EE 144/244: Fundamental Algorithms for System Modeling, Analysis, and Optimization
Fall 2016

Timed Automata

Stavros Tripakis
University of California, Berkeley

Timed Automata

- A formal model for dense-time systems [Alur and Dill(1994)]
- Developed mainly with verification in mind:
  - in the basic TA variant, model-checking is decidable
- But also an elegant theoretical extension of the standard theory of regular and ω-regular languages.
- Many different TA variants, some undecidable.
- We will look at a basic variant.

Timed Automaton

A TA is a tuple

$$(C, Q, q_0, \mathsf{Inv}, \triangleright)$$

- C: finite set of *clocks*
- Q: finite set of *control states*; $q_0 \in Q$: initial control state
- Inv: a function assigning to each $q \in Q$ an *invariant*
- $\triangleright$: a finite set of *actions*, each being a tuple

$$(q, q', g, C')$$

  - $q, q' \in Q$: source and destination control states
  - g: clock *guard*
  - C': set of clocks to *reset* to 0, $C' \subseteq C$
- Invariants and guards are simple constraints on clocks, e.g.,

$$c \leq 1, \quad 0 < c_1 < 2 \wedge c_2 = 4, \quad \text{etc.}$$

Timed Automaton

A TA is a tuple

$$(C, Q, q_0, \mathsf{Inv}, \triangleright)$$

- C: finite set of *clocks*
- Q: finite set of *control states*; $q_0 \in Q$: initial control state
- Inv: a function assigning to each $q \in Q$ an *invariant*
- $\triangleright$: a finite set of *actions*, each being a tuple

$$(q, q', g, C')$$

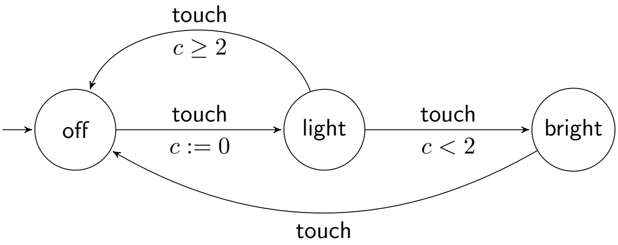
  - $q, q' \in Q$: source and destination control states
  - g: clock *guard*
  - C': set of clocks to *reset* to 0, $C' \subseteq C$
- Invariants and guards are simple constraints on clocks, e.g.,

$$c \leq 1, \quad 0 < c_1 < 2 \wedge c_2 = 4, \quad \text{etc.}$$

Can also have atomic propositions labeling control states, labels on actions, communication via shared memory or message passing, etc.

Example: Timed Automaton

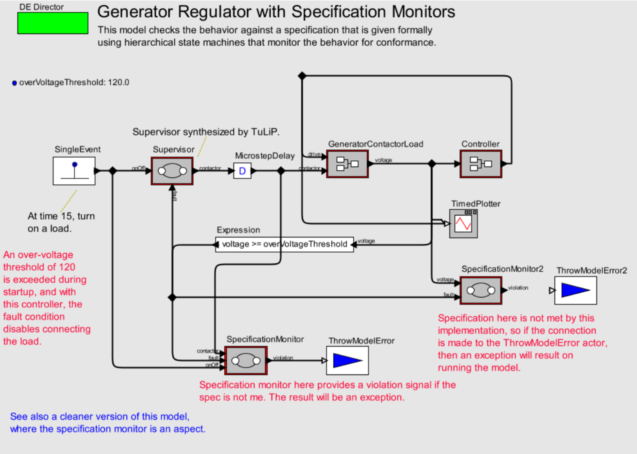
A simple light controller:



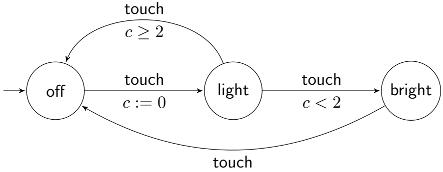
- $C = \{c\}$
- $Q = \{$ off, light, bright $\}$
- $q_0 =$ off
- touch: action label (can be seen as the input symbol)
- Inv$(q) = \mathit{true}$ for all $q \in Q$
- Actions: (off, light, $\mathit{true}, \{c\}$), (light, off, $c \geq 2, \{\}$), ...

Event-based vs. state-based models

High-level:



Low-level:



Timed Automata: Semantics

A TA $(C, Q, q_0, \mathsf{Inv}, \triangleright)$ defines a transition system

$$(S, S_0, R)$$

such that

- Set of states: $S = Q \times \mathbb{R}_+^C$
  - $\mathbb{R}_+^C$: the set of all functions $v : C \to \mathbb{R}_+$
  - each v is called a *valuation*: it assigns a value to every clock
- Set of initial states: $S_0 = \{(q_0, v_0)\}$, where we define $v_0(c) = 0$ for all $c \in C$ (i.e., all clocks are initially set to 0)
- Set of transitions: $R = R_t \cup R_d$
  - R_t: set of transitions modeling passage of time
  - R_d: set of discrete transitions ("jumps" between control states)

Timed Automata: Semantics

A TA $(C, Q, q_0, \mathsf{Inv}, \triangleright)$ defines a transition system

$$(S, S_0, R)$$

such that

- Set of states: $S = Q \times \mathbb{R}_+^C$
  - $\mathbb{R}_+^C$: the set of all functions $v : C \to \mathbb{R}_+$
  - each v is called a *valuation*: it assigns a value to every clock
- Set of initial states: $S_0 = \{(q_0, v_0)\}$, where we define $v_0(c) = 0$ for all $c \in C$ (i.e., all clocks are initially set to 0)
  - we could also define $S_0 = \{q_0\} \times \mathbb{R}_+^C$ – what does this say?
- Set of transitions: $R = R_t \cup R_d$
  - R_t: set of transitions modeling passage of time
  - R_d: set of discrete transitions ("jumps" between control states)

Timed Automata: Discrete and Time Transitions

$$
\begin{aligned}
R_t &= \{\big((q,v),(q,v+t)\big) \mid \forall t' \le t : v + t' \models \mathsf{Inv}(q)\} \\
R_d &= \{\big((q,v),(q',v')\big) \mid \exists a = (q,q',g,C') \in \rhd : \\
&\qquad\qquad v \models g \wedge v' = v[C' := 0]\}
\end{aligned}
$$

where:

- $v + t$ is a new valuation u such that $u(c) = v(c) + t$ for all c
- if g is a constraint, then $v \models g$ means v satisfies g
- $v[C' := 0]$ is a new valuation u such that $u(c) = 0$ if $c \in C'$ and $u(c) = v(c)$ otherwise

Timed Automata: Discrete and Time Transitions

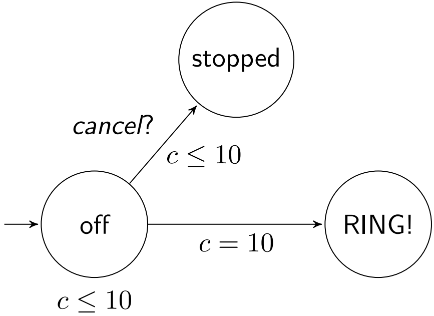
$$
\begin{aligned}
R_t &= \{\big((q,v),(q,v+t)\big) \mid \forall t' \le t : v+t' \models \mathsf{Inv}(q)\} \\
R_d &= \{\big((q,v),(q',v')\big) \mid \exists a = (q,q',g,C') \in \rhd : \\
&\qquad\qquad\qquad\qquad\qquad v \models g \wedge v' = v[C' := 0]\}
\end{aligned}
$$

where:

- $v+t$ is a new valuation u such that $u(c) = v(c) + t$ for all c
- if g is a constraint, then $v \models g$ means v satisfies g
- $v[C' := 0]$ is a new valuation u such that $u(c) = 0$ if $c \in C'$ and $u(c) = v(c)$ otherwise

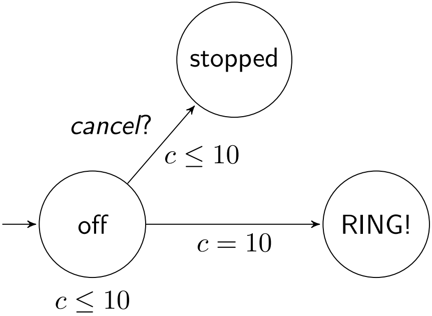
Instead of $\big((q,v),(q,v+t)\big) \in R_t$ we write $(q,v) \xrightarrow{t} (q,v+t)$.
Instead of $\big((q,v),(q',v')\big) \in R_d$ we write $(q,v) \xrightarrow{a} (q',v')$.

Example: Alarm Modeled as a Timed Automaton



Inv(off) $= c \leq 10$: automaton cannot spend more than 10 time units at control state "off".

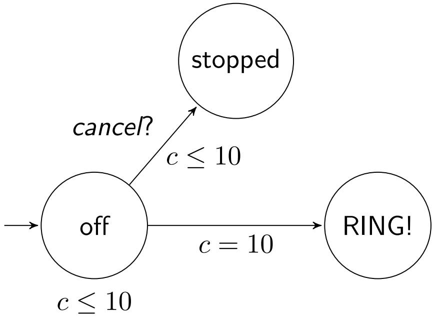
Example: Alarm Modeled as a Timed Automaton



Inv(off) $= c \leq 10$: automaton cannot spend more than 10 time units at control state "off".

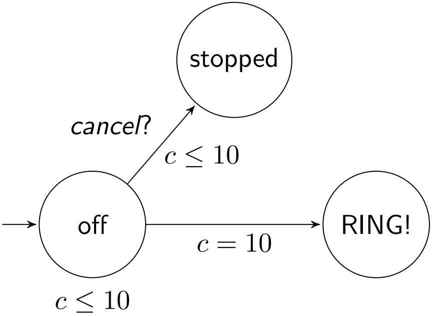
What if we omit the invariant?

Example: Alarm Modeled as a Timed Automaton



Does it work correctly if *cancel* arrives exactly when $c = 10$?

Example: Alarm Modeled as a Timed Automaton



Does it work correctly if *cancel* arrives exactly when $c = 10$?

Depends on the semantics of composition: if it's non-deterministic (as usually done) then alarm may still ring. Otherwise, must give higher priority to the cancel transition.

Timed Automata Model-Checking: Reachability

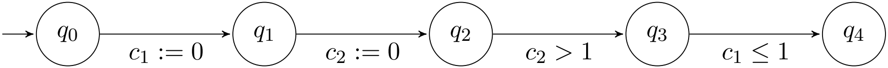
- Basic question: is a given **control state** q **reachable**?
  - i.e., does there exist some reachable state $s = (q, v)$ in the transition system defined by the timed automaton?
- Many interesting questions about timed automata can be reduced to this question.

Timed Automata Model-Checking: Reachability

- Basic question: is a given **control state** q **reachable**?
  - i.e., does there exist some reachable state $s = (q, v)$ in the transition system defined by the timed automaton?
- Many interesting questions about timed automata can be reduced to this question.
- Is the basic control-state reachability question decidable?

Timed Automata Reachability

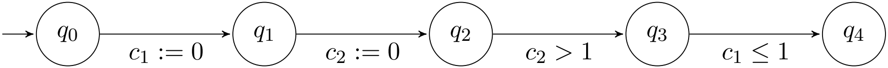
Not the same as discrete-state reachability!



q_4 is reachable if we ignore the timing constraints.
But is it really reachable?

Timed Automata Reachability

Not the same as discrete-state reachability!

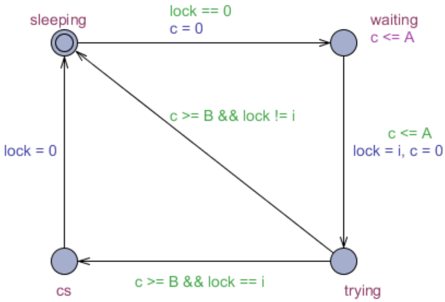


q_4 is reachable if we ignore the timing constraints.
But is it really reachable?

No: at q_3, $c_2 > 1$ and $c_1 \geq c_2$, therefore $c_1 > 1$ also.

Timed Automata Model-Checking: Reachability

A less obvious example: Fischer's mutual exclusion protocol.



Suppose we have many processes, each behaving like the TA above.
Is mutual-exclusion guaranteed?
I.e., at most 1 process is in critical section (control state cs) at any given time.

Timed Automata Model-Checking: Reachability

Brute-force idea: exhaustive state-space exploration of the transition system defined by the timed automaton

- does not work since state-space is infinite (even uncountable)

Timed Automata Model-Checking: Reachability

Brute-force idea: exhaustive state-space exploration of the transition system defined by the timed automaton

- does not work since state-space is infinite (even uncountable)

Yet problem is decidable! [Alur-Dill'94]

Key idea:

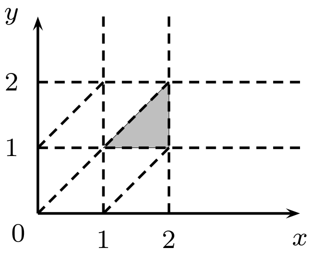
- *Region equivalence*: partitions the state-space into **finite** number of equivalence classes (*regions*)
- Perform reachability on finite (abstract) state-space
- Can prove that q is reachable in the abstract space iff it is reachable in the concrete space

The Region Equivalence

Key idea: two valuations v_1, v_2 are equivalent iff:

1. v_1 satisfies a guard g iff v_2 satisfies g.
2. v_1 can lead to some v_1' satisfying a guard g with a discrete transition iff v_2 can do the same.
3. v_1 can lead to some v_1' satisfying a guard g with a time transition iff v_2 can do the same.

Region = equivalence class w.r.t. region equivalence = set of all equivalent valuations.

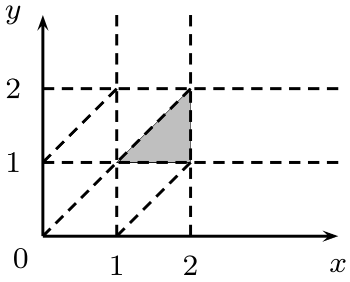


Region in gray:
$1 < x < 2 \wedge 1 < y < 2 \wedge x > y$.
Other regions:
$x = y = 0$,
$0 < x = y < 1$,
$x = 0 \wedge 0 < y < 1$,
etc.

Pictures in this and other slides taken from [Bouyer(2005)].

The Region Equivalence: Finiteness

Finite number of equivalence classes: bounded by constant $c =$ maximal constant appearing in a guard or invariant.



Some regions are unbounded, e.g.:

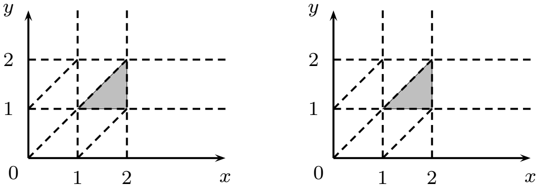
$x > 2 \wedge 0 < y < 1$

$x > 2 \wedge y > 2$

etc.

The Region Graph

A graph of regions: one region space for each control location.



q_1 q_2

Nodes: pairs (q, r) where

- q is a control location of the timed automaton.
- r is a region.

Two types of edges:

- $(q, r) \xrightarrow{a} (q', r')$: discrete transition
- $(q, r) \xrightarrow{time} (q, r')$: time transition

Decidability

Theorem ([Alur and Dill(1994)])

$\exists$ reachable state (q, v) in a timed automaton
iff
$\exists$ reachable node (q, r) in its region graph.

Finite # regions and control states $\Rightarrow$ Region graph is finite $\Rightarrow$ Reachability is decidable.

The Problem with Regions

STATE EXPLOSION!

Worst-case number of regions:

$$O(2^n \cdot n! \cdot c^n)$$

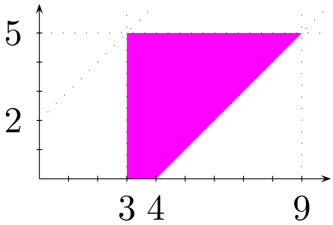
where n is the number of clocks and c is the maximal constant.

This is actually often close to the actual number of regions $\Rightarrow$ no practical tool uses regions.

Model-checkers for TA (Uppaal, Kronos, ...) have improved upon the region-graph idea and use *symbolic* techniques.

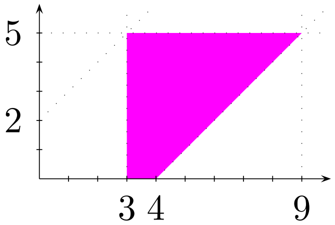
From Regions to Zones

Zone: a convex union of regions, e.g., $x_1 \geq 3 \wedge x_2 \leq 5 \wedge x_1 - x_2 \leq 4$.



From Regions to Zones

Zone: a convex union of regions, e.g., $x_1 \geq 3 \wedge x_2 \leq 5 \wedge x_1 - x_2 \leq 4$.



Key property: can be represented efficiently using *difference bound matrices* (DBMs) [Dill(1989)].

$$x_1 \geq 3 \ \wedge x_2 \leq 5 \wedge x_1 \leq x_2 + 4 \quad : \quad \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \left(\begin{array}{ccc} \infty & -3 & \infty \\ \infty & \infty & 4 \\ 5 & \infty & \infty \end{array} \right) \end{array}$$

Symbolic Manipulations of Zones using DBMs

DBMs = the BDDs of the timed automata world.

Time elapse, guard intersection, clock resets, are all easily implementable in DBMs.

Symbolic Manipulations of Zones using DBMs

DBMs = the BDDs of the timed automata world.

Time elapse, guard intersection, clock resets, are all easily implementable in DBMs.

Is zone union implementable with DBMs?

Symbolic Manipulations of Zones using DBMs

DBMs = the BDDs of the timed automata world.

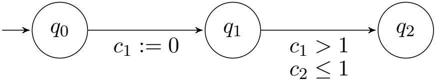
Time elapse, guard intersection, clock resets, are all easily implementable in DBMs.

Is zone union implementable with DBMs?

No! The union of two zones in general is not a zone.

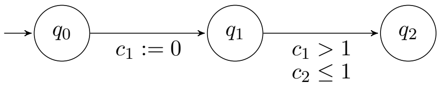
$\Rightarrow$ often state explosion even with zones ...

Timed Automata Reachability: Simple Example



Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Timed Automata Reachability: Simple Example

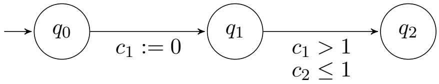


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$

Timed Automata Reachability: Simple Example

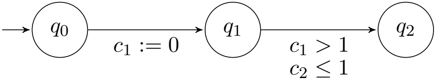


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	

Timed Automata Reachability: Simple Example

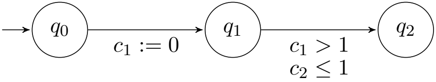


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	$(q_0, c_1 = c_2)$

Timed Automata Reachability: Simple Example

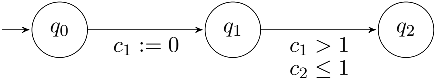


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	$(q_0, c_1 = c_2)$
take discrete transition to q_1:	

Timed Automata Reachability: Simple Example

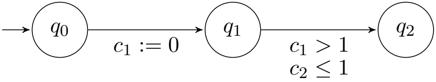


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	$(q_0, c_1 = c_2)$
take discrete transition to q_1:	$(q_1, c_1 = 0 \wedge c_2 \geq c_1)$

Timed Automata Reachability: Simple Example

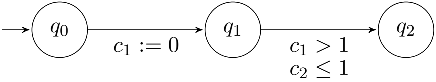


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	$(q_0, c_1 = c_2)$
take discrete transition to q_1:	$(q_1, c_1 = 0 \wedge c_2 \geq c_1)$
let time elapse:	

Timed Automata Reachability: Simple Example

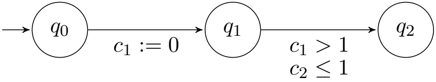


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	$(q_0, c_1 = c_2)$
take discrete transition to q_1:	$(q_1, c_1 = 0 \wedge c_2 \geq c_1)$
let time elapse:	$(q_1, c_2 \geq c_1)$

Timed Automata Reachability: Simple Example

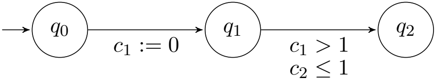


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	$(q_0, c_1 = c_2)$
take discrete transition to q_1:	$(q_1, c_1 = 0 \wedge c_2 \geq c_1)$
let time elapse:	$(q_1, c_2 \geq c_1)$
take discrete transition to q_2:	

Timed Automata Reachability: Simple Example

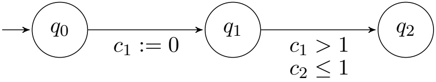


Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	$(q_0, c_1 = c_2)$
take discrete transition to q_1:	$(q_1, c_1 = 0 \wedge c_2 \geq c_1)$
let time elapse:	$(q_1, c_2 \geq c_1)$
take discrete transition to q_2:	cannot because
	$c_1 > 1 \wedge c_2 \leq 1 \wedge c_2 \geq c_1$ is UNSAT

Timed Automata Reachability: Simple Example



Is q_2 reachable? (initially, $c_1 = c_2 = 0$)

Symbolic reachability analysis:

step	symbolic state
initially:	$(q_0, c_1 = 0 \wedge c_2 = 0)$
let time elapse:	$(q_0, c_1 = c_2)$
take discrete transition to q_1:	$(q_1, c_1 = 0 \wedge c_2 \geq c_1)$
let time elapse:	$(q_1, c_2 \geq c_1)$
take discrete transition to q_2:	cannot because
	$c_1 > 1 \wedge c_2 \leq 1 \wedge c_2 \geq c_1$ is UNSAT
therefore q_2 not reachable	

Bibliography

R. Alur.
Timed automata.
NATO-ASI 1998 Summer School on Verification of Digital and Hybrid Systems, 1998.

R. Alur and D. Dill.
A theory of timed automata.
Theoretical Computer Science, 126:183–235, 1994.

P. Bouyer.
An introduction to timed automata.
At `http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/bouyer-etr05.pdf`, 2005.

D. Dill.
Timing assumptions and verification of finite-state concurrent systems.
In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 197–212. Springer, 1989.