A Theory of Timed Automata....Kaboom!

Speaker: Sandeep Chintabathina

Papers used

- *A Theory of Timed Automata* **(1994) Rajeev Alur and David Dill**
- *Timed Automata: Semantics, Algorithms and Tools* **(2004) Johan Bengtsson and Wang Yi**

Talk outline

- **Introduction**
- ω**-automata**
- **Timed automata**
- **Timed regular languages**
- **Verification using timed automata**

Introduction

- **The goal of this research is to use automata for the specification and verification of systems.**
- **When reasoning about systems it is possible to abstract away from time and retain only the sequence of events - *qualitative temporal reasoning*.**
- **A sequence of events (trace) describes a valid behavior of the system.**
- **A set of such event sequences constitutes all valid behaviors of the system.**
- **Since the set of sequences is a formal language, we can use automata theory for specification and verification of systems.**

Introduction

- **Finite automata and a variety of competing formalisms are capable of manipulating and analyzing system behavior.**
- **In particular we will look at ω-automata because it is capable of describing traces that are infinite.**
- **But these formalisms are limited to qualitative reasoning only.**
- **When reasoning about systems such as airplane control systems or toasters correct functioning depends on real time considerations - quantitative reasoning is needed.**

Objective : Specify and verify real-time systems by modifying finite automata

Outcome : A theory of timed automata

Introduction

- **Timing information can be added to an event trace by pairing it with a sequence of times.**
- **The i'th element of time sequence gives the time of occurrence of i'th event in the event sequence.**
- **Fundamental question: what is the nature of time?**
- ***Discrete-time* model and *Dense-time* model.**

Introduction

- **Discrete-time model requires the time sequence to be monotonically increasing sequence of integers.**
- **It is possible to reduce a timed trace into a untimed trace.**
- **Timed trace (e1:1),(e2:4),(e3:6)..... can be reduced to the untimed trace e1,silent,silent,e2,silent,e6......**
- **The time of each event is same as its position.**
- **This behavior can be modeled using finite automata.**

Introduction

Drawbacks of discrete model:

- **Events do not always take place at integer-valued times.**
- **Continuous time must be approximated limiting the accuracy with which systems are modeled.**

Introduction

- **Dense-time model is a more natural model for physical processes operating over continuous time.**
- **Times of events are real numbers which increase monotonically without bounds.**
- **Cannot use finite automata because it is not obvious how to transform dense-time traces into untimed traces.**
- **For this reason a theory of timed languages and timed automata was developed.**

Introduction

Timed automata can capture interesting aspects of real-time systems:

- **qualitative features - liveness, fairness, nondeterminism.**
- **quantitative features - periodicity, bounded response, timing delays.**

ω-automata

- ω-language consists of infinite words.
- ω-language over a finite alphabet Σ is a subset of Σ^ω - the set of all infinite words over Σ.
- ω-automata provides a finite representation for ω-languages.
- It is a nondeterministic finite automata with acceptance condition modified to handle infinite input words.
- We will consider a type of ω-automata called *Buchi* automata.

Transition table

- **A *transition table* A is a $\langle \Sigma, S, S_0, E \rangle$ where Σ is a set of input symbols, S is a finite set of states, $S_0 \subseteq S$ is a set of start states and $E \subseteq S \times S \times \Sigma$ is a set of edges.**
- **If $\langle s, s', a \rangle \in E$ then automaton can change state from s to s' reading the input symbol a.**

Run of A

- **For a word $\sigma = \sigma_1\sigma_2...$ over alphabet Σ, we say that**

$$r : s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} s_2 \xrightarrow{\sigma_3} ...$$

  is a *run* of A over σ provided $s_0 \in S_0$ and $\langle s_{i-1}, s_i, \sigma_i \rangle \in E$ for all $i \geq 1$.

- **For such a run the set $inf(r)$ consists of states $s \in S$ such that $s = s_i$ for infinitely many $i \geq 0$.**

Buchi automaton

- **A *Buchi automaton* A is a transition table $\langle \Sigma, S, S_0, E \rangle$ along with additional set $F \subseteq S$ of accepting states.**
- **A run of A over a word σ is an *accepting run* iff** $inf(r) \cap F \neq \emptyset$
- **The language $L(A)$ accepted by the A is**

$$L(A) = \{\sigma \mid \sigma \in \Sigma^\omega \wedge \textbf{A has an accepting run over } \sigma\}$$

Buchi Automaton

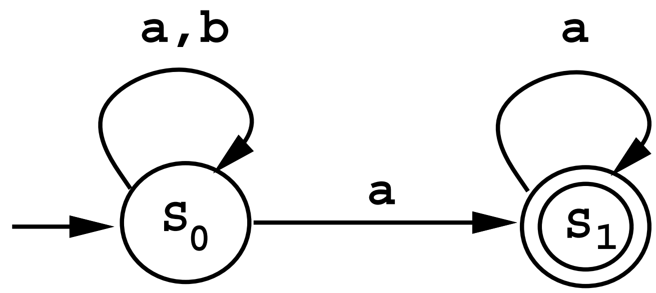

Figure 1: Büchi automaton accepting $(a+b)^*a^\omega$

Properties of Buchi Automata

- **An ω-language is called ω-regular iff it is accepted by some Buchi automaton.**
- **The class of ω-regular languages are closed under all boolean operations.**
- **If Buchi automaton is used for modeling finite state concurrent systems, the verification problem reduces to that of language inclusion. But it leads to exponential blow-up in number of states.**
- **However, the inclusion problem for deterministic automaton takes only polynomial time.**
- **The class of languages accepted by deterministic Buchi automaton is strictly smaller than the class of ω-regular languages.**

Timed languages

- **A timed word is formed by coupling a real-valued time with each symbol in a word.**
- **The behavior of a real-time system corresponds to a timed word over the alphabet of events.**
- **A *time sequence* $\tau = \tau_1\tau_2...$ is an infinite sequence of time values $\tau_i \in R$ with $\tau_i > 0$, satisfying the constraints:**
  - ***Monotonocity* : τ increases strictly monotonically $\tau_i < \tau_{i+1}$ for all $i \geq 1$.**
  - ***Progress* : For every $t \in R$, there is some $i \geq 1$ such that $\tau_i > t$**

Timed languages

- **A *timed word* over an alphabet Σ is a pair (σ, τ) where $\sigma = \sigma_1\sigma_2\ldots$ is an infinite word over Σ and τ is a time sequence.**
- **A *timed language* over Σ is a set of timed words over Σ.**
- **Example : Let $\Sigma = \{a, b\}$ and language L_1 consists of all timed words (σ, τ) such that there is no b after time 5.6**

$$L_1 = \{(\sigma, \tau) \mid \forall i.((\tau_i > 5.6) \rightarrow (\sigma_i = a))\}$$

Given timed language L over Σ

$$Untime(L) = \{\sigma \mid \sigma \in \Sigma^\omega \wedge (\sigma, \tau) \in L\}$$

$Untime(L_1) = (a + b)^* a^\omega$

Timed Transition tables

- **They are extension of transition tables to read timed words.**
- **In this table, a transition depends upon the input symbol as well as the time of the input symbol relative to the times of previously read symbols.**
- **For this reason, a finite set of (real valued) clocks are associated with each table.**
- **The set of clocks can be viewed as set of stop-watches that can be reset and checked independently of one another, but all of them refer to the same clock.**
- **A clock constraint is associated with each transition and only when the current clock values satisfy this constraint will a transition be taken.**

Example timed transition table

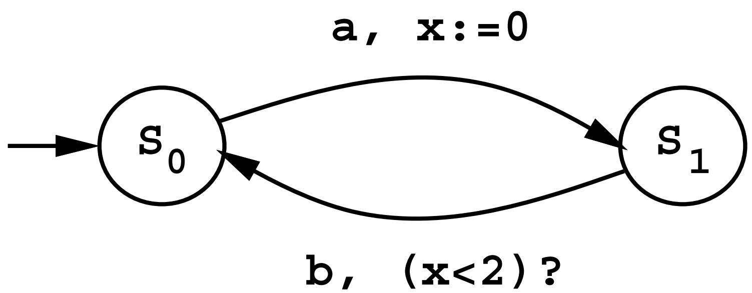

Figure 3: Example of a timed transition table

Example timed transition table

The timing constraint expressed by the transition table is that the delay between a and the following b is always less than 2; more formally the language is

$$\{((ab)^\omega, \tau) \mid \forall i.(\tau_{2i} < \tau_{2i-1} + 2)\}$$

Timed transition table with two clocks

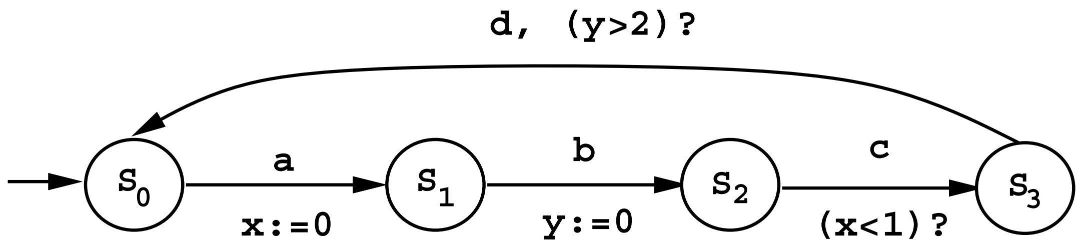

Figure 4: Timed transition table with 2 clocks

Timed transition table with 2 clocks

The table uses two clocks and accepts the language

$$\{((abcd)^\omega, \tau) \mid \forall j.((\tau_{4j+3} < \tau_{4j+1} + 1) \wedge (\tau_{4j+4} > \tau_{4j+2} + 2))\}$$

The clock constraints ensure that the time delay between c **and preceding** a **is less than 1 and the time delay between** d **and preceding** b **is greater than 2.**

Clock constraints and clock interpretations

- **For a set X of clock variables, the set $\Phi(X)$ of clock constraints δ is defined inductively by**

$$\delta := x \leq c \mid c \leq x \mid \neg\delta \mid \delta_1 \wedge \delta_2$$

where $x \in X$ and $c \in R$.

- **Constraints such as $true$, $x = c$, $x \in [2, 5)$ are considered abbreviations.**
- **A *clock interpretation* v for a set X of clocks is a mapping from X to R.**
- **Clock interpretation v for X *satisfies* a clock constraint δ iff δ evaluates to true using the value given by v.**

Clock interpretations

Here we introduce some notation

- **For** $t \in R$**,** $v + t$ **denotes the clock interpretation which maps every clock** x **to the value** $v(x) + t$
- **For** $Y \subseteq X$**,** $[Y \to t]v$ **denotes the clock interpretation for** X **which assigns** t **to each** $x \in Y$**, and agrees with** v **over the rest of the clocks.**

Timed Transition tables

A timed transition table A is a $\langle \Sigma, S, S_0, C, E \rangle$, where

- Σ **is a finite alphabet**
- S **is a finite set of states**
- $S_0 \subseteq S$ **is a set of start states**
- **C is a finite set of clocks**
- $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ **gives the set of transitions.**

An edge $\langle s, s', a, \lambda, \delta \rangle$ represents transition from state s to s' on input symbol a. λ is the set of clocks that will be reset and δ is a clock constraint.

Run of a timed transition table

We will define the transitions of a timed table by defining runs.

A run r, denoted by $\langle \bar{s}, \bar{v} \rangle$, of a timed transition table $\langle \Sigma, S, S_0, C, E \rangle$ over a timed word (σ, τ) is an infinite sequence of the form

$$r : \langle s_0, v_0 \rangle \xrightarrow[\tau_1]{\sigma_1} \langle s_1, v_1 \rangle \xrightarrow[\tau_2]{\sigma_2} \langle s_2, v_2 \rangle \xrightarrow[\tau_3]{\sigma_3} \ldots$$

with $s_i \in S$ and $v_i \in [C \to R]$, for all $i \geq 0$, satisfying the requirements

- ***Initiation*: $s_0 \in S_0$, and $v_0(x) = 0$ for all $x \in C$.**
- ***Consecution*: for all $i \geq 1$, there is an edge in E of the form $\langle s_{i-1}, s_i, \sigma_i, \lambda_i, \delta_i \rangle$ such that $v_{i-1} + \tau_i - \tau_{i-1}$ satisfies δ_i and v_i equals $[\lambda_i \to 0](v_{i-1} + \tau_i - \tau_{i-1})$.**

Timed transition table with two clocks

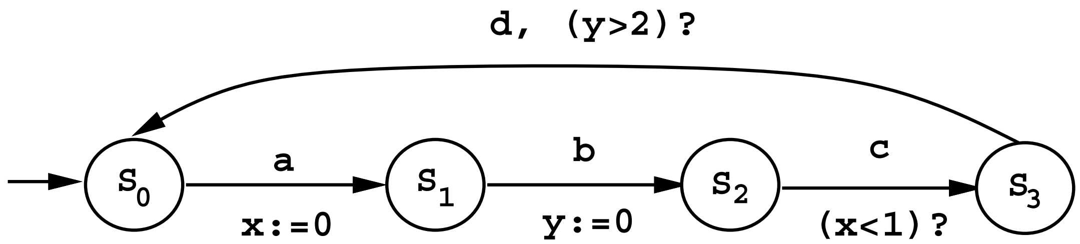

Figure 4: Timed transition table with 2 clocks

Example run

Consider a timed word corresponding to example shown above.

$$(a, 2), (b, 2.7), (c, 2.8), (d, 5), \ldots$$

An initial segment of the run is as follows. The clock interpretation is represented by listing values $[x, y]$.

$$\langle s_0, [0,0]\rangle \xrightarrow[2]{a} \langle s_1, [0,2]\rangle \xrightarrow[2.7]{b} \langle s_2, [0.7,0]\rangle \xrightarrow[2.8]{c} \langle s_3, [0.8,0.1]\rangle \xrightarrow[5]{d} \langle s_0, [3,2.3]\rangle$$

The set $inf(r)$ is the set of all $s \in S$ such that $s = s_i$ for infinitely many $i \geq 0$.

Timed regular languages

- **A *timed Buchi automaton* (TBA) is a tuple $\langle \Sigma, S, S_0, C, E, F \rangle$, where $\langle \Sigma, S, S_0, C, E \rangle$ is a timed transition table and $F \subseteq S$ is set of accepting states.**
- **A run $r = (\bar{s}, \bar{v})$ of a TBA over timed word (σ, τ) is called an accepting run iff $inf(r) \cap F \neq \emptyset$.**
- **The language $L(A)$ of timed words accepted by A is the set**

$$\{(\sigma, \tau) \mid \textbf{A has an accepting run over } (\sigma, \tau)\}$$

The class of timed languages accepted by TBA are called *timed regular languages*.

Example of a Timed automaton

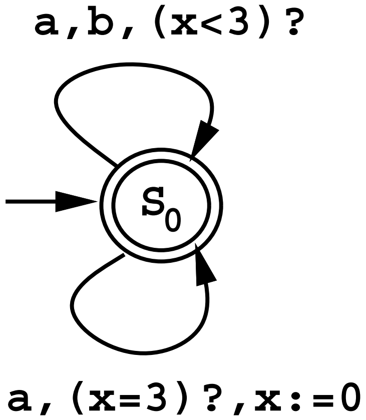

Figure 6: Timed automaton specifying periodic behavior

Example of a Timed automaton

The automaton shown above accepts the following language over the alphabet $\{a, b\}$.

$$\{(\sigma, \tau) \mid \forall i . \exists j (\tau_j = 3i \wedge \sigma_j = a)\}$$

The automaton requires that whenever clock equals 3 there is an a symbol. Therefore a happens at all time values that are multiples of 3.

Verification example

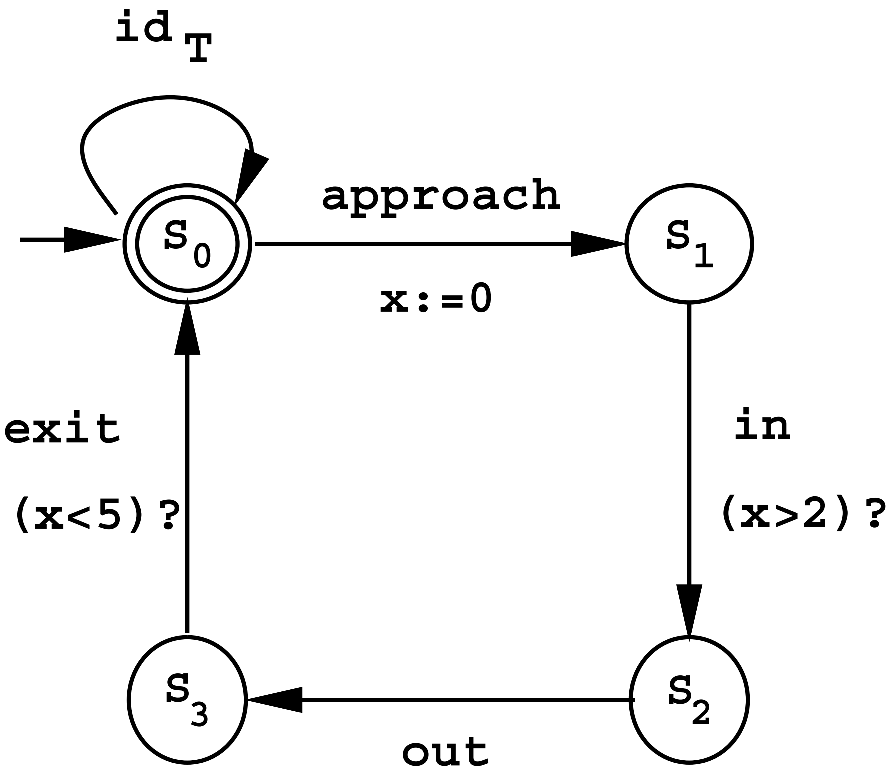

Figure 16: TRAIN

Verification example

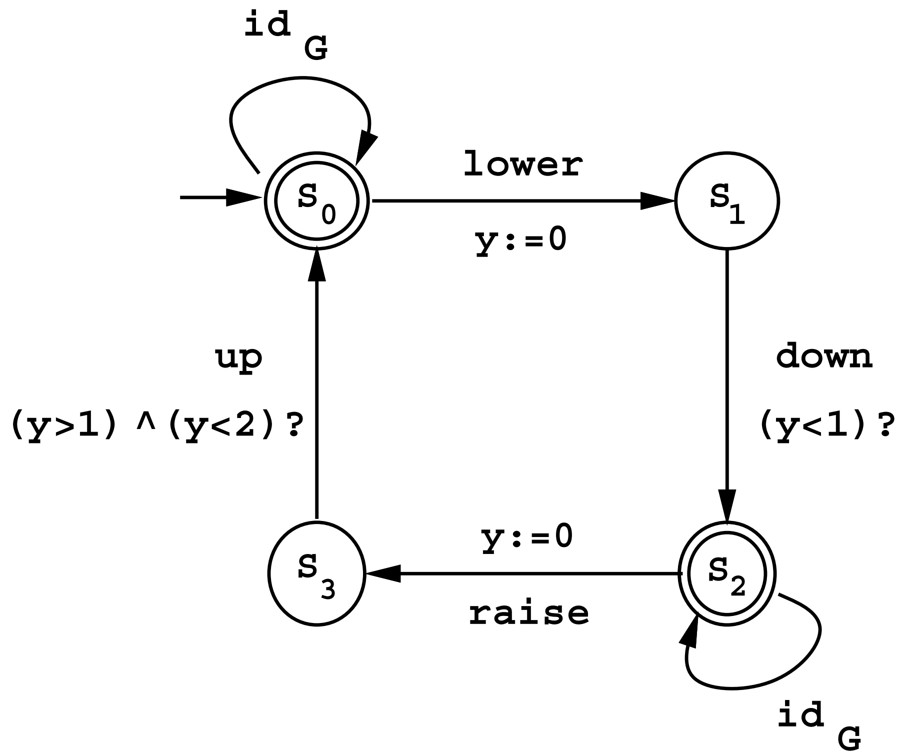

Figure 17: GATE

Verification example

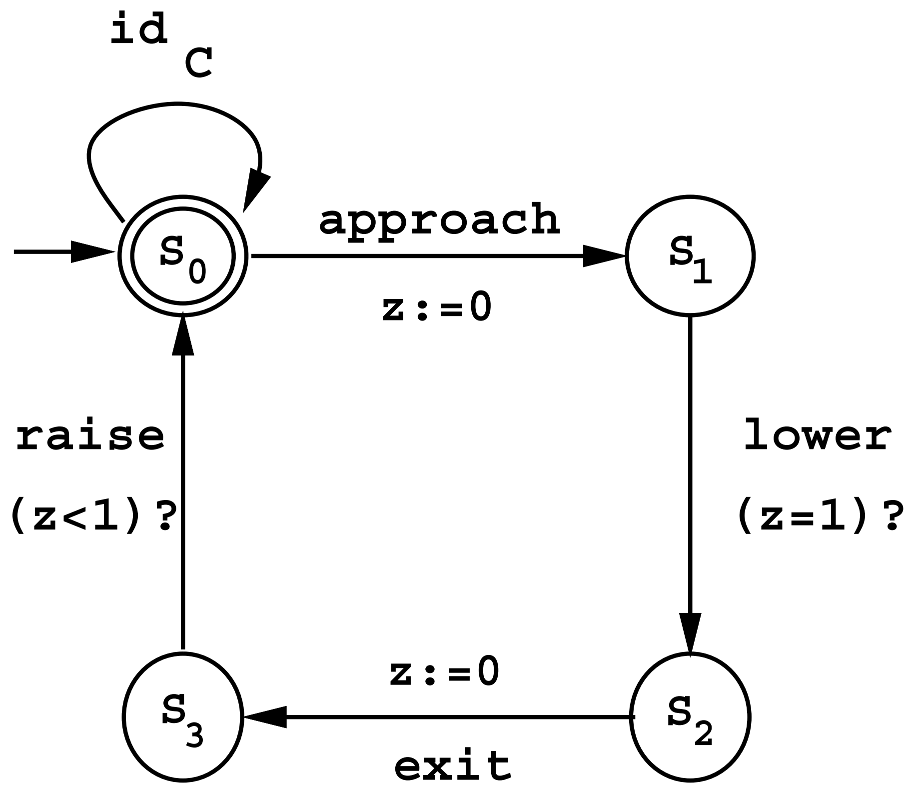

Figure 18: CONTROLLER

Correctness requirements

Implementation of the system is [TRAIN || GATE || CONTROLLER]

Specification of the system:

- **Safety: Whenever the train in inside the gate, the gate should be closed.**
- **Liveness : The gate is never closed at a stretch for more than 10 minutes.**

Correctness requirements

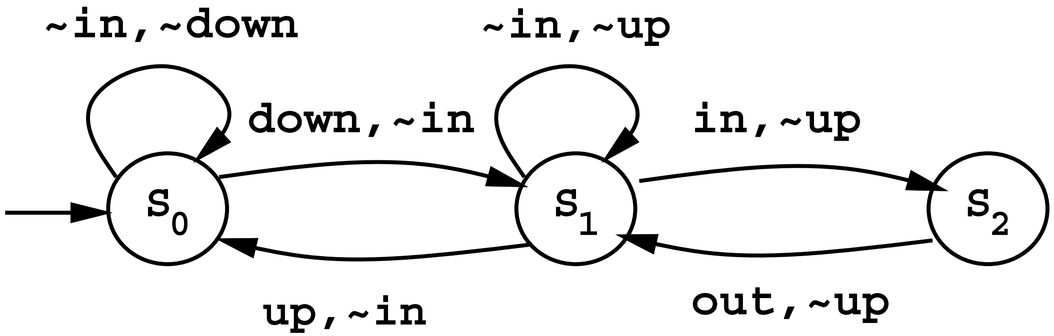

Figure 19: Safety property

Correctness requirements

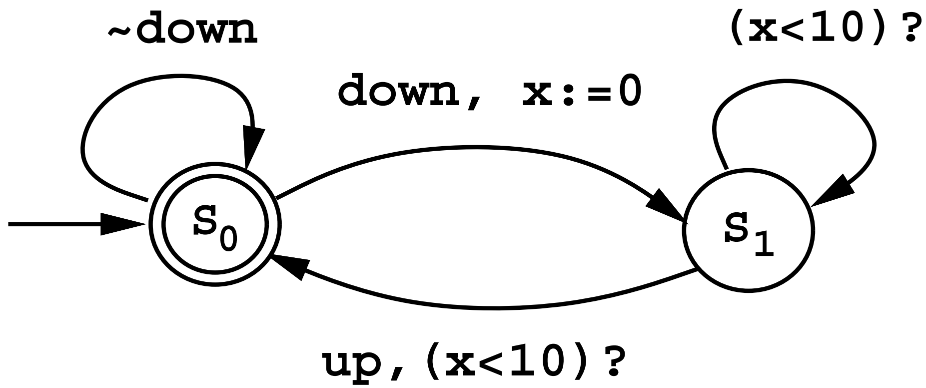

Figure 20: Real-time liveness property