Digitization Techniques for Timed Automata

Joël Ouaknine

Carnegie Mellon University

SVC October 1, 2002

Modeling Time

Two paradigms:

- **Discrete**: - Easy to model check
  - Potentially inaccurate
- **Dense**: - Precise
  - Infinite (uncountable) state space
  - Generally more expensive to model check

Note: In either case, model checking is PSPACE-complete.

Verification

A spectrum of verification approaches:

- **Untimed analysis (timewise refinement)** (Schneider)
- ⋮
- **Digitization techniques** (Henzinger, Manna, Pnueli)
- ⋮
- **Region graphs** (Alur, Courcoubetis, Dill)

(Also: symbolic techniques, proof systems, etc.)

Try to find and apply the cheapest technique that works!

Modeling Framework

Timed Automata (variation on Alur and Dill)

Timed automaton =

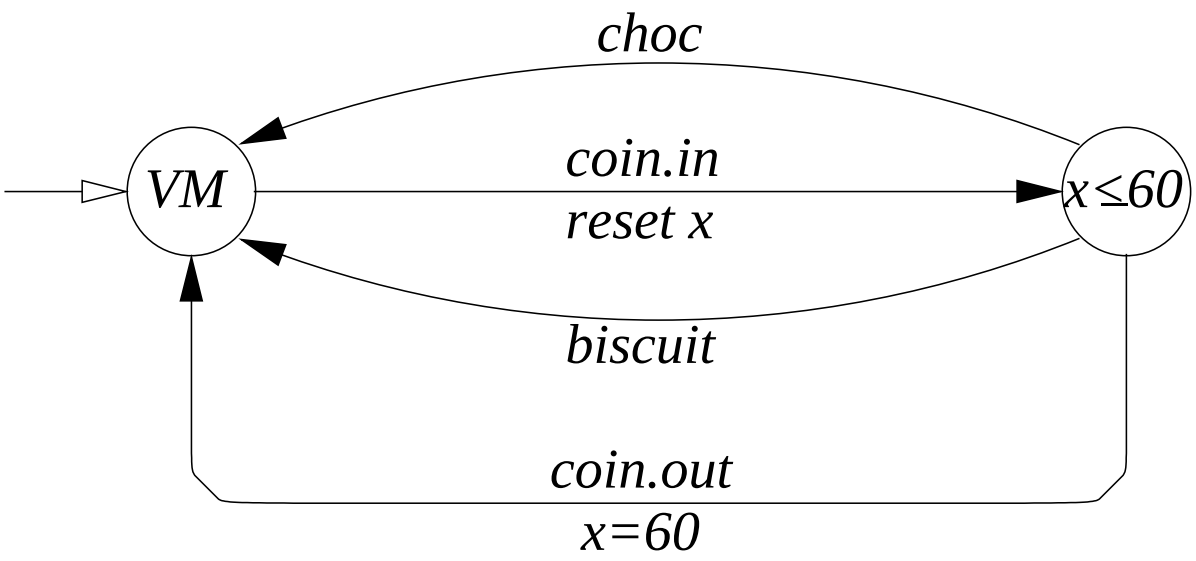
- **Automaton** (edges labelled with events) +
- Real-valued **clocks** (can be reset on transitions) +
- **Invariant** (state) and **enabling** (transition) **clock constraints**.

Clock constraints are boolean combinations of

$$\text{true} \mid x < k \mid x \leqslant k \mid x > k \mid x \geqslant k$$

where $k \in \mathbb{N}$.

Example: Vending Machine



Semantics

A **timed trace** $\langle (t_1, a_1), (t_2, a_2), \ldots, (t_k, a_k) \rangle$ is a finite sequence of timed events with timestamps non-negative real-valued and non-decreasing.

- $\mathbb{R}[\![A]\!] =$ set of dense-time timed traces of A
- $\mathbb{Z}[\![A]\!] =$ set of integral-time timed traces of A

Specifications

We consider **safety** specifications ('nothing bad happens'):

A **specification** is an 'allowable' set of timed traces.

Ex: $S =$ set of all traces not containing the event $error$.
(S is the most common type of safety property.)

Specifications can be expressed in temporal logic (e.g., MTL), mathematically, as the set of timed traces of another automaton, etc.
Specifications in practice tend to be very simple.

Let A be a timed automaton and S a specification:

- $A \vDash_{\mathbb{R}} S$ if $\mathbb{R}[\![A]\!] \subseteq S$
- $A \vDash_{\mathbb{Z}} S$ if $\mathbb{Z}[\![A]\!] \subseteq S\restriction\mathbb{Z}$

($S\restriction\mathbb{Z}$ denotes the integral-time timed traces of S.)

Verification

How do we check that a process meets its specification?

Difficulty: The number of possible behaviors (timed traces) is infinite (uncountable)!

We need to **discretize** processes. We will examine:

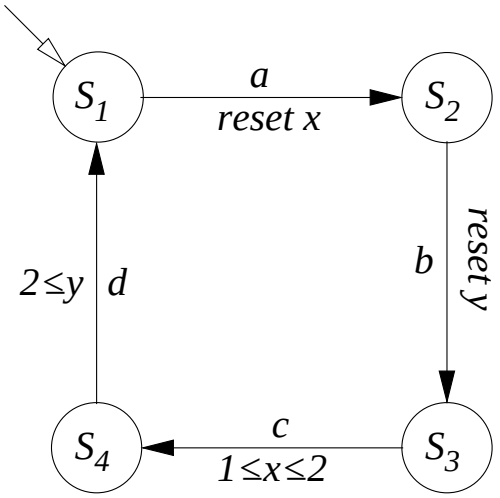
- **Untimed analysis (timewise refinement)** (Schneider)
- **Digitization techniques** (Henzinger, Manna, Pnueli)
- **Region graphs** (Alur, Courcoubetis, Dill)

Region Graphs

Sound and complete verification of TCTL formulas.

(TCTL: Qualitative timed version of CTL.)

Consider the following timed automaton A:



$\Sigma = \{a,b,c,d\}$

$States = \{S_1,S_2,S_3,S_4\}$

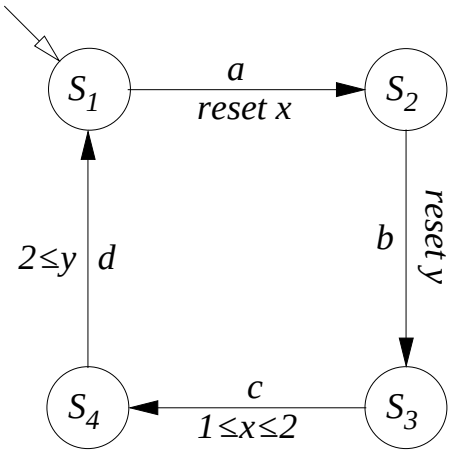
$Clocks = \{x,y\}$

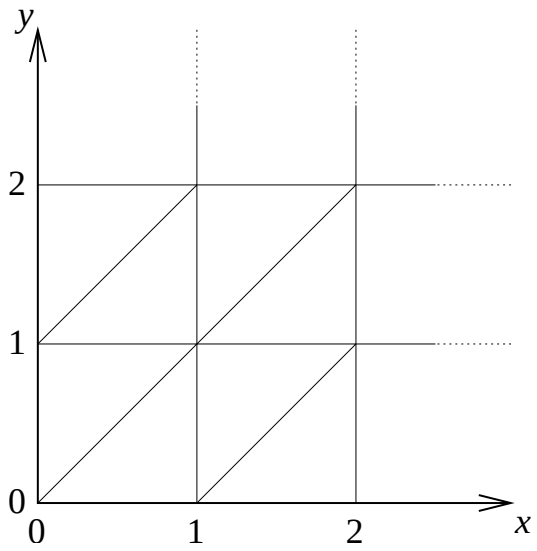
State space: $RxRx\{S_1,S_2,S_3,S_4\}$
(Uncountably infinite)

The idea is to build an associated **finite automaton** $RG(A)$ whose discrete behaviors correspond to **equivalence classes** of behaviors of the original timed automaton A.

Timed states of A within the same equivalence class are indistinguishable as far as TCTL specifications are concerned.

Thus for any TCTL specification S, $A \models_{\mathbb{R}} S \iff RG(A) \models S$.





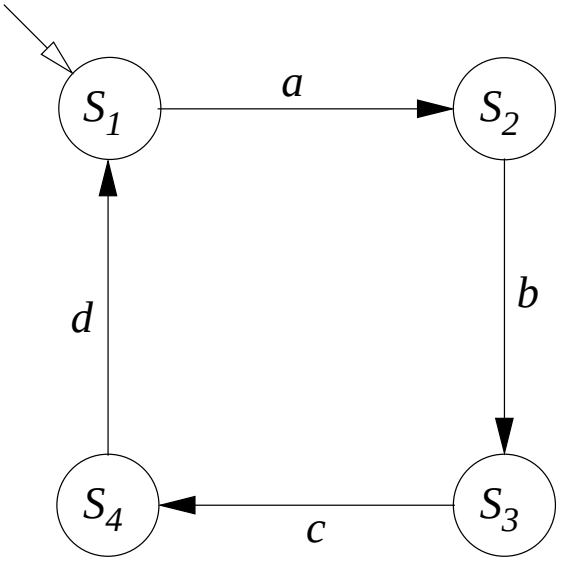
9+22+13 = 44 clock regions

State space: 44x4 = 176

Untimed Analysis

Drop all timing constraints! Get new untimed automaton $\Theta(A)$.

Sound verification of safety specifications. *Not* complete!



Σ = {a,b,c,d}

States = $\{S_1, S_2, S_3, S_4\}$

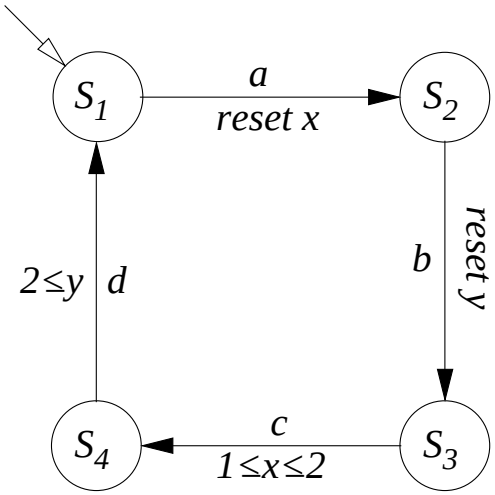
State space: 4

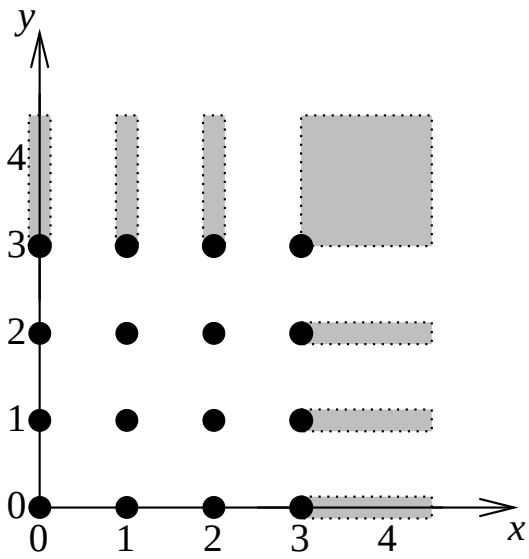
For any timed safety specification S, $\Theta(A) \models \Theta(S) \implies A \models_{\mathbb{R}} S$.

Digitization

Only consider integral values for clocks.

Sound and complete verification of specifications closed under inverse digitization on closed timed automata.





16 clock regions

State space: 16x4 = 64

For any specification S closed under inverse digitization,

$$A \vDash_{\mathbb{Z}} S \iff A \vDash_{\mathbb{R}} S \ .$$

Digitization In Greater Detail

Take, say, $t = 3.7$. How to 'digitize' it?

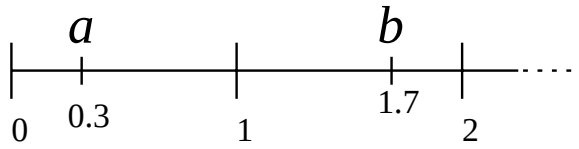
- Floor: $\lfloor 3.7 \rfloor = 3$
- Ceiling: $\lceil 3.7 \rceil = 4$
- Round: $[3.7] = 4$

Each of these operations is a special case of ε-**digitization**: rounding with respect to $\varepsilon \in [0, 1]$.

- $[3.7]_{0.2} = 4$, $[3.7]_{0.9} = 3$.

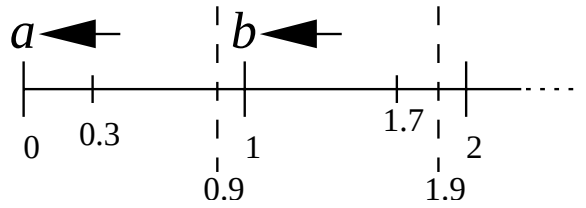
Thus in general we have $\lfloor t \rfloor = [t]_1$, $\lceil t \rceil = [t]_0$, $[t] = [t]_{0.5}$.

For a fixed ε, this operation applies pointwise to timed traces:

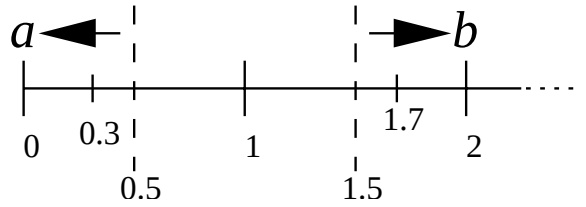


$$tr = \langle (0.3, a), (1.7, b) \rangle$$

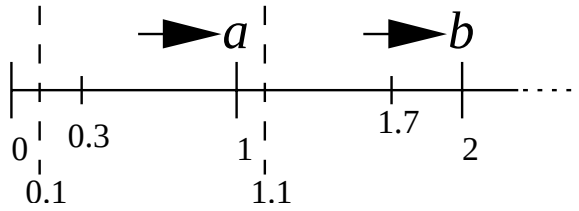
We have three possible ε-digitizations:



$$\varepsilon = 0.9 \implies [tr]_{0.9} = \langle (0, a), (1, b) \rangle$$



$$\varepsilon = 0.5 \implies [tr]_{0.5} = \langle (0, a), (2, b) \rangle$$



$$\varepsilon = 0.1 \implies [tr]_{0.1} = \langle (1, a), (2, b) \rangle$$

Note that $\langle (1, a), (1, b) \rangle$ is **not** a possible digitization of tr!

Digitization Closure Properties

Definitions (Henzinger, Manna, Pnueli):

- A set A of timed traces is **closed under digitization** if, for all $\varepsilon \in [0, 1]$, $[A]_\varepsilon \subseteq A$.
- A set S of timed traces is **closed under inverse digitization** if, **whenever** a timed trace tr is such that $[tr]_\varepsilon \in S$ for every value of ε, **then** $tr \in S$.

  Contrapositive: S closed under inverse digitization means that, if $tr \notin S$, then there exists ε such that $[tr]_\varepsilon \notin S$.

Examples

- Is the set $\{(t, a) \mid 0 \leqslant t < 2\}$ closed under digitization? Under inverse digitization?
- $\{(t, a) \mid t \in [1, 2] \cup [3, 5]\}$?
- $\{tr \mid error$ does not occur in $tr\}$?
- $\{\langle(2n, a)\rangle \mid n \in \mathbb{N}\}$?

In general:

Most specifications in practice are closed under inverse digitization. In particular, **bounded-invariance** and **bounded-response properties** are.

Closed timed automata are closed under digitization.

Open timed automata are closed under inverse digitization.

Mixed timed automata:

Clock constraints are boolean combinations of

$$\mathrm{true} \mid x < k \mid x \leqslant k \mid x > k \mid x \geqslant k$$

where $k \in \mathbb{N}$.

Closed timed automata:

Constraints are positive ($\wedge$, $\vee$, no negation) boolean combinations of

$$\mathrm{true} \mid x \leqslant k \mid x \geqslant k \ .$$

Open timed automata:

Constraints are positive ($\wedge$, $\vee$, no negation) boolean combinations of

$$\mathrm{true} \mid x < k \mid x > k \ .$$

Verification

Theorem (Henzinger, Manna, Pnueli):

Let A be a set of timed traces that is closed under digitization, and let S be a set of timed traces that is closed under inverse digitization. Then $A\restriction\mathbb{Z} \subseteq S\restriction\mathbb{Z} \iff A \subseteq S$.

Theorem:

Closed timed automata are closed under digitization. Moreover, any timed automaton can be infinitesimally conservatively approximated by a closed timed automaton.

Verification

Corollary:

Let S be a specification that is closed under inverse digitization. Then, for any closed timed automaton A,

$$A \models_{\mathbb{Z}} S \iff A \models_{\mathbb{R}} S \ .$$

This result shows that digitization is a **sound and complete** technique for verifying specifications that are closed under inverse digitization on closed timed automata.

Model Checking

How do we check that $A \models_{\mathbb{R}} S$ using FDR?[a]

1. Make sure that S is closed under inverse digitization!
2. Express $S\restriction\mathbb{Z}$ using CSP + $tock$.
3. Express $A\restriction\mathbb{Z}$ using CSP + $tock$.
4. Verify whether $S\restriction\mathbb{Z} \sqsubseteq A\restriction\mathbb{Z}$ using FDR + priority.

Case studies have discovered three bugs...

[a] FDR is a commercial product of Formal Systems (Europe) Ltd.

Full Abstraction

Definition:

Let A, B be two closed timed automata. Define $A \simeq B$ if, for all specifications S that are closed under inverse digitization,

$$A \vDash_{\mathbb{R}} S \iff B \vDash_{\mathbb{R}} S \ .$$

Theorem:

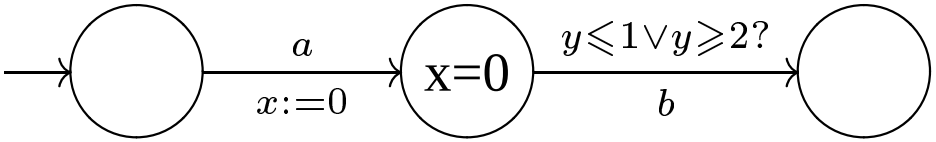
The semantics $\mathbb{Z}[\![-]\!]$ is fully abstract with respect to specifications closed under inverse digitization. In other words,

$$A \simeq B \iff \mathbb{Z}[\![A]\!] = \mathbb{Z}[\![B]\!] \ .$$

Intuitively, this indicates that integral behaviors contain no superfluous information—you cannot do with a simpler model.

Non-Safety Specifications

Consider $S =$ 'There are no timelocks'.



The integral behaviors exhibit no timelock, but the dense-time ones do!

How can we extend digitization techniques to LTL/CTL?

Note that the half-integral behaviors exhibit a timelock.

In Timed CSP, the onset of e.g. timelocks, originates from τ-transitions. Therefore (integral) digitization applies in this case.

Other Directions

- What about considering timesteps larger than 1 (CEGAR)?

  E.g., start with timesteps of size 32, then keep dividing by 2 as needed.

- Digitization techniques should work for rectangular hybrid automata: choose units so that all rates are integral multiples of *dt*. If all differential inclusions are closed, implementation will be closed under digitization.

- More case studies of standard protocols, and other comparisons with UppAal etc. are needed.