Timed automata

models, languages, dynamics

Eugene Asarin

LIAFA - University Paris Diderot and CNRS

PIMS/EQINOCS Workshop on Automata Theory and Symbolic Dynamics

Context

Timed automata

- A model for verification of real-time systems
- Invented by Alur and Dill in early 1990s
- Precursors: time Petri nets (Bethomieu)
- Now: an efficient model for verification, supported by tools (UPPAAL)
- A popular researh topic (¿8000 citation for papers by Alur and Dill)
  - modeling and verification
  - decidability and algorithmics
  - automata and language theory
  - very recent: dynamics
- Inspired by TA: hybrid automata, data automata, automata on nominal sets

Outline

Before we begin: timed words and languages

- A *word*: $u = abbabb$ represents a sequence of events in some Σ.

Before we begin: timed words and languages

- A *word*: $u = abbabb$ represents a sequence of events in some Σ.
- A *timed word*: $w = 0.8a2.66b1.5b0a3.14159b2.71828b$ represents a sequence of events and delays.

Before we begin: timed words and languages

- A *word*: $u = abbabb$ represents a sequence of events in some Σ.
- A *timed word*: $w = 0.8a2.66b1.5b0a3.14159b2.71828b$ represents a sequence of events and delays.
- It lives in a *timed monoid* $\Sigma^* \oplus \mathbb{R}_+$,

Before we begin: timed words and languages

- A *word*: $u = abbabb$ represents a sequence of events in some Σ.
- A *timed word*: $w = 0.8a2.66b1.5b0a3.14159b2.71828b$ represents a sequence of events and delays.
- It lives in a *timed monoid* $\Sigma^* \oplus \mathbb{R}_+$,but forget about it
- For us it sits in $(\mathbb{R}_+ \times \Sigma)^*$ (words on an infinite alphabet), that is $w =$ $(0.8, a), (2.66, b), (1.5, b), (0, a), (3.14159, b), (2.71828, b)$.

Before we begin: timed words and languages

- A *word*: $u = abbabb$ represents a sequence of events in some Σ.
- A *timed word*: $w = 0.8a2.66b1.5b0a3.14159b2.71828b$ represents a sequence of events and delays.
- It lives in a *timed monoid* $\Sigma^* \oplus \mathbb{R}_+$,but forget about it
- For us it sits in $(\mathbb{R}_+ \times \Sigma)^*$ (words on an infinite alphabet), that is $w =$ $(0.8, a), (2.66, b), (1.5, b), (0, a), (3.14159, b), (2.71828, b)$.
- Geometrically w is a point in several copies of $\mathbb{R}^n$:

$$w = (0.8, 2.66, 1.5, 0, 3.14159, 2.71828) \in \mathbb{R}^6_{abbabb}$$

Before we begin: timed words and languages

- A *word*: $u = abbabb$ represents a sequence of events in some Σ.
- A *timed word*: $w = 0.8a2.66b1.5b0a3.14159b2.71828b$ represents a sequence of events and delays.
- It lives in a *timed monoid* $\Sigma^* \oplus \mathbb{R}_+$,but forget about it
- For us it sits in $(\mathbb{R}_+ \times \Sigma)^*$ (words on an infinite alphabet), that is $w =$ $(0.8, a), (2.66, b), (1.5, b), (0, a), (3.14159, b), (2.71828, b)$.
- Geometrically w is a point in several copies of $\mathbb{R}^n$:

$$w = (0.8, 2.66, 1.5, 0, 3.14159, 2.71828) \in \mathbb{R}^6_{abbabb}$$

- A *timed language* is a set of timed words - examples below.

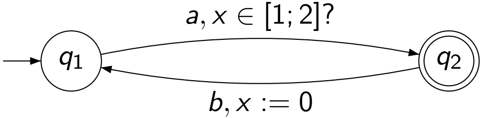
What are TA

Recipe: how to make a timed automaton

- take a finite automaton
- put it into continuous time
- add some variables $x_1, \ldots, x_n$, called clocks.
- make all them run: $\dot{x}_i = 1$ everywhere.
- add guards to some transitions (e.g. $x_3 < 7$)
- add resets to some transitions (e.g. $x_2 := 0$)
- serve and enjoy!

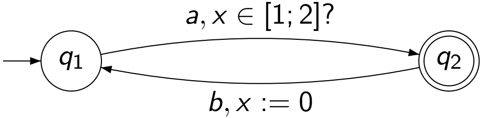
An example of a timed automaton

- Timed automaton (we forget to write $\dot{x} = 1$):



An example of a timed automaton

- Timed automaton (we forget to write $\dot{x} = 1$):

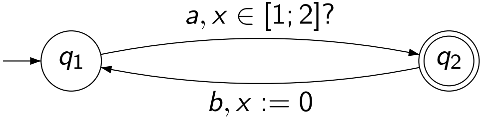


- Its run

$$(q_1, 0) \xrightarrow{1.83} (q_1, 1.83) \xrightarrow{a} (q_2, 1.83) \xrightarrow{4.1} (q_2, 5.93) \xrightarrow{b} (q_1, 0) \xrightarrow{1} (q_1, 1)$$

An example of a timed automaton

- Timed automaton (we forget to write $\dot{x} = 1$):



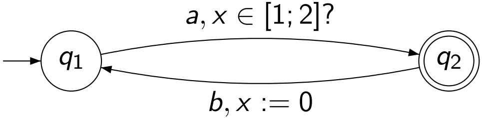
- Its run

$(q_1, 0) \xrightarrow{1.83} (q_1, 1.83) \xrightarrow{a} (q_2, 1.83) \xrightarrow{4.1} (q_2, 5.93) \xrightarrow{b} (q_1, 0) \xrightarrow{1} (q_1, 1)$

- Its *trace* $1.83\, a\, 4.1\, b\, 1\, a$ *a timed word*

An example of a timed automaton

- Timed automaton (we forget to write $\dot{x} = 1$):



- Its run

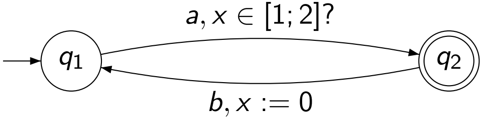
$$(q_1, 0) \xrightarrow{1.83} (q_1, 1.83) \xrightarrow{a} (q_2, 1.83) \xrightarrow{4.1} (q_2, 5.93) \xrightarrow{b} (q_1, 0) \xrightarrow{1} (q_1, 1)$$

- Its *trace* $1.83\,a\,4.1\,b\,1\,a$ a *timed word*
- Its *timed language*: set of all the traces starting in q_1, ending in q_2:

$$\{t_1\, a\, s_1\, b\, t_2\, a\, s_2\, b \ldots t_n\, a \mid \forall i . t_i \in [1;2]\}$$

An example of a timed automaton

- Timed automaton (we forget to write $\dot{x} = 1$):



- Its run

$$(q_1, 0) \xrightarrow{1.83} (q_1, 1.83) \xrightarrow{a} (q_2, 1.83) \xrightarrow{4.1} (q_2, 5.93) \xrightarrow{b} (q_1, 0) \xrightarrow{1} (q_1, 1)$$

- Its *trace* $1.83\, a\, 4.1\, b\, 1\, a$ *a timed word*
- Its *timed language*: set of all the traces starting in q_1, ending in q_2:

$$\{t_1\, a\, s_1\, b\, t_2\, a\, s_2\, b \ldots t_n\, a \mid \forall i. t_i \in [1; 2]\}$$

Observation

Clock value of x: time since the last reset of x.

Some simple exercises

Draw timed automata for specifications:

- Request a arrives every 5 minutes.

Some simple exercises

Draw timed automata for specifications:

- Request a arrives every 5 minutes.
- Request a arrives every 5 to 7 minutes.

Some simple exercises

Draw timed automata for specifications:

- Request a arrives every 5 minutes.
- Request a arrives every 5 to 7 minutes.
- a arrives every 5 to 7 minutes; and b arrives every 3 to 10 minutes.

Some simple exercises

Draw timed automata for specifications:

- Request a arrives every 5 minutes.
- Request a arrives every 5 to 7 minutes.
- a arrives every 5 to 7 minutes; and b arrives every 3 to 10 minutes.
- Request a is serviced within 2 minutes by c or rejected within 1 minute by r.

Some simple exercises

Draw timed automata for specifications:

- Request a arrives every 5 minutes.
- Request a arrives every 5 to 7 minutes.
- a arrives every 5 to 7 minutes; and b arrives every 3 to 10 minutes.
- Request a is serviced within 2 minutes by c or rejected within 1 minute by r.
- The same, but a arrives every 5 to 7 minutes.

Modeling exercise 2

Scheduling

Schedule two jobs on one CPU and one printer with a total execution time up to 16 minutes.

- Job 1 : Compute (10 min); Print (5 min)
- Job 2 : Download (3 min); Compute (1 min); Print (2 min)

Try it :

1. without preemption;
2. with preemptible computing.

Main theorem

Theorem (Alur, Dill)

Reachability is decidable for timed automata.

Main theorem

Theorem (Alur, Dill)

Reachability is decidable for timed automata.

Classical formulation

Empty language problem is decidable for TA

Main theorem

Theorem (Alur, Dill)

Reachability is decidable for timed automata.

Classical formulation

Empty language problem is decidable for TA

Both are the same

Non-empty language $\Leftrightarrow$ Reach(Init,Fin)

Proof idea

- Split the state space $Q \times \mathbb{R}^n$ into regions s.t.
  - all the states in one region have the same behavior;
  - there are finitely many regions;

Proof idea

- Split the state space $Q \times \mathbb{R}^n$ into regions s.t.
  - all the states in one region have the same behavior;
  - there are finitely many regions;
- Build a region automaton (its states are regions)

Proof idea

- Split the state space $Q \times \mathbb{R}^n$ into regions s.t.
  - all the states in one region have the same behavior;
  - there are finitely many regions;
- Build a finite region automaton (its states are regions)

Proof idea

- Split the state space $Q \times \mathbb{R}^n$ into regions s.t.
  - all the states in one region have the same behavior;
  - there are finitely many regions;
- Build a finite region automaton (its states are regions)
- Test reachability in this region automaton.

Proof idea

- Split the state space $Q \times \mathbb{R}^n$ into regions s.t.
  - all the states in one region have the same behavior;
  - there are finitely many regions;
- Build a finite region automaton (its states are regions)
- Test reachability in this region automaton.

Two difficulties

- What does it mean: the same behavior?
- How to invent it?

Proof idea

- Split the state space $Q \times \mathbb{R}^n$ into regions s.t.
  - all the states in one region have the same behavior;
  - there are finitely many regions;
- Build a finite region automaton (its states are regions)
- Test reachability in this region automaton.

Two difficulties

- What does it mean: the same behavior? Bisimulation.
- How to invent it? A&D invented it using ideas of Berthomieu (Time Petri nets). In fact it is rather natural.

Region equivalence

Definition

Two states of a TA are region equivalent: $(q, \mathbf{x}) \approx (p, \mathbf{y})$ if

- Same location: $p = q$
- Same integer parts of clocks: $\forall i\, (\lfloor x_i \rfloor = \lfloor y_i \rfloor)$
- Same order of fractional parts of clocks
  $\forall i, j\, (\{x_i\} < \{x_j\} \Leftrightarrow \{y_i\} < \{y_j\})$

Look at the picture!

Region equivalence

Definition

Two states of a TA are region equivalent: $(q, \mathbf{x}) \approx (p, \mathbf{y})$ if

- Same location: $p = q$
- Same integer parts of clocks: $\forall i \, (\lfloor x_i \rfloor = \lfloor y_i \rfloor)$
- Same order of fractional parts of clocks $\forall i, j \, (\{x_i\} < \{x_j\} \Leftrightarrow \{y_i\} < \{y_j\})$

Look at the picture!

Region equivalence

Definition

Two states of a TA are region equivalent: $(q, \mathbf{x}) \approx (p, \mathbf{y})$ if

- Same location: $p = q$
- Same integer parts of clocks: $\forall \text{small} i\, (\lfloor x_i \rfloor = \lfloor y_i \rfloor)$
- Same order of fractional parts of clocks $\forall \text{small} i, j\, (\{x_i\} < \{x_j\} \Leftrightarrow \{y_i\} < \{y_j\})$
- Or they are both big : $\forall i\, ((x_i > M) \Leftrightarrow (y_i > M))$

Look at the picture!

Region equivalence

Definition

Two states of a TA are region equivalent: $(q, \mathbf{x}) \approx (p, \mathbf{y})$ if

- Same location: $p = q$
- Same integer parts of clocks: $\forall \text{small} i \, (\lfloor x_i \rfloor = \lfloor y_i \rfloor)$
- Same order of fractional parts of clocks $\forall \text{small} i, j \, (\{x_i\} < \{x_j\} \Leftrightarrow \{y_i\} < \{y_j\})$
- Or they are both big : $\forall i \, ((x_i > M) \Leftrightarrow (y_i > M))$

Look at the picture!

Definition

Equivalence classes of $\approx$ are called regions.

Region equivalence

Definition

Two states of a TA are region equivalent: $(q, \mathbf{x}) \approx (p, \mathbf{y})$ if

- Same location: $p = q$
- Same integer parts of clocks: $\forall \text{small} i \, (\lfloor x_i \rfloor = \lfloor y_i \rfloor)$
- Same order of fractional parts of clocks $\forall \text{small} i, j \, (\{x_i\} < \{x_j\} \Leftrightarrow \{y_i\} < \{y_j\})$
- Or they are both big : $\forall i \, ((x_i > M) \Leftrightarrow (y_i > M))$

Look at the picture!

Definition

Equivalence classes of $\approx$ are called regions.

Lemma (Region equivalence is a bisimulation)

Equivalent states can make the same transitions, and arrive to equivalent states.

Decision algorithm

- Build a region automaton RA
  - States are regions.
  - There is a transition $r_1 \xrightarrow{a} r_2$ if some (all) element of r_1 can go to some element of r_2 on a.
  - There is a transition $r_1 \xrightarrow{\tau} r_2$ if some (all) element of r_1 can go to some element of r_2 on some $t > 0$

Decision algorithm

- Build a region automaton RA
  - States are regions.
  - There is a transition $r_1 \xrightarrow{a} r_2$ if some (all) element of r_1 can go to some element of r_2 on a.
  - There is a transition $r_1 \xrightarrow{\tau} r_2$ if some (all) element of r_1 can go to some element of r_2 on some $t > 0$
- Check whether some final region in RA is reachable from initial region.

Outline

1. TA: the model
   - Decidability
2. Timed language theory
3. Timed symbolic dynamics
4. Conclusions

Closure properties

Definition

Timed regular language is a language accepted by a TA

Closure properties

Definition

Timed regular language is a language accepted by a TA

Theorem

Timed regular languages are closed under $\cap, \cup$*, projection, but not complementation.*

Closure properties

Definition

Timed regular language is a language accepted by a TA

Theorem

Timed regular languages are closed under $\cap, \cup$, *projection, but not complementation.*

Fact

Determinization impossible for timed automata.

Decidability properties

Definition

Timed regular language (TRL) is a language accepted by a TA

Decidability properties

Definition

Timed regular language (TRL) is a language accepted by a TA

Theorem

Decidable for TRL (represented by TA): $L = \emptyset$, $w \in L$, $L \cap M = \emptyset$.

Decidability properties

Definition

Timed regular language (TRL) is a language accepted by a TA

Theorem

Decidable for TRL (represented by TA): $L = \emptyset$, $w \in L$, $L \cap M = \emptyset$.

Proof.

Immediate from Alur&Dill's theorem. □

Decidability properties

Definition
Timed regular language (TRL) is a language accepted by a TA

Theorem
Decidable for TRL (represented by TA): $L = \emptyset$, $w \in L$, $L \cap M = \emptyset$.

Theorem
Undecidable for TRL (represented by TA): L *universal (contains all the timed words),* $L \subset M$, $L = M$.

Decidability properties

Definition

Timed regular language (TRL) is a language accepted by a TA

Theorem

Decidable for TRL (represented by TA): $L = \emptyset$, $w \in L$, $L \cap M = \emptyset$.

Theorem

Undecidable for TRL (represented by TA): L universal (contains all the timed words), $L \subset M$, $L = M$.

Proof.

Encoding of runs of Minsky Machine as a timed languages. □

Reminder: regular expressions

Definition

Regular expressions: $E ::= 0 \mid \varepsilon \mid a \mid E + E \mid E \cdot E \mid E^*$

Theorem (Kleene)

Finite automata and regular expression define the same class of languages.

Reminder: regular expressions

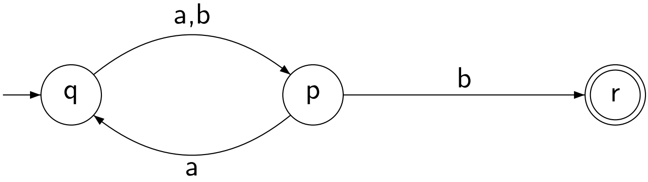
Definition

Regular expressions: $E ::= 0 \mid \varepsilon \mid a \mid E + E \mid E \cdot E \mid E^*$

Theorem (Kleene)

Finite automata and regular expression define the same class of languages.

Example



$$((a+b)a)^*(a+b)b$$

Timed regular expressions

A natural question

How to define regular expressions for timed languages?

Timed regular expressions

A natural question

How to define regular expressions for timed languages?

$$E ::= 0 \mid \varepsilon \mid \underline{\mathbf{t}} \mid a \mid E + E \mid E \cdot E \mid E^* \mid \langle E \rangle_I \mid E \wedge E \mid [a \mapsto z]E$$

Timed regular expressions

A natural question

How to define regular expressions for timed languages?

$$E ::= 0 \mid \varepsilon \mid \underline{\mathbf{t}} \mid a \mid E+E \mid E\cdot E \mid E^* \mid \langle E\rangle_I \mid E \wedge E \mid [a \mapsto z]E$$

Semantics:

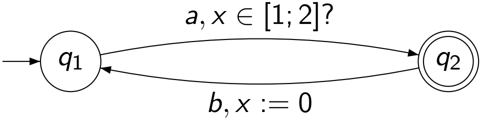
$$\|\underline{\mathbf{t}}\| = \mathbb{R}_{\geq 0} \quad \|a\| = \{a\} \qquad \|0\| = \emptyset \quad \|\varepsilon\| = \{\varepsilon\}$$

$$\|E_1 \cdot E_2\| = \|E_1\| \cdot \|E_2\| \qquad \|E_1 + E_2\| = \|E_1\| \cup \|E_2\|$$

$$\|\langle E\rangle\|_I = \{\sigma \in \|E\| \mid \ell(\sigma) \in I\} \qquad \|E^*\| = \|E\|^*$$

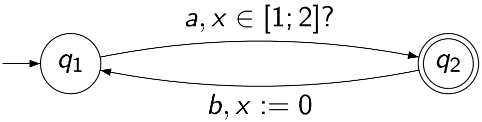
$$\|E_1 \wedge E_2\| = \|E_1\| \cap \|E_2\| \qquad \|[a \mapsto z]E\| = [a \mapsto z]\|E\|$$

A good example and a theorem



$$\{L = \{t_1\, a\, s_1\, b\, t_2\, a\, s_2\, b \ldots t_n\, a \mid \forall i . t_i \in [1;2]\}$$

A good example and a theorem



$$\{L = \{t_1\, a\, s_1\, b\, t_2\, a\, s_2\, b \ldots t_n\, a \mid \forall i . t_i \in [1;2]\}$$

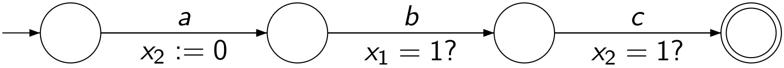
An expression for L : $\left(\langle \underline{\mathbf{t}} a \rangle_{[1;2]} \underline{\mathbf{t}} b\right)^*$

Theorem (A., Caspi, Maler,95)

Timed Automata and Timed regular expressions (with $\wedge$ and $[a \mapsto z]$) define the same class of timed languages

A nasty example

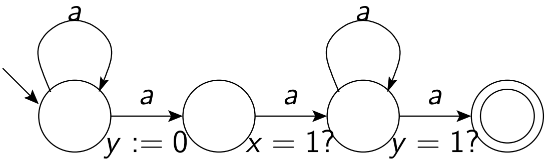
Intersection needed [ACM]



$$\{t_1 a t_2 b t_3 c \mid t_1 + t_2 = 1, t_2 + t_3 = 1\} = \underline{\mathbf{t}} a \langle \underline{\mathbf{t}} b \underline{\mathbf{t}} c \rangle_1 \wedge \langle \underline{\mathbf{t}} a \underline{\mathbf{t}} b \rangle_1 \underline{\mathbf{t}} c$$

Another nasty example

Renaming needed [Herrmann]



$$[b \mapsto a]\Big((\underline{\mathbf{t}}a)^*\langle \underline{\mathbf{t}}b(\underline{\mathbf{t}}a)^*\rangle_1 \wedge \langle(\underline{\mathbf{t}}a)^*\underline{\mathbf{t}}b\rangle_1(\underline{\mathbf{t}}a)^*\Big).$$

Outline

Preliminary considerations

- Aim: describe timed regular languages as dynamical systems
- Solution: from Nicolas Basset's MSc thesis (2010)
- Limitations: deterministic automata, bounded intervals between events

Shifts

Full timed shift

- Fix Σ an alphabet and $M \in \mathbb{R}$.
- Let $C = \Sigma \times [0; M]$ - a compact set (with a natural metrics).
- $S = C^{\mathbb{Z}}$: set of bi-infinite timed words (with a natural metrics). E.g. $\dots t_{-1}a_{-1}t_0a_0t_1a_1t_2\dots$
- Shift $\sigma : S \to S$ E.g. $\sigma(\{c_n\}) = \{c_{n+1}\}$.

Shifts

Full timed shift

- Fix Σ an alphabet and $M \in \mathbb{R}$.
- Let $C = \Sigma \times [0; M]$ - a compact set (with a natural metrics).
- $S = C^{\mathbb{Z}}$: set of bi-infinite timed words (with a natural metrics). E.g. $\dots t_{-1} a_{-1} t_0 a_0 t_1 a_1 t_2 \dots$
- Shift $\sigma : S \to S$ E.g. $\sigma(\{c_n\}) = \{c_{n+1}\}$.

The simplest example: studied by Weiss and Lindenstrauss!

- $\Sigma = \{a\}; M = 1; C \cong [0; 1]$
- $S = [0; 1]^{\mathbb{Z}} = \{t_{-1}, t_0, t_1, \dots\}$

Timed subshifts

Definition

A subshift $X \subset S$ with X closed and σ-invariant.

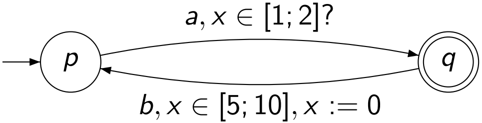
Two standard ways to define subshifts

- by an open set of forbidden patterns;
- by a closed set of allowed patterns for some lengths.

A timed automaton

- which is deterministic, w/o initial final states;
- with closed guards;
- corresponds to a timed subshift;
- we call it sofic!

Timed dynamics – an example



exercice

Compute forbidden patterns.

What about entropy?

We try to compute it for $[0;1]^{\mathbb{Z}}$

- Define $C(n, \epsilon)$: size of ϵ-net in $[0;1]^n$.
- Compute it: $C(n, \epsilon) = (1/\epsilon)^n$.

What about entropy?

We try to compute it for $[0;1]^{\mathbb{Z}}$

- Define $C(n,\epsilon)$: size of ϵ-net in $[0;1]^n$.
- Compute it: $C(n,\epsilon) = (1/\epsilon)^n$.
- Growth rate of C:

$$h_\epsilon = \lim_n \frac{\log C(n,\epsilon)}{n} = \log \frac{1}{\epsilon}.$$

- Entropy $h = \lim_{\epsilon\to 0} h_\epsilon =$

What about entropy?

We try to compute it for $[0;1]^{\mathbb{Z}}$

- Define $C(n,\epsilon)$: size of ϵ-net in $[0;1]^n$.
- Compute it: $C(n,\epsilon) = (1/\epsilon)^n$.
- Growth rate of C:

$$h_\epsilon = \lim_n \frac{\log C(n,\epsilon)}{n} = \log \frac{1}{\epsilon}.$$

- Entropy $h = \lim_{\epsilon \to 0} h_\epsilon = \infty$

What about entropy?

We try to compute it for $[0;1]^{\mathbb{Z}}$

- Define $C(n,\epsilon)$: size of ϵ-net in $[0;1]^n$.
- Compute it: $C(n,\epsilon) = (1/\epsilon)^n$.
- Growth rate of C:

$$h_\epsilon = \lim_n \frac{\log C(n,\epsilon)}{n} = \log \frac{1}{\epsilon}.$$

- Entropy $h = \lim_{\epsilon\to 0} h_\epsilon = \infty$

The same is true for almost all reasonable timed subshifts.

Entropy renormalized

We still try!

Given a timed subshift L

- Define L_n -set of timed words of length n (btw $L_n \subset \mathbb{R}^n \times \Sigma^n$)
- Define $C(n, \epsilon)$: size of ϵ-net in L_n.
- Compute it: $C(n, \epsilon) = (1/\epsilon)^n \operatorname{Vol}(L_n)$.

Entropy renormalized

We still try!

Given a timed subshift L

- Define L_n -set of timed words of length n (btw $L_n \subset \mathbb{R}^n \times \Sigma^n$)
- Define $C(n,\epsilon)$: size of ϵ-net in L_n.
- Compute it: $C(n,\epsilon) = (1/\epsilon)^n \operatorname{Vol}(L_n)$.
- Growth rate of C:

$$h_\epsilon = \lim_n \frac{\log C(n,\epsilon)}{n} = \log \frac{1}{\epsilon} + \lim_n \frac{\log \operatorname{Vol}(L_n)}{n}.$$

- We call the last term *volumic entropy*:

$$H(L) = \lim_n \frac{\log \operatorname{Vol}(L_n)}{n}$$

Entropy renormalized

We still try!

Given a timed subshift L

- Define L_n -set of timed words of length n (btw $L_n \subset \mathbb{R}^n \times \Sigma^n$)
- Define $C(n,\epsilon)$: size of ϵ-net in L_n.
- Compute it: $C(n,\epsilon) = (1/\epsilon)^n \operatorname{Vol}(L_n)$.
- Growth rate of C:

$$h_\epsilon = \lim_n \frac{\log C(n,\epsilon)}{n} = \log\frac{1}{\epsilon} + \lim_n \frac{\log \operatorname{Vol}(L_n)}{n}.$$

- We call the last term *volumic entropy*:

$$H(L) = \lim_n \frac{\log \operatorname{Vol}(L_n)}{n}$$

Information production in a timed subshift

Outline

It was in this talk

- Timed automata: a beautiful variant of automat coming from practice
- Infinite-state, but many questions decidable
- A non-trivial theory of languages
- Symbolic dynamics can be defined, many thing remain to study
- . . . in particular the flow

Advertisement

- Next talk (Aldric): how to characterize and compute the volumes and the volumic entropy.
- After that (Nicolas): MME for timed automata
- Merci EQINOCS, more results will follow