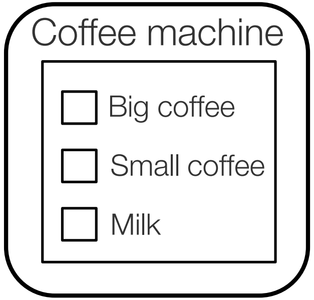
Semantics of Clocks in Timed Automata

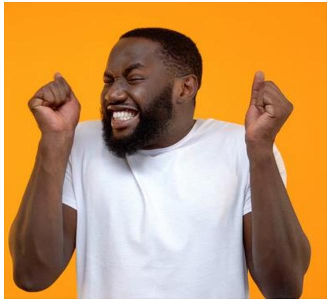
KIT junior professorship interview, lecture excerpt

29.03.2022

Ana Petrovska
Technical University of Munich
Department of Informatics
ana.petrovska@tum.de

How do we know we get the right amount of coffee?





TIMED AUTOMATA.

Finite-state automaton

Summary and Limitations

Overview of finite-state automaton

A **finite-state machine** (FSM) or **finite-state automaton** (*pl.* automata), or simply automaton is a mathematical model of computation.

- αὐτόματος, which means "self-acting, self-willed, self-moving".

An automaton is a behaviour model and automata theory provides foundation for model checking.

Overview of finite-state automaton

A **finite-state machine** (FSM) or **finite-state automaton** (*pl.* automata), or simply automaton is a mathematical model of computation.

- αὐτόματος, which means "self-acting, self-willed, self-moving".

An automaton is a behaviour model and automata theory provides foundation for model checking.

Model checking tries to prove if a program is correct or finds a property that is violated by the program, by exploring the full state space of the program and check if the states satisfy or violate the specifications.

Overview of finite-state automaton

A **finite-state machine** (FSM) or **finite-state automaton** (*pl.* automata), or simply automaton is a mathematical model of computation.

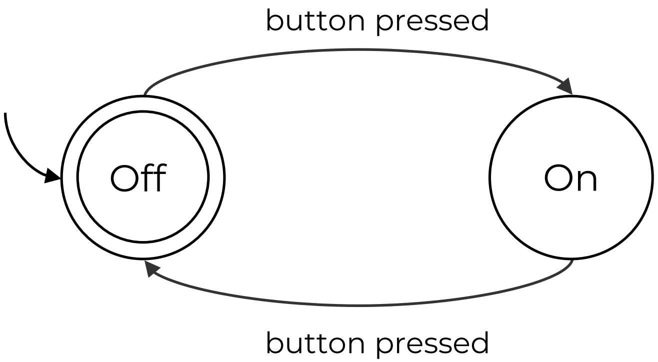
- αὐτόματος, which means "self-acting, self-willed, self-moving".

An automaton is a behaviour model and automata theory provides foundation for model checking.

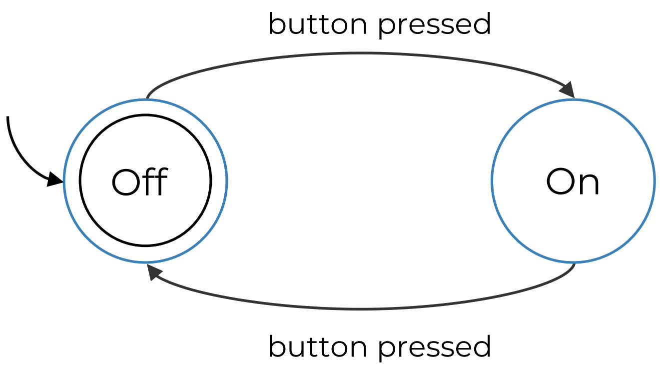
Model checking tries to prove if a program is correct or finds a property that is violated by the program, by exploring the full state space of the program and check if the states satisfy or violate the specifications.

Automata / state machines are used to model the components of the system, which later helps us to verify if a system violates a specification or not.

Overview of finite-state automaton

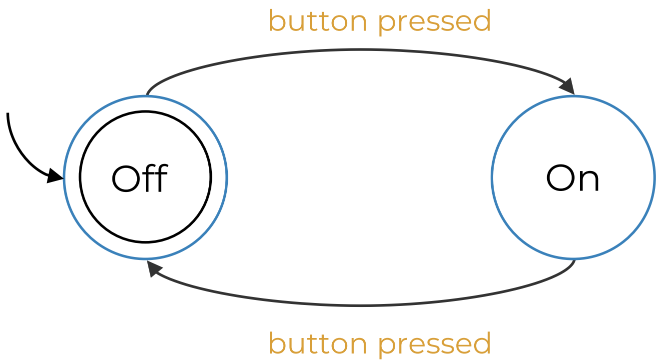


Overview of finite-state automaton



Finite set of states

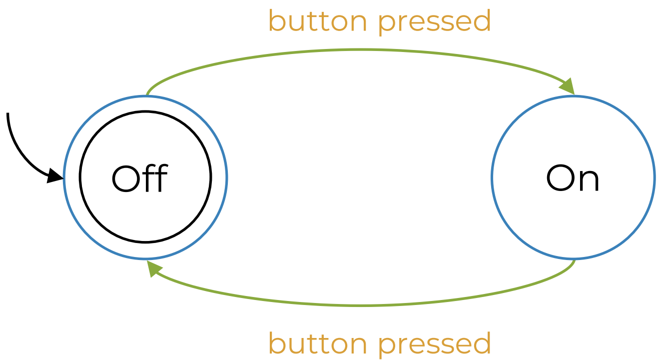
Overview of finite-state automaton



Finite set of states

Input

Overview of finite-state automaton



Finite set of states

Input

Transitions

Formal definition of finite-state automaton

Definition: A deterministic finite-set machine is a quintuple $S = (\Sigma, S, s_0, \delta, F)$, where:

- Σ is the input alphabet
- S is a finite non-empty set of states
- $s_0 \in S$ is an initial state
- δ is the state-transition function $\delta: S \times \Sigma \rightarrow S$
- F is the set of final states.

Limitations of finite-state automaton

Timing issues are of a crucial importance for many systems, especially real-time systems.

The need emerges to extend the classical automaton with some timing capabilities in order to be able to model and reason about time constraints.

Timing issues can be modelled:

- in a discrete time domain, and
- in a continuous time domain.

Limitations of finite-state automaton

Timing issues are of a crucial importance for many systems, especially real-time systems.

The need emerges to extend the classical automaton with some timing capabilities in order to be able to model and reason about time constraints.

Timing issues can be modelled:

- in a discrete time domain, and
- in a continuous time domain.

In a discrete time domain time has a discrete nature, i.e., time is advanced by discrete steps

- time is modelled by naturals; actions can only happen at natural time values

  + conceptual simplicity

  - inadequate for asynchronous systems. e.g., distributed systems.

Limitations of finite-state automaton

Timing issues are of a crucial importance for many systems, especially real-time systems.

The need emerges to extend the classical automaton with some timing capabilities in order to be able to model and reason about time constraints.

Timing issues can be modelled:

- in a discrete time domain, and
- in a **continuous time domain**.

In a discrete time domain time has a discrete nature, i.e., time is advanced by discrete steps

- time is modelled by naturals; actions can only happen at natural time values

  + conceptual simplicity

  - inadequate for asynchronous systems. e.g., distributed systems.

In a continuous time-domain time is continuous, and state changes can happen at any point in time

- time is modelled with real time values.

Timed Automata

Timed automata

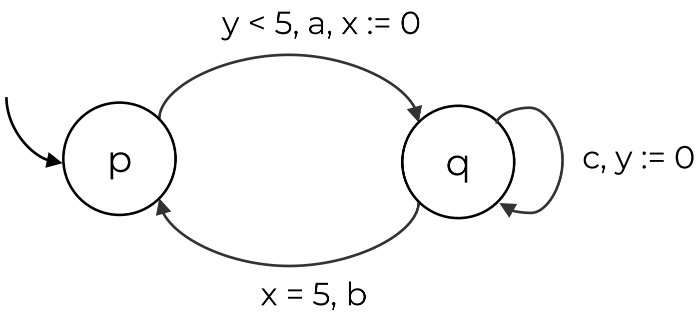
Timed automata are extension of finite-state machines with real-valued variables, called **clocks**

- clocks handle time which is continuous
- clock values can be tested
  - comparison of a clock with constants in invariants and guards
- clock values can be reset
  - only a reset to value 0 is allowed
  - clocks can be reset at the transitions
- multiple clocks can be used
  - with a uniform flow of time (all clocks have the same rate/run at the same speeds)
- the automaton spends time only in locations, not in edges

Timed automata

Timed automata are extension of finite-state machines with real-valued variables, called clocks

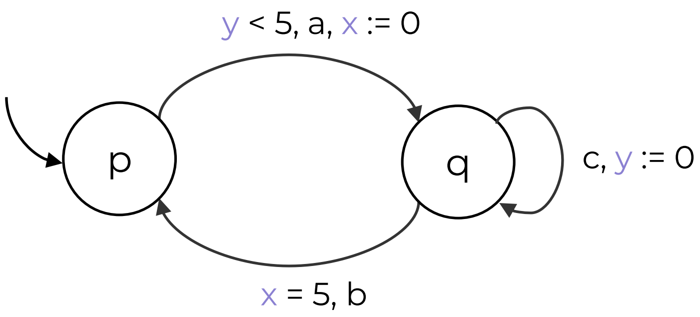
- clocks handle time which is continuous
- clock values can be tested
    - comparison of a clock with constants in invariants and guards
- clock values can be reset
    - only a reset to value 0 is allowed
    - clocks can be reset at the transitions
- multiple clocks can be used
    - with a uniform flow of time (all clocks have the same rate/run at the same speeds)
- the automaton spends time only in locations, not in edges



Timed automata

Timed automata are extension of finite-state machines with real-valued variables, called clocks

- clocks handle time which is continuous
- clock values can be tested
  - comparison of a clock with constants in invariants and guards
- clock values can be reset
  - only a reset to value 0 is allowed
  - clocks can be reset at the transitions
- multiple clocks can be used
  - with a uniform flow of time (all clocks have the same rate/run at the same speeds)
- the automaton spends time only in locations, not in edges

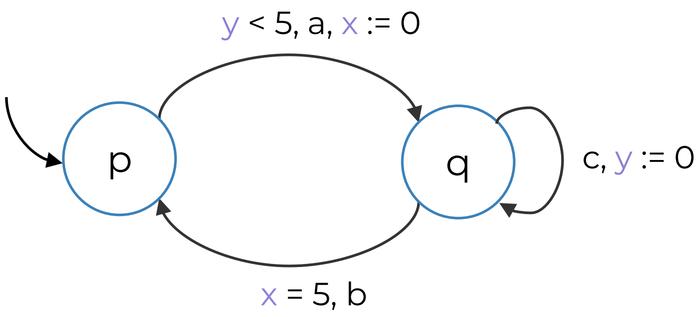


Clocks: x, y

Timed automata

Timed automata are extension of finite-state machines with real-valued variables, called clocks

- clocks handle time which is continuous
- clock values can be tested
  - comparison of a clock with constants in invariants and guards
- clock values can be reset
  - only a reset to value 0 is allowed
  - clocks can be reset at the transitions
- multiple clocks can be used
  - with a uniform flow of time (all clocks have the same rate/run at the same speeds)
- the automaton spends time only in locations, not in edges



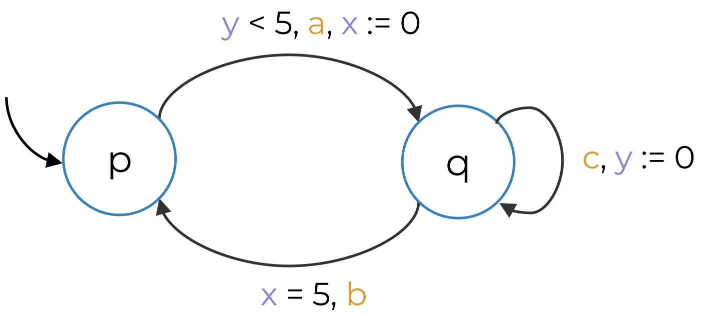
Clocks: x, y

Locations: p, q (not states!)

Timed automata

Timed automata are extension of finite-state machines with real-valued variables, called clocks

- clocks handle time which is continuous
- clock values can be tested
  - comparison of a clock with constants in invariants and guards
- clock values can be reset
  - only a reset to value 0 is allowed
  - clocks can be reset at the transitions
- multiple clocks can be used
  - with a uniform flow of time (all clocks have the same rate/run at the same speeds)
- the automaton spends time only in locations, not in edges



Clocks: x, y

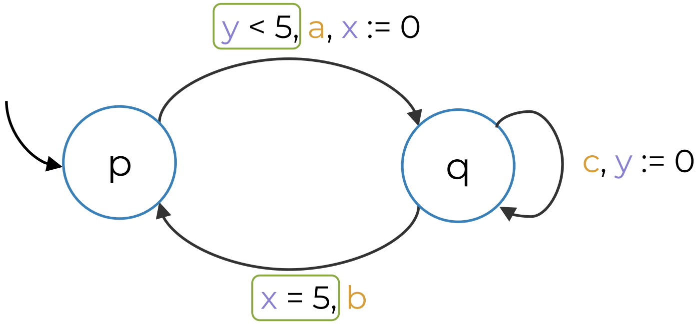
Locations: p, q (not states!)

Actions: a, b, c

Timed automata

Timed automata are extension of finite-state machines with real-valued variables, called clocks

- clocks handle time which is continuous
- clock values can be tested
  - comparison of a clock with constants in invariants and guards
- clock values can be reset
  - only a reset to value 0 is allowed
  - clocks can be reset at the transitions
- multiple clocks can be used
  - with a uniform flow of time (all clocks have the same rate/run at the same speeds)
- the automaton spends time only in locations, not in edges



Clocks: x, y

Locations: p, q (not states!)

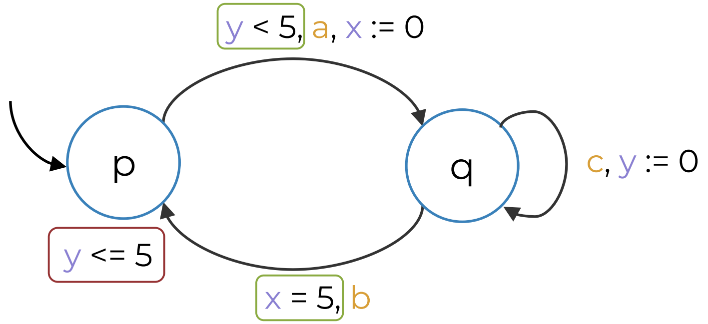
Actions: a, b, c

Transition guards: properties to be verified to enable a transition

Timed automata

Timed automata are extension of finite-state machines with real-valued variables, called clocks

- clocks handle time which is continuous
- clock values can be tested
  - comparison of a clock with constants in invariants and guards
- clock values can be reset
  - only a reset to value 0 is allowed
  - clocks can be reset at the transitions
- multiple clocks can be used
  - with a uniform flow of time (all clocks have the same rate/run at the same speeds)
- the automaton spends time only in locations, not in edges



Clocks: x, y

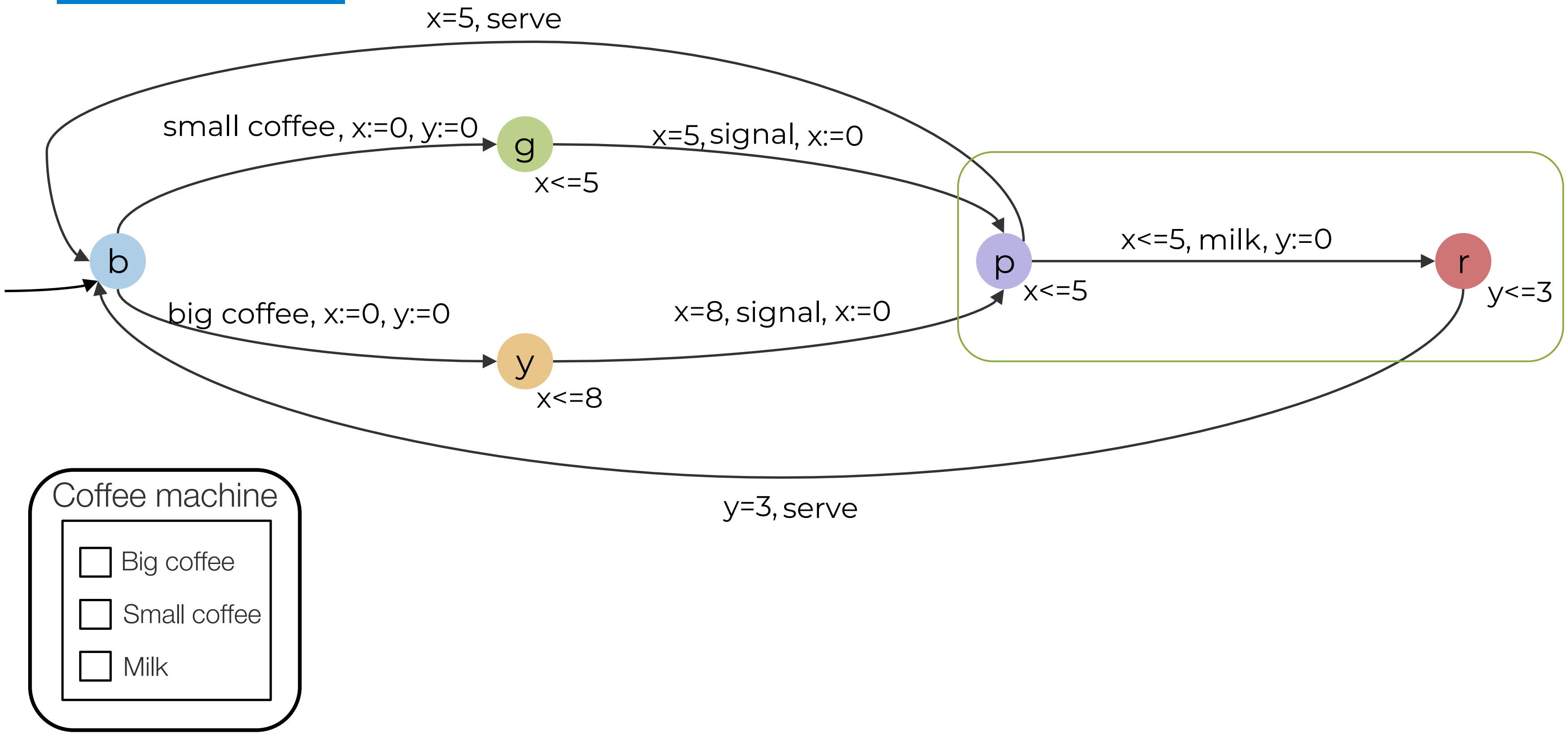
Locations: p, q (not states!)

Actions: a, b, c

Transition guards: properties to be verified to enable a transition

Invariants: properties to be verified to stay at a location.

Timed automata - Example



Formal definition of timed automaton

Definition: A timed automaton is a 4-tuple: $\mathrm{A} = (L, X, I_0, E)$, where:

- L is a finite set of locations
- X is a finite set of clocks
- $I_0 \in L$ is an initial location
- $E \subseteq \mathrm{L} \times \mathrm{C(X)} \times 2^X \times L$ is a set of edges

Edge = (source location, clock constraint, set of clocks to be reset, target location).

Clock valuation

Clock valuation v is a function $v: C \rightarrow \mathbb{R}^+$

- $v + \delta$ is a clock valuation for any $\delta \in \mathbb{R}^+$ defined as:
  $(v + \delta)(c) = v(c) + \delta$ for all $c \in C$

- $v[Y := 0]$ is a clock valuation for any $Y \subseteq C$, which resets the clocks from Y and it is defined as:
  $$v[Y := 0](c) = \begin{cases} 0 & if\ c \in Y \\ v(c) & otherwise \end{cases}$$

- $v \vDash g$ means that the valuation v satisfies the constraint g

Evaluation of clock constraint $(v \vDash g)$:

- $v \vDash c < k$ iff $v(c) < k$
- $v \vDash c \leq k$ iff $v(c) \leq k$
- $v \vDash g_1 \wedge g_2$ iff $v \vDash g_1$ and $v \vDash g_2$

Semantics of timed automaton

Definition: A timed automaton A is defined as a transition system $S_A = (S, s_0, \rightarrow)$, where:

- the state s is a pair (l, v) where l is a location and v is a clock valuation, $s \in \mathrm{S}, \mathrm{S} \subseteq \mathrm{L} \times (C \rightarrow \mathbb{R}^+)$
- $s_0 = (l_0, v)$ is the initial state if l_0 is the initial location and $v_0(c) = 0$ for all $\mathrm{c} \in C$
- transition relation $\rightarrow \subseteq S \times S$ is defined as:
  1. Time transitions (delay actions): $(l, v) \xrightarrow{\delta} (l, v + \delta)$
  2. Location switch transitions (discrete actions): $(l, v) \rightarrow (l', v')$ iff there exists $(l, g, Y, l') \in E$ such that $v \vDash g$, $v' = v[Y \coloneqq 0]$.

Clock valuation - Examples

Let $v = (x = 0.3, y = 1.4, z = 1)$

- What is $v[y \coloneqq 0]$?
- What is $v + 3.4$?
- Does $v \vDash y < 3$?
- Does $v \vDash x < 1 \wedge z > 2$?

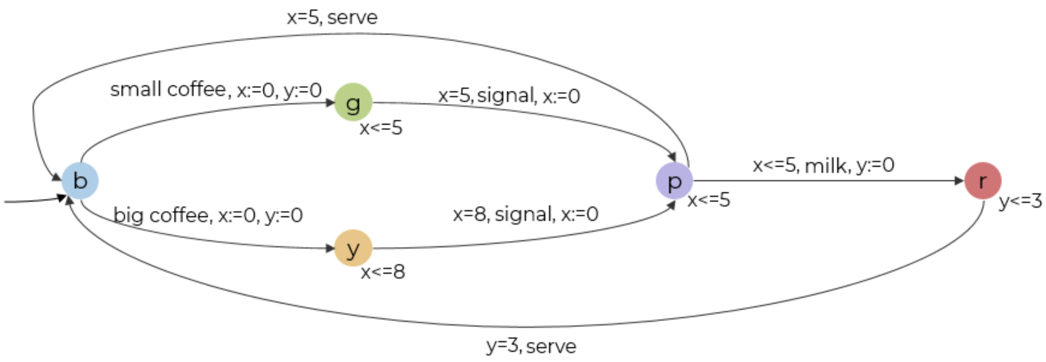
Go to www.menti.com

8213 0500

Semantics of timed automaton - Examples

Give an example of a run.

A run is alternating sequences of concrete states and actions or time elapses.



$$(b, x = 0, y = 0) \xrightarrow{2.4} (b, x = 2.4\ y = 2.4) \xrightarrow{small\ coffee} (g, x = 0, y = 0) \xrightarrow{3,3} (g, x = 3.3, y = 3.3) \xrightarrow{signal}$$

$$(g, x = 3.3, y = 3.3) \xrightarrow{2.7} (g, x = 5, y = 5) \xrightarrow{signal} (p, x = 0, y = 5)$$

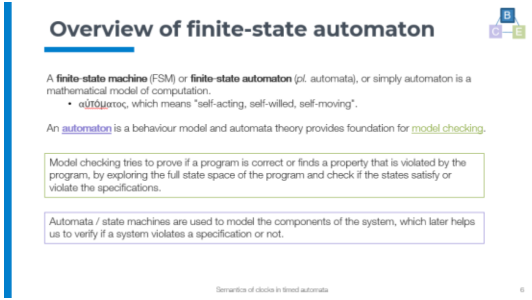
Some of the slides in this lecture are adopted from the following sources:

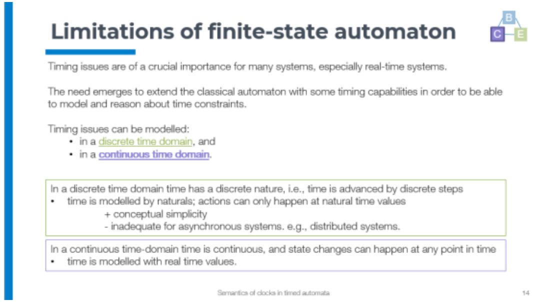
1. R. Alur and D.L. Dill. *"A Theory of Timed Automata"*
2. *C. Baier and J. Katoen. "Principles of Model Checking"*
3. *J. Katoen. "Timed Automata", lecture notes*
4. N. Saeedloei. *"An Introduction to Timed Automata", lecture notes*
5. *E. Andre. "Parametric Timed Automata, basic definitions and examples", lecture notes*

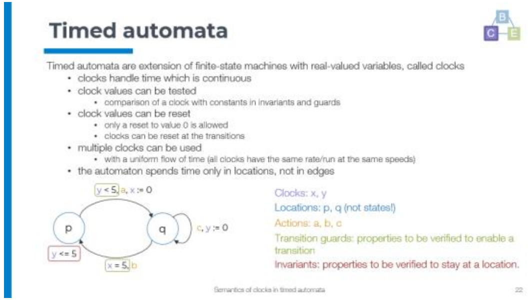
The sources of the images on the second slide:

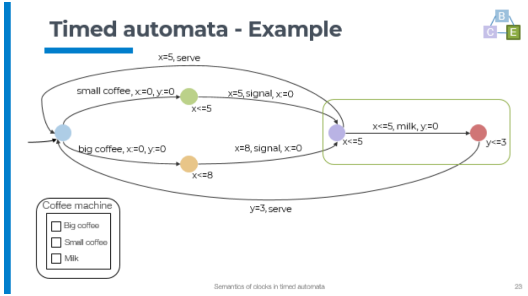
https://www.shutterstock.com/image-photo/sad-white-man-49687765
https://www.alibaba.com/product-detail/High-efficient-coffee-machine-special-for_1886447932.html
https://www.istockphoto.com/de/foto/coffee-cup-isoliert-gm1143290013-306985477
https://stock.adobe.com/search?k=overflowing+coffee&asset?id=54402508
https://stock.adobe.com/hk/search?k=excited%20black%20man

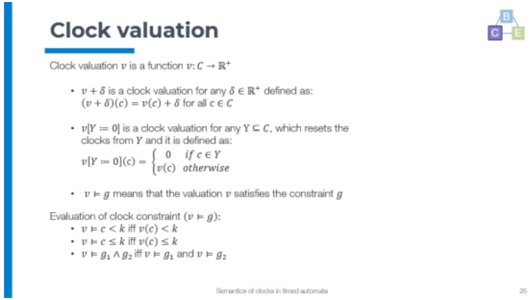
Summary

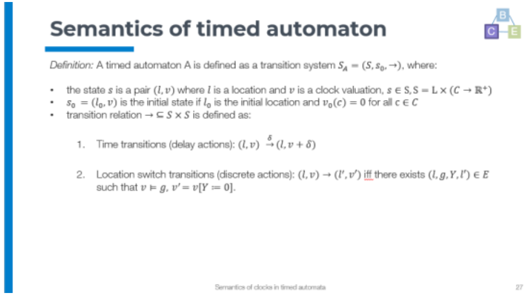












You can download the slides on the following link: https://github.com/tum-i4/KIT-Timed-Automata