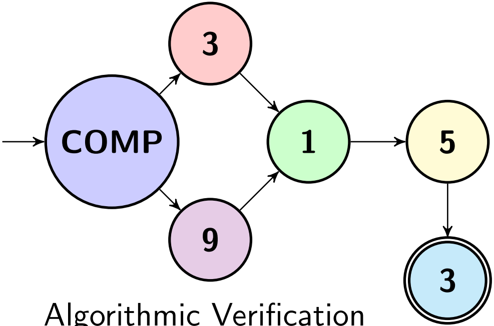


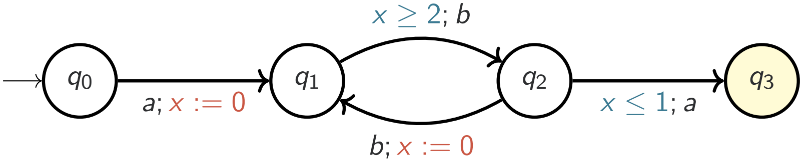
Algorithmic Verification

Timed Reachability and Abstraction

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

Reachability

The state space of timed automata are uncountably infinite (because of real valued clocks).
How can we compute if a state is reachable?



Key Idea

We will group clock values into equivalence classes to make a finite abstraction of the timed automata which preserves reachability.

Instead of specific amounts of time passing, we will add a generic "time passing" action called δ to our abstraction.

Time Abstraction

Ignoring the specific time elapsed and instead just using δ is called *time abstraction*.

Equivalences

Our normal notions of bisimulation and trace equivalence apply also to timed automata. For example, two timed automata are *timed trace equivalent* iff they accept the same timed language.

Two automata are *time-abstract equivalent* iff they appear equivalent after applying time abstraction.

Timed equivalence is *finer* than time-abstract equivalence, but time-abstract systems are more tractable.

Regions

Regions

Our clock valuations are grouped into *regions*. Regions should:

1. Satisfy the same clock constraints.
2. Reach the same regions from time passing (called the *successor regions*).

Recall our definition of clock constraints:

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

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

The smallest regions these constraints can distinguish (in 2D):

1. Integer Points: $x = 0 \wedge y = 0$? **Yep!**
2. Open Intervals: $x = 0 \wedge 2 < y < 3$? **Yep!**
3. Open Squares: $0 < x < 1 \wedge 1 < y < 2$? **No!** We can go finer!
4. Diagonals: $x - y = 1 \wedge 1 < x < 2$? **Yep!**
5. Open Triangles: $x - y < 2 \wedge 3 < x < 4$? **Yep!**

A finite bound

Depending on the specific constraints involved, we can sometimes merge regions where the difference between them is not important.

Maximal Constant

The *maximal constant* K of a system is the highest integer constant that occurs in the set of all clock constraints.
Regions dealing with clock values between K and ∞ can always be merged.

The maximal constant merging gives us a finite bound on the number of regions: $\mathcal{O}(|X|! \cdot K^{|X|})$

Region Graph Definition

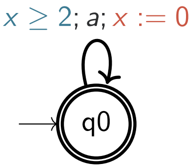
Region Graph

The region graph of a TA $\mathcal{A} = (L, \ell_0, \mathrm{Act}, X, \mathrm{Inv}, \longrightarrow)$ is the automaton $(L \times \mathrm{Regions}(\mathcal{A}), (\ell_0, \overline{0}), \mathrm{Act} \cup \{\delta\}, \Longrightarrow)$ where:

- $\mathrm{Regions}(\mathcal{A})$ is the set of all regions.
- $\overline{0}$ is the region where all clock values are zero.
- For $a \in \mathrm{Act}$, $(\ell, R) \stackrel{a}{\Longrightarrow} (\ell', R')$ if
  - There is an edge $\ell \xrightarrow{g;a;r} \ell'$,
  - R implies g
  - $R' = r(R)$
- $(\ell, R) \stackrel{\delta}{\Longrightarrow} (\ell', R')$ if
  - R' is a successor of R
  - R implies $\mathrm{Inv}(\ell)$ and R' implies $\mathrm{Inv}(\ell')$
- For any open region R we also have $(\ell, R) \stackrel{\delta}{\Longrightarrow} (\ell, R)$.

Tiny Example

Let's try a simple region graph following the formal definition for this one-dimensional, one state TA:



Key Properties

The big win

A location is reachable in the region graph of a timed automaton A iff it is reachable in A.

As the region graph is just a normal finite automaton, reachability is decidable for TA, although PSPACE-complete.