarrow{a} s'\}, \qquad \text{e.g., } \mathrm{Post}(s_1, \text{off}) = \{s_0\},$$

$$\mathrm{Post}(s) = \bigcup_{a \in A} \mathrm{Post}(s, a), \qquad \text{e.g., } \mathrm{Post}(s_1) = \{s_0, s_2\},$$

$$\mathrm{Pre}(s, a) = \{s' \in S : s' \xrightarrow{a} s\}, \qquad \text{e.g., } \mathrm{Pre}(s_1, \text{on}) = \{s_0\},$$

$$\mathrm{Pre}(s) = \bigcup_{a \in A} \mathrm{Pre}(s, a), \qquad \text{e.g., } \mathrm{Post}(s_0) = \{s_1, s_2\}.$$

Nondeterministic and Nonterminating Behaviour



A nondeterministic and nonterminating LTS

- $LTS = (S, A, \rightarrow)$ is called deterministic iff

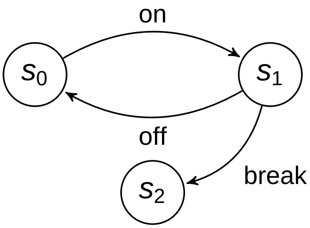
$$\left|\text{Post}(s)\right| < 2, \qquad \text{for all } s \in S.$$

- Otherwise, LTS is called nondeterministic.
- $LTS = (S, A, \rightarrow)$ is called terminating iff

$$\exists s \in S : \text{Post}(s) = \emptyset$$

- Otherwise LTS is nonterminating.

Nondeterministic and Nonterminating Behaviour



A nondeterministic and terminating LTS

- $LTS = (S, A, \rightarrow)$ is called deterministic iff

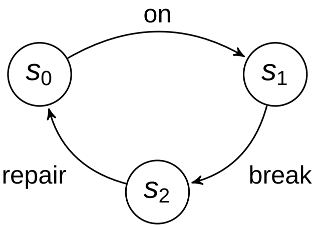
$$\left|\text{Post}(s)\right| < 2, \qquad \text{for all } s \in S.$$

- Otherwise, LTS is called nondeterministic.
- $LTS = (S, A, \rightarrow)$ is called terminating iff

$$\exists s \in S : \text{Post}(s) = \emptyset$$

- Otherwise LTS is nonterminating.

Nondeterministic and Nonterminating Behaviour



A deterministic and nonterminating LTS

- $LTS = (S, A, \rightarrow)$ is called deterministic iff

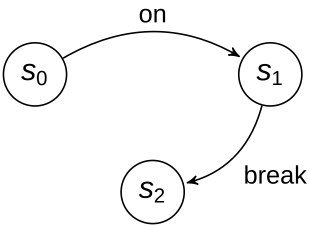
$$\left|\text{Post}(s)\right| < 2, \qquad \text{for all } s \in S.$$

- Otherwise, LTS is called nondeterministic.
- $LTS = (S, A, \rightarrow)$ is called terminating iff

$$\exists s \in S : \text{Post}(s) = \emptyset$$

- Otherwise LTS is nonterminating.

Nondeterministic and Nonterminating Behaviour



A deterministic and terminating LTS

- $LTS = (S, A, \rightarrow)$ is called deterministic iff

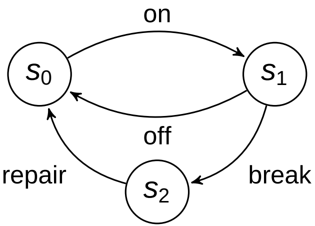
$$\left|\text{Post}(s)\right| < 2, \qquad \text{for all } s \in S.$$

- Otherwise, LTS is called nondeterministic.
- $LTS = (S, A, \rightarrow)$ is called terminating iff

$$\exists s \in S : \text{Post}(s) = \emptyset$$

- Otherwise LTS is nonterminating.

Execution Paths



A finite execution path $\pi = s_0\, a_1\, s_1\, a_2\, \cdots\, a_n\, s_n$ denotes a sequence of transitions $s_i \xrightarrow{a_{i+1}} s_{i+1}$, with $s_i \in S, 0 \leq i \leq n$ and $a_i \in A, 0 < i \leq n$.
An infinite execution path $\pi_\infty = s_0\, a_1\, s_1\, a_2\, \cdots$ denotes an infinite sequence of transitions such that $s_i \xrightarrow{a_{i+1}} s_{i+1}$ for all $i \geq 0$.

Examples

$$\pi' = s_0 \text{ on } s_1 \text{ off } s_0 \text{ on } s_1 \text{ break } s_2 \text{ repair } s_0$$

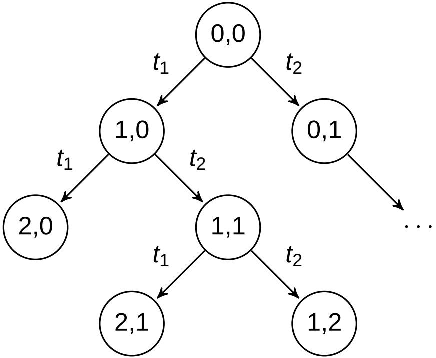
$$\pi_\infty = s_1 \text{ off } s_0 \text{ on } s_1 \text{ off } s_0 \text{ on } s_1 \text{ off } s_0 \text{ on } s_1 \text{ off } s_0 \text{ on } s_1 \cdots$$

Labelled Transitions Systems of Concurrency

$x, y \leftarrow 0$
thread 1 do
  while $x < 2$ and $y < 2$ **do**
    $x \leftarrow x + 1$
  end while
end thread

thread 2 do
  while $x < 2$ and $y < 2$ **do**
    $y \leftarrow y + 1$
  end while
end thread

Suppose that **while** blocks are atomic. What are the final values of x and y when the program terminates?



Structured Operational Semantics

Structured Operational Semantics[1]

A syntax-driven labelled transition system. In our case,

- Define the set of well-formed phrases of a language (typically using Backus-Naur form)
- Describe inference rules in the form

$$\left(\frac{\textbf{premise}}{\textbf{conclusion}}\right) \qquad \frac{E_1 \xrightarrow{a_1} E_1' \quad E_2 \xrightarrow{a_2} E_2' \quad \cdots \quad E_m \xrightarrow{a_m} E_m'}{op(E_1, E_2, \ldots, E_m) \xrightarrow{a} op(E_1', E_2', \ldots, E_m')}, \text{ where}$$

$E_1, \ldots, E_m$ are syntactically valid expressions
$a_1, \ldots, a_m, a$ are transition labels
op is an operator of the language with arity m

- An axiom is a rule in the form

$$\frac{}{E \xrightarrow{a} E'}$$

[1] G. Plotkin. A Structural Approach to Operational Semantics, *J. Log. Algebr. Program.*, 2004, (**60–61**), 17–139.

Example: Regular Expressions

Syntax of Regular Expressions

$E \ ::= \ 1 \mid a \mid E + E \mid E \cdot E \mid E^*, \quad a \in A$ and $\mu \in A \cup \{\varepsilon\}$

- Usual order for the binding strength: $^*, \ \cdot, \ +$
  For instance $a \cdot b^* + c = \big(a \cdot (b^*)\big) + c$
- Is $a \cdot b$ allowed (i.e., is it well formed)?
- Is $a \cdot b \cdot c$ well formed?
- Is $a \cdot b \cdot \varepsilon$ well formed?
- For convenience we may use $E\,F$ in lieu of $E \cdot F$.
- Sometimes it is useful to think of well-formed expressions in terms of parse trees. . .

Example: Regular Expressions

Syntax of Regular Expressions

$E \ ::= \ 1 \mid a \mid E + E \mid E \cdot E \mid E^*, \quad a \in A$ and $\mu \in A \cup \{\varepsilon\}$

Operational Semantics of Regular Expressions

(Tic) $\dfrac{}{1 \xrightarrow{\varepsilon} 1}$ (Atom) $\dfrac{}{a \xrightarrow{a} 1}$

(Sum$_1$) $\dfrac{e \xrightarrow{\mu} e'}{e + f \xrightarrow{\mu} e'}$ (Sum$_2$) $\dfrac{f \xrightarrow{\mu} f'}{e + f \xrightarrow{\mu} f'}$

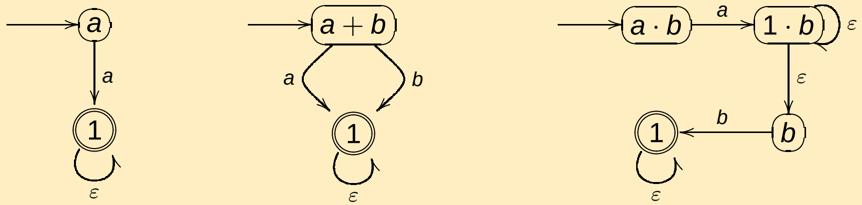
(Seq$_1$) $\dfrac{e \xrightarrow{\mu} e'}{e \cdot f \xrightarrow{\mu} e' \cdot f}$ (Seq$_2$) $\dfrac{e \xrightarrow{\varepsilon} 1}{e \cdot f \xrightarrow{\varepsilon} f}$

(Star$_1$) $\dfrac{}{e^* \xrightarrow{\varepsilon} 1}$ (Star$_2$) $\dfrac{e \xrightarrow{\mu} e'}{e^* \xrightarrow{\mu} e' \cdot e^*}$

The Automaton Associated with a Regular Expression

The SOS inference rules implicitly define a particular automaton for each regular expression e:

- the initial state is e (we shall often omit to mark it)
- the set of labels is $A \cup \{\varepsilon\}$
- the set of states consists of all r.e. that can be reached starting from e via a sequence of transitions
- the transition relation is the one induced from the SOS inference rules
- the only final state is 1 (we shall often omit to mark it)



Sequences of Transitions

$e \xrightarrow{s} e'$

Let $s = \mu_1\mu_2 \cdots \mu_n$ be the string obtained as the concatenation of $\mu_1, \mu_2, \ldots, \mu_n \in A \cup \{\varepsilon\}$ (remind that ε behaves as the empty string).

We write $e \xrightarrow{s} e'$ if there exist $e_1, e_2, \ldots, e_{n-1}$ such that:

$$e \xrightarrow{\mu_1} e_1 \xrightarrow{\mu_2} e_2 \cdots e_{n-1} \xrightarrow{\mu_n} e'$$

Example: $a \cdot b \cdot c \xrightarrow{abc} 1$

We have $abc = a\varepsilon\varepsilon\varepsilon b\varepsilon c$ and:

$$a \cdot b \cdot c \xrightarrow{a} 1 \cdot b \cdot c \xrightarrow{\varepsilon} 1 \cdot b \cdot c \xrightarrow{\varepsilon} 1 \cdot b \cdot c \xrightarrow{\varepsilon} b \cdot c \xrightarrow{b} 1 \cdot c \xrightarrow{\varepsilon} c \xrightarrow{c} 1$$

A Few Examples of Regular Expressions

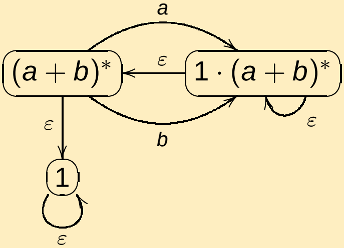
$(a+b)^* \xrightarrow{a} 1 \cdot (a+b)^*$

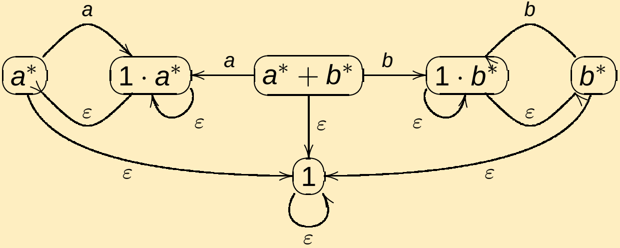
$$\dfrac{\dfrac{\dfrac{}{a \xrightarrow{a} 1}\,(Atom)}{a+b \xrightarrow{a} 1}\,(Sum_1)}{(a+b)^* \xrightarrow{a} 1 \cdot (a+b)^*}\,(Star_2)$$

$1 \cdot (a+b)^* \xrightarrow{\varepsilon} (a+b)^*$

$$\dfrac{\dfrac{}{1 \xrightarrow{\varepsilon} 1}\,(Tic)}{1 \cdot (a+b)^* \xrightarrow{\varepsilon} (a+b)^*}\,(Seq_2)$$

LTS Fragments for $(a+b)^*$ and $a^* + b^*$





Another Example on Regular Expressions

$$(a^* + b^*)^* \xrightarrow{b} 1 \cdot b^* \cdot (a^* + b^*)^*$$

$$\cfrac{\cfrac{\cfrac{\cfrac{}{b \xrightarrow{b} 1}\ (Atom)}{b^* \xrightarrow{b} 1 \cdot b^*}\ (Star_2)}{a^* + b^* \xrightarrow{b} 1 \cdot b^*}\ (Sum_2)}{(a^* + b^*)^* \xrightarrow{b} 1 \cdot b^* \cdot (a^* + b^*)^*}\ (Star_2)$$

LTS Fragment for $(a^* + b^*)^*$