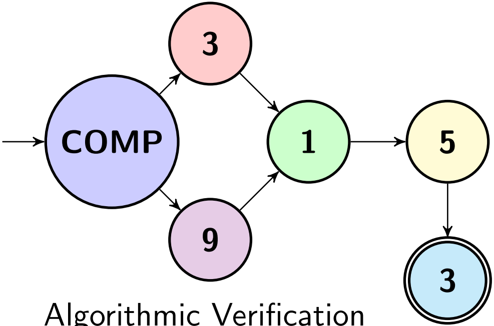


Algorithmic Verification

Timed Automata and Languages

Dr. Liam O'Connor
CSE, UNSW (and LFCS, University of Edinburgh)
Term 1 2020

Timed Systems

The systems we have examined so far have a notion of time, but only of events happening one after the other. This is an abstraction called *discrete time*.

Timed Systems

The systems we have examined so far have a notion of time, but only of events happening one after the other. This is an abstraction called *discrete time*.

For *dense time* systems, we care about a real-valued *continuous* time clocks.

Timed Systems

The systems we have examined so far have a notion of time, but only of events happening one after the other. This is an abstraction called *discrete time*.
For *dense time* systems, we care about a real-valued *continuous* time clocks.

Example (Dense Time System)

A light controlled by one button, where a “double press” of the button increases the brightness of the light. The second button press must be at most 3 time units after the first button press for the “double press” behaviour to trigger.
After 12 time units, the light must turn off.

Can we get away with discrete time?

No

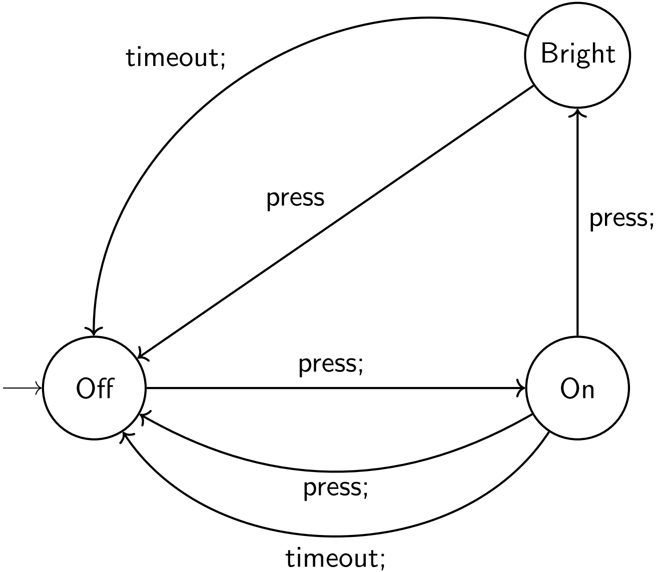
Can we get away with discrete time?

No

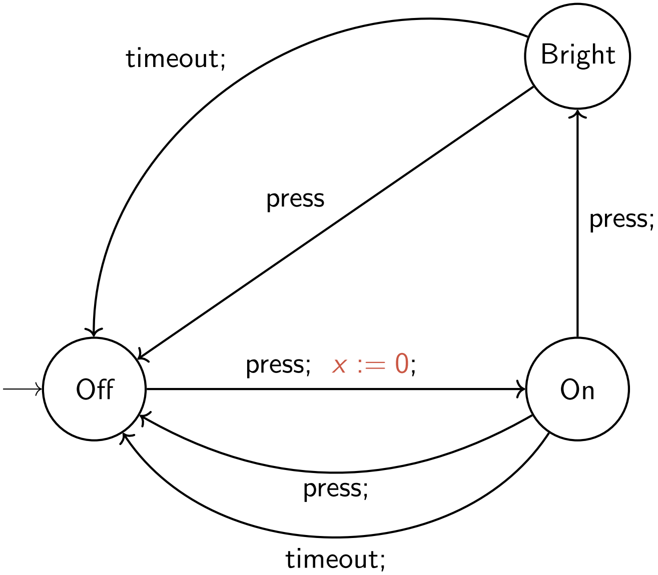
Theorem (Brzozowski and Seger)

For every $k \geq 1$ there is a system where the set of states reachable in dense time is strictly larger than the set of states reachable in discrete time in $\frac{1}{k}$ steps.
This is shown for asynchronous circuits, but applies generally.

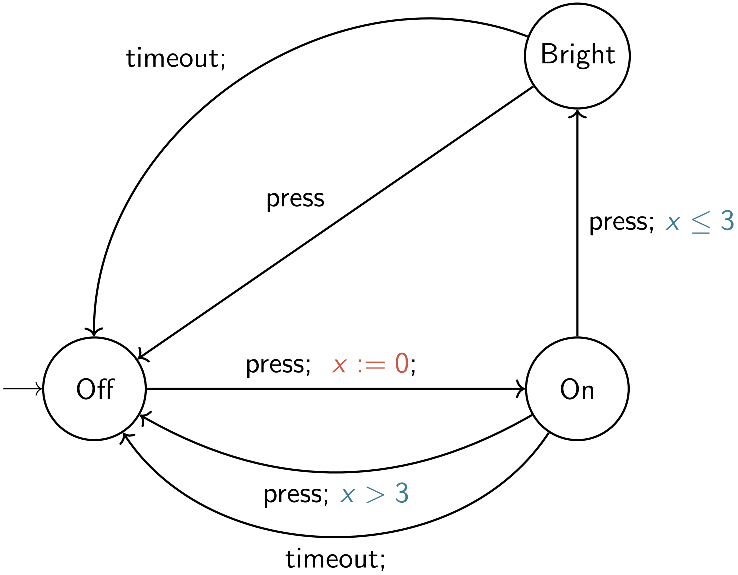
Towards Timed Automata



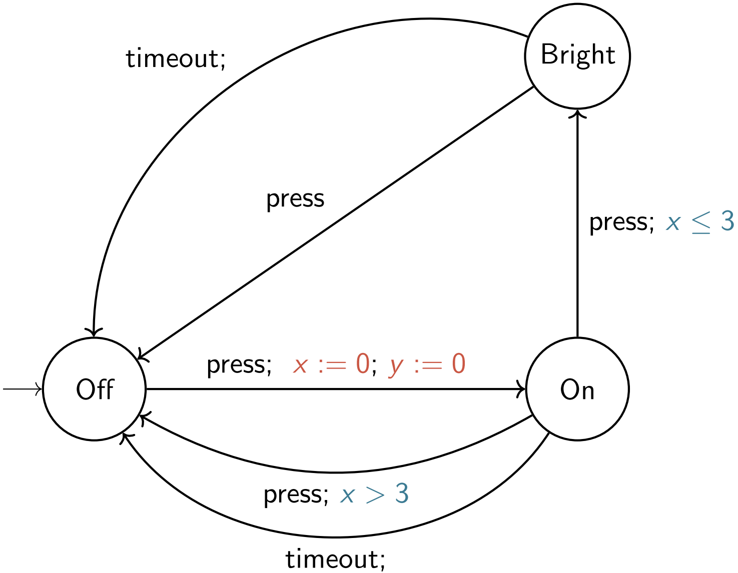
Towards Timed Automata



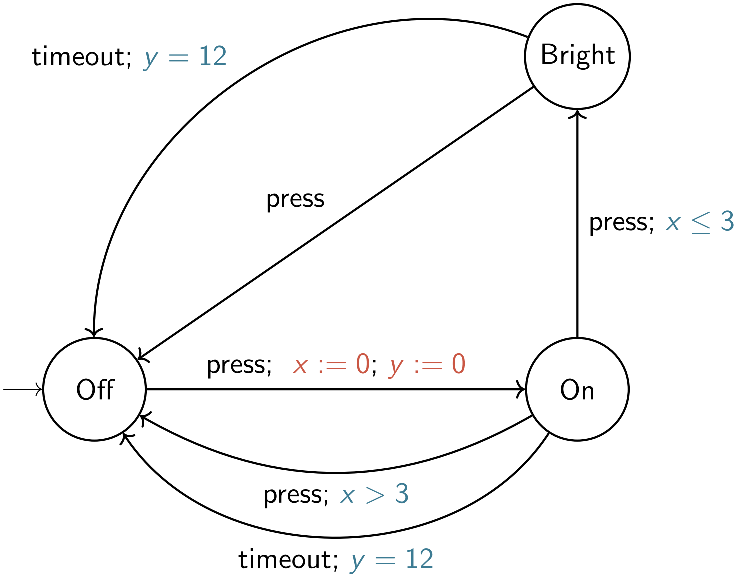
Towards Timed Automata



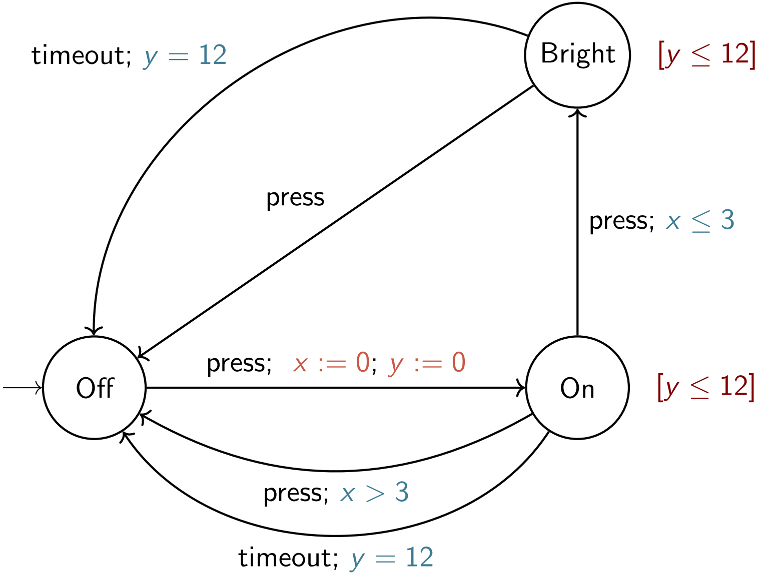
Towards Timed Automata



Towards Timed Automata



Towards Timed Automata



Timed Automata

A timed automaton is a finite automaton with one or more real valued clocks. Transitions are annotated with resets and guards and states may be annotated with invariants.

Timed Automata

A timed automaton is a finite automaton with one or more real valued clocks. Transitions are annotated with resets and guards and states may be annotated with invariants.

Definition

A timed automaton $\mathcal{A}$ is a 6-tuple $(L, \ell_0, \mathrm{Act}, X, \mathrm{Inv}, \longrightarrow)$ where:

- L is a set of locations.
- ℓ_0 is the initial location.
- Act is the set of discrete actions.
- X is the set of clock variables.
- $\mathrm{Inv}(\ell)$ is a *clock constraint* invariant associated with ℓ.
- Transitions are defined as $\ell \xrightarrow{g;a;r} \ell'$ where
  - g is zero or more clock constraint *guards*.
  - a is an action $\in$ Act
  - r is zero or more *clock resets*

Clock Constraints

For reasons that will become clear later, we want to restrict clock constraints to linear subtractions:

$$\varphi ::= x \sim k \mid x - y \sim k \mid \varphi_1 \wedge \varphi_2$$

where $x, y \in X$ and $k \in \mathbb{Z}$ and $(\sim) \in \{<, \leq, =, \geq, >\}$

States and Runs

States

The *state* of a timed automaton is a tuple of the location $\in L$ and the values of all clocks $\in \mathbb{R}$.

States and Runs

States

The *state* of a timed automaton is a tuple of the location $\in L$ and the values of all clocks $\in \mathbb{R}$.

Runs

A *run* of a timed automaton is some interleaving of *delay* steps (which pass some time) and *discrete* steps, which take an action.

States and Runs

States

The *state* of a timed automaton is a tuple of the location $\in L$ and the values of all clocks $\in \mathbb{R}$.

Runs

A *run* of a timed automaton is some interleaving of *delay* steps (which pass some time) and *discrete* steps, which take an action.

Example (For the light automaton. . .)

$$(\mathrm{Off}, x = y = 0)$$

States and Runs

States

The *state* of a timed automaton is a tuple of the location $\in L$ and the values of all clocks $\in \mathbb{R}$.

Runs

A *run* of a timed automaton is some interleaving of *delay* steps (which pass some time) and *discrete* steps, which take an action.

Example (For the light automaton…)

$$(\mathrm{Off}, x = y = 0) \xrightarrow{265+\pi^2} (\mathrm{Off}, x = y = 265 + \pi^2)$$

States and Runs

States

The *state* of a timed automaton is a tuple of the location $\in L$ and the values of all clocks $\in \mathbb{R}$.

Runs

A *run* of a timed automaton is some interleaving of *delay* steps (which pass some time) and *discrete* steps, which take an action.

Example (For the light automaton...)

$$(\mathsf{Off}, x = y = 0) \xrightarrow{265+\pi^2} (\mathsf{Off}, x = y = 265 + \pi^2) \xrightarrow{\text{press}} (\mathsf{On}, x = y = 0)$$

States and Runs

States

The *state* of a timed automaton is a tuple of the location $\in L$ and the values of all clocks $\in \mathbb{R}$.

Runs

A *run* of a timed automaton is some interleaving of *delay* steps (which pass some time) and *discrete* steps, which take an action.

Example (For the light automaton. . .)

$$(\mathsf{Off}, x = y = 0) \xrightarrow{265+\pi^2} (\mathsf{Off}, x = y = 265 + \pi^2) \xrightarrow{\mathsf{press}}$$
$$(\mathsf{On}, x = y = 0) \xrightarrow{3.2} (\mathsf{On}, x = y = 3.2)$$

States and Runs

States

The *state* of a timed automaton is a tuple of the location $\in L$ and the values of all clocks $\in \mathbb{R}$.

Runs

A *run* of a timed automaton is some interleaving of *delay* steps (which pass some time) and *discrete* steps, which take an action.

Example (For the light automaton...)

$$(\mathrm{Off}, x = y = 0) \xrightarrow{265+\pi^2} (\mathrm{Off}, x = y = 265 + \pi^2) \xrightarrow{\text{press}}$$
$$(\mathrm{On}, x = y = 0) \xrightarrow{3.2} (\mathrm{On}, x = y = 3.2) \xrightarrow{\text{press}}$$
$$(\mathrm{Off}, x = y = 3.2)$$

States and Runs

States

The *state* of a timed automaton is a tuple of the location $\in L$ and the values of all clocks $\in \mathbb{R}$.

Runs

A *run* of a timed automaton is some interleaving of *delay* steps (which pass some time) and *discrete* steps, which take an action.

Example (For the light automaton...)

$$(\text{Off}, x = y = 0) \xrightarrow{265+\pi^2} (\text{Off}, x = y = 265 + \pi^2) \xrightarrow{\text{press}}$$
$$(\text{On}, x = y = 0) \xrightarrow{3.2} (\text{On}, x = y = 3.2) \xrightarrow{\text{press}}$$
$$(\text{Off}, x = y = 3.2) \xrightarrow{27.87} (\text{Off}, x = y = 31.07) \longrightarrow \cdots$$

Product

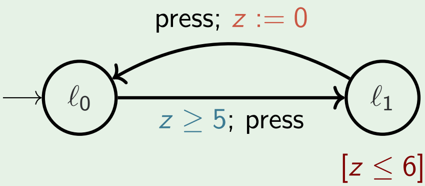
The product of timed automata is as the discrete time product, where we take the conjunction of all guards and invariants, and the union of all resets.

Product

The product of timed automata is as the discrete time product, where we take the conjunction of all guards and invariants, and the union of all resets.

Example (Boardwork)

Let's compute the product of the light automaton with this user automaton:



Timed Words and Languages

Timed Words

A timed word is a finite or infinite sequence of pairs of actions and time stamps.

Timed Words and Languages

Timed Words

A timed word is a finite or infinite sequence of pairs of actions and time stamps.

Example (overlines indicate sequences)

$$L_1 = \{\overline{(a,t)} \mid \overline{a} = (01)^*0 \land \forall i \geq 0.\ t_{2(i+1)} - t_{2i} = 3\}$$
$$L_2 = \{\overline{(a,t)} \mid \overline{a} = (01)^\omega \land \forall i \geq 0.\ t_{2i+1} \leq t_{2i} + 1\}$$

Timed Words and Languages

Timed Words

A timed word is a finite or infinite sequence of pairs of actions and time stamps.

Example (overlines indicate sequences)

$$L_1 = \{\overline{(a,t)} \mid \overline{a} = (01)^*0 \wedge \forall i \geq 0.\ t_{2(i+1)} - t_{2i} = 3\}$$
$$L_2 = \{\overline{(a,t)} \mid \overline{a} = (01)^\omega \wedge \forall i \geq 0.\ t_{2i+1} \leq t_{2i} + 1\}$$

Timed Languages

Extend TA definition of $\mathcal{A}$ with a set F of final states and a set R of repeating states.

- A finite word w is $\in \mathcal{L}(\mathcal{A})$ iff a run generating the word w ends in a state F.
- An infinite word w is $\in \mathcal{L}(\mathcal{A})$ iff a run generating the word w visits states in R infinitely often.

Timed Regular Languages

Definition

A language is *timed regular* iff there exists a timed automaton that recognises it.

Timed Regular Languages

Definition

A language is *timed regular* iff there exists a timed automaton that recognises it.

Timed regular languages are closed under union and intersection in the usual way — merge the initial states and product respectively.

Timed Regular Languages

Definition

A language is *timed regular* iff there exists a timed automaton that recognises it.

Timed regular languages are closed under union and intersection in the usual way — merge the initial states and product respectively.

Problem

Write a timed automaton for words on the alphabet $\{a, b\}$ that contain two a's exactly 1 time unit apart.

Timed Regular Languages

Definition

A language is *timed regular* iff there exists a timed automaton that recognises it.

Timed regular languages are closed under union and intersection in the usual way — merge the initial states and product respectively.

Problem

Write a timed automaton for words on the alphabet $\{a, b\}$ that contain two a's exactly 1 time unit apart.
Complement is not timed regular $\Rightarrow$ not closed.

ε-Transitions

Unlike for discrete time systems, ε transitions add power for timed automata.

ε-Transitions

Unlike for discrete time systems, ε transitions add power for timed automata.

Example

Consider the language where actions must occur on integer time stamps. This can be done with a ε reset, but cannot be expressed as a timed automaton without ε.

True Invariants

Do we need invariants? Not to recognise a given timed regular language.

True Invariants

Do we need invariants? Not to recognise a given timed regular language.

This is because our timed words only pair time stamps with discrete actions, so violating invariants by sitting still does not change the set of recognised words.

True Invariants

Do we need invariants? Not to recognise a given timed regular language.
This is because our timed words only pair time stamps with discrete actions, so violating invariants by sitting still does not change the set of recognised words.
So, we just move the invariants to both the incoming and outgoing transitions like so:

$$\ell_1 \xrightarrow{g;a;r} \ell_2$$

becomes

$$\ell_1 \xrightarrow{g \ \land \ \mathsf{Inv}(\ell_1) \ \land \ r(\mathsf{Inv}(\ell_2));a;r} \ell_2$$

Where $r(\varphi)$ is applying the resets r as a substitution to φ.