Strong Turing Completeness of Continuous Chemical Reaction Networks

Amaury Pouly

Joint work with Olivier Bournez, François Fages, Guillaume Le Guludec and Daniel Graça

10 october 2018

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N}, f = \text{rate})$

Example :

$$2\text{H} \quad + \quad \text{O} \quad \rightarrow \quad \text{H}_2\text{O}$$

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Example :

$$2\text{H} \quad + \quad \text{O} \quad \rightarrow \quad \text{H}_2\text{O}$$

Assumption : law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \quad \rightsquigarrow \quad f(y) = k \prod_i y_i^{a_i}$$

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \dots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N}, f = \text{rate})$

Example :

$$2\text{H} \quad + \quad \text{O} \quad \rightarrow \quad \text{H}_2\text{O}$$

Assumption : law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \quad \rightsquigarrow \quad f(y) = k \prod_i y_i^{a_i}$$

Semantics :

- discrete
- differential
- stochastic

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i$ $\quad (a_i, b_i \in \mathbb{N}$, $f =$ rate)

Example :

$$2H \;+\; O \;\rightarrow\; H_2O$$

Assumption : law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \;\rightsquigarrow\; f(y) = k \prod_i y_i^{a_i}$$

Semantics :

- discrete
- differential $\rightarrow$
- stochastic

$$y_i' = \sum_{\text{reaction } R} (b_i^R - a_i^R) f^R(y)$$

Example :

$$[H_2O]' = f(H_2O)$$

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N}, f = \text{rate})$

Example :

$$2\text{H} \quad + \quad \text{O} \quad \rightarrow \quad \text{H}_2\text{O}$$

Assumption : law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \quad \rightsquigarrow \quad f(y) = k \prod_i y_i^{a_i}$$

Semantics :

- discrete
- differential →
- stochastic

$$y_i' = \sum_{\text{reaction } R} (b_i^R - a_i^R) k^R \prod_j y_j^{a_j}$$

Example :

$$[\text{H}_2\text{O}]' = [\text{O}][\text{H}]^2$$

⇝ Polynomial ODE !

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Example :

$$
\begin{array}{cccccc}
2H & + & O & \rightarrow & H_2O \\
C & + & O_2 & \rightarrow & CO_2
\end{array}
$$

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Example :

$$\begin{array}{cccccc} 2H & + & O & \rightarrow & H_2O \\ C & + & O_2 & \rightarrow & CO_2 \end{array}$$

Not limited to simple chemical reactions :

- DNA strand displacement
- RNA
- protein reactions

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i$ $\quad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Example :

$$\begin{array}{rcccl} 2H & + & O & \rightarrow & H_2O \\ C & + & O_2 & \rightarrow & CO_2 \end{array}$$

Not limited to simple chemical reactions :

- DNA strand displacement
- RNA
- protein reactions

Implementing CRNs is a recent and active research field.

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Some reactions are unrealistic :

$$y_1 + 26y_2 + 7y_3 \to 13y_4 + y_5$$

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i$ $\qquad (a_i, b_i \in \mathbb{N}$, $f =$ rate)

Some reactions are unrealistic :

$$y_1 + 26y_2 + 7y_3 \rightarrow 13y_4 + y_5$$

Only consider elementary reactions : at most two reactants

- $A + B \xrightarrow{k} C$
- $A \xrightarrow{k} B + C$
- $A \xrightarrow{k} B$
- $A \xrightarrow{k} \varnothing$
- $\varnothing \xrightarrow{k} A$

Chemical Reaction Networks

A reaction system is a finite set of

- molecular species $y_1, \dots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i$ $\quad (a_i, b_i \in \mathbb{N}, f = \text{rate})$

Some reactions are unrealistic :

$$y_1 + 26y_2 + 7y_3 \rightarrow 13y_4 + y_5$$

Only consider elementary reactions : at most two reactants

- $A + B \xrightarrow{k} C$
- $A \xrightarrow{k} B + C$
- $A \xrightarrow{k} B$
- $A \xrightarrow{k} \varnothing$
- $\varnothing \xrightarrow{k} A$

Example : $A + B \xrightarrow{k} C$

$$A' = -kAB \qquad B' = -kAB \qquad C' = kAB$$

$\rightsquigarrow$ Quadratic ODE !

Can we use CRNs to compute ?

Chemical Reaction Networks : what can we compute ?

Can we use CRNs to compute ? What does it even mean ?

Can we use CRNs to compute ? What does it even mean ?

Chemical Reaction Networks : what can we compute ?

Can we use CRNs to compute ? What does it even mean ?

It depends a lot on how we define computability, in particular :

- rate : dependent/independent
- semantics : discrete/stochastic/differential
- kinetics : mass action/Michaelis/...
- species : finite/unbounded/infinite
- encoding : molecule count/concentration/digits
- more : robust, stable, ...

Chemical Reaction Networks : what can we compute ?

Can we use CRNs to compute ? What does it even mean ?

It depends a lot on how we define computability, in particular :

- rate : dependent/independent
- semantics : discrete/stochastic/differential
- kinetics : mass action/Michaelis/...
- species : finite/unbounded/infinite
- encoding : molecule count/concentration/digits
- more : robust, stable, ...

Extreme examples :

rate-independent, differential, any kinetics, finite species, value is concentration, stable

$\rightsquigarrow$ piecewise linear functions

Chemical Reaction Networks : what can we compute ?

Can we use CRNs to compute ? What does it even mean ?

It depends a lot on how we define computability, in particular :

- rate : dependent/independent
- semantics : discrete/stochastic/differential
- kinetics : mass action/Michaelis/...
- species : finite/unbounded/infinite
- encoding : molecule count/concentration/digits
- more : robust, stable, ...

Extreme examples :

rate-independent, differential, any kinetics, finite species, value is concentration, stable

$\rightsquigarrow$ piecewise linear functions

rate-dependent, stochastic, Markov, finite species, value is molecule count (must be small)

$\rightsquigarrow$ probabilistic Turing machine

Chemical Reaction Networks : main result

A reaction is elementary if it has at most two reactants
⇒ can, in principle, be implemented with DNA, RNA or proteins

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Chemical Reaction Networks : main result

A reaction is elementary if it has at most two reactants
$\Rightarrow$ can, in principle, be implemented with DNA, RNA or proteins

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Note : in fact the following elementary reactions suffice :

$$\emptyset \xrightarrow{k} x \qquad x \xrightarrow{k} x + z \qquad x + y \xrightarrow{k} x + y + z \qquad x + y \xrightarrow{k} \emptyset$$

Chemical Reaction Networks : main result

A reaction is elementary if it has at most two reactants
$\Rightarrow$ can, in principle, be implemented with DNA, RNA or proteins

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Note : in fact the following elementary reactions suffice :

$$\varnothing \xrightarrow{k} x \qquad x \xrightarrow{k} x+z \qquad x+y \xrightarrow{k} x+y+z \qquad x+y \xrightarrow{k} \varnothing$$

We can even say something about the complexity :

$f \in$ FPTIME $\Rightarrow$ CRN computes f in
- polynomial time&space

or equivalently
- polynomial length

Chemical Reaction Networks : mathematics

mass-action-law reaction system on finite universes of molecules under the differential semantics

$$\Updownarrow$$

Chemical Reaction Networks : mathematics

mass-action-law reaction system on finite universes of molecules under the differential semantics

$$\Updownarrow$$

Polynomial ODE :

$$\begin{cases} y_1' = p_1(y_1, \dots, y_n) \\ \vdots \\ y_n' = p_n(y_1, \dots, y_n) \end{cases}$$

with constraints :

- nonnegative values (concentration)
- restricted negative feedback : $x' = -xyz$

Chemical Reaction Networks : mathematics

Elementary mass-action-law reaction system on finite universes of molecules under the differential semantics

$$\Updownarrow$$

Polynomial ODE :

$$\begin{cases} y_1' = p_1(y_1, \ldots, y_n) \\ \vdots \\ y_n' = p_n(y_1, \ldots, y_n) \end{cases}$$

with constraints :

- nonnegative values (concentration)
- restricted negative feedback : $x' = -xyz$
- quadratic : $p_k(y) = \sum_{ij} \alpha_{ij} y_i y_j$

Chemical Reaction Networks : mathematics

Elementary mass-action-law reaction system on finite universes of molecules under the differential semantics

$$\Updownarrow$$

Polynomial ODE :

$$\begin{cases} y_1' = p_1(y_1, \ldots, y_n) \\ \vdots \\ y_n' = p_n(y_1, \ldots, y_n) \end{cases}$$

with constraints :

- nonnegative values (concentration)
- restricted negative feedback : $x' = -xyz$
- quadratic : $p_k(y) = \sum_{ij} \alpha_{ij} y_i y_j$

$\Updownarrow$ clever rewriting
value encoding : $y = y^+ - y^-$

Polynomial ODE : $y' = p(y)$

Chemical Reaction Networks : mathematics

Elementary mass-action-law reaction system on finite universes of molecules under the differential semantics

$$\Updownarrow$$

Polynomial ODE :

$$\begin{cases} y_1' = p_1(y_1, \ldots, y_n) \\ \vdots \\ y_n' = p_n(y_1, \ldots, y_n) \end{cases}$$

with constraints :

- nonnegative values (concentration)
- restricted negative feedback : $x' = -xyz$
- quadratic : $p_k(y) = \sum_{ij} \alpha_{ij} y_i y_j$

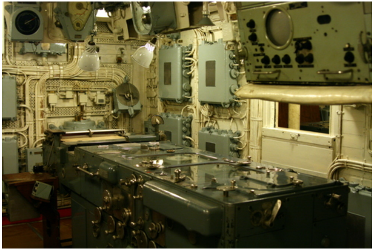
$\Updownarrow$ clever rewriting
value encoding : $y = y^+ - y^-$

Polynomial ODE : $y' = p(y)$

What can we compute with polynomial ODEs ?

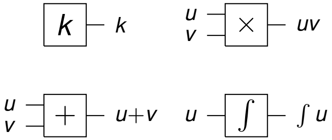
Analog Computers

Differential Analyser
“Mathematica of the 1920s”



Admiralty Fire Control Table
British Navy ships (WW2)

Polynomial Differential Equations



General Purpose
Analog Computer

Differential Analyzer

Newton mechanics

Reaction networks :

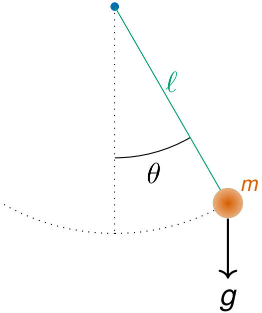
- chemical
- enzymatic

polynomial differential equations :

$$\begin{cases} y(0) = y_0 \\ y'(t) = p(y(t)) \end{cases}$$

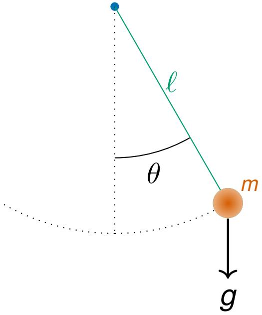
- Rich class
- Stable $(+, \times, \circ, /, \mathrm{ED})$
- No closed-form solution

Example of dynamical system



$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$

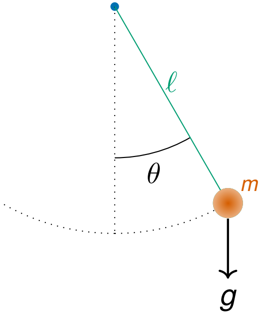
Example of dynamical system



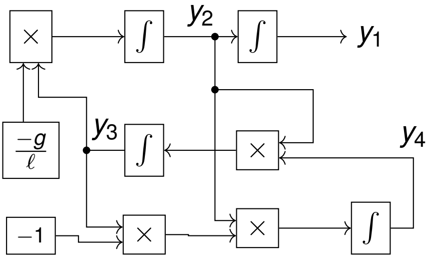
$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$

$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{g}{l} y_3 \\ y_3' = y_2 y_4 \\ y_4' = -y_2 y_3 \end{cases} \Leftrightarrow \begin{cases} y_1 = \theta \\ y_2 = \dot{\theta} \\ y_3 = \sin(\theta) \\ y_4 = \cos(\theta) \end{cases}$$

Example of dynamical system

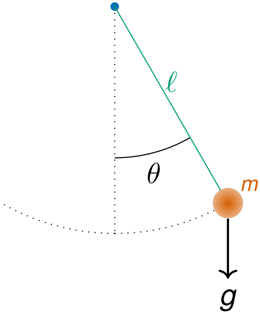


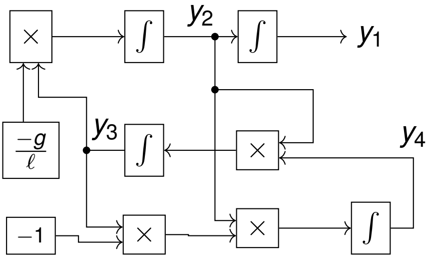
$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$



$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{g}{l} y_3 \\ y_3' = y_2 y_4 \\ y_4' = -y_2 y_3 \end{cases} \Leftrightarrow \begin{cases} y_1 = \theta \\ y_2 = \dot{\theta} \\ y_3 = \sin(\theta) \\ y_4 = \cos(\theta) \end{cases}$$

Example of dynamical system





$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$

$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{g}{l} y_3 \\ y_3' = y_2 y_4 \\ y_4' = -y_2 y_3 \end{cases} \Leftrightarrow \begin{cases} y_1 = \theta \\ y_2 = \dot{\theta} \\ y_3 = \sin(\theta) \\ y_4 = \cos(\theta) \end{cases}$$

Historical remark : the word “analog”

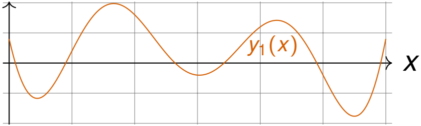
The pendulum and the circuit have the same equation. One can study one using the other by analogy.

Computing with differential equations

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



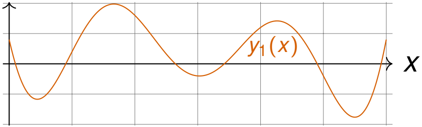
Shannon's notion

Computing with differential equations

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$f(x) = y_1(x)$



Shannon's notion

$\sin, \cos, \exp, \log, \ldots$

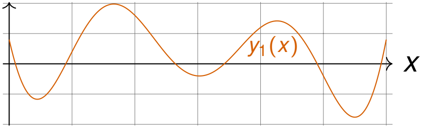
Strictly weaker than Turing machines [Shannon, 1941]

Computing with differential equations

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



Shannon's notion

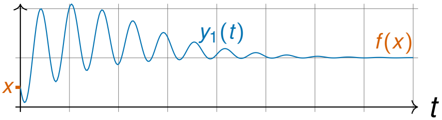
$\sin, \cos, \exp, \log, \ldots$

Strictly weaker than Turing machines [Shannon, 1941]

Computable

$$\begin{cases} y(0) = q(x) \\ y'(t) = p(y(t)) \end{cases} \qquad \begin{matrix} x \in \mathbb{R} \\ t \in \mathbb{R}_+ \end{matrix}$$

$$f(x) = \lim_{t \to \infty} y_1(t)$$



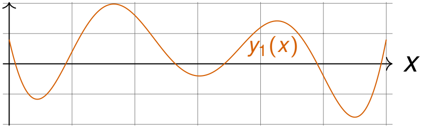
Modern notion

Computing with differential equations

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



Shannon's notion

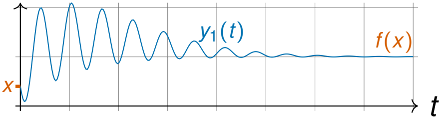
sin, cos, exp, log, ...

Strictly weaker than Turing machines [Shannon, 1941]

Computable

$$\begin{cases} y(0) = q(x) & x \in \mathbb{R} \\ y'(t) = p(y(t)) & t \in \mathbb{R}_+ \end{cases}$$

$$f(x) = \lim_{t \to \infty} y_1(t)$$



Modern notion

sin, cos, exp, log, Γ, ζ, ...

Turing powerful [Bournez et al., 2007]

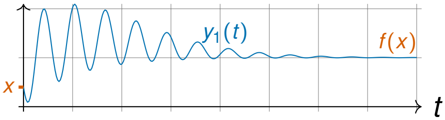
Equivalence with computable analysis

Definition (Bournez et al, 2007)

f **computable by GPAC** if $\exists p$ polynomial such that $\forall x \in [a,b]$

$$y(0) = (x, 0, \dots, 0) \qquad y'(t) = p(y(t))$$

satisfies $|f(x) - y_1(t)| \leqslant y_2(t)$ et $y_2(t) \xrightarrow[t\to\infty]{} 0$.



$y_1(t) \xrightarrow[t\to\infty]{} f(x)$

$y_2(t) =$ error bound

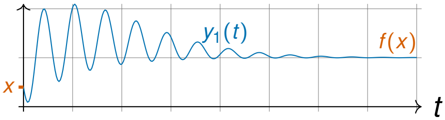
Equivalence with computable analysis

Definition (Bournez et al, 2007)

f **computable by GPAC** if $\exists p$ polynomial such that $\forall x \in [a, b]$

$$y(0) = (x, 0, \ldots, 0) \qquad y'(t) = p(y(t))$$

satisfies $|f(x) - y_1(t)| \leqslant y_2(t)$ et $y_2(t) \xrightarrow[t\to\infty]{} 0$.



$y_1(t) \xrightarrow[t\to\infty]{} f(x)$

$y_2(t) =$ error bound

Theorem (Bournez et al, 2007)

$f : [a, b] \to \mathbb{R}$ *computable*[1] $\Leftrightarrow$ f *computable by GPAC*

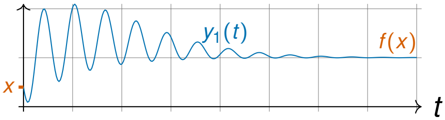
Equivalence with computable analysis

Definition (Bournez et al, 2007)

f **computable by GPAC** if $\exists p$ polynomial such that $\forall x \in [a, b]$

$$y(0) = (x, 0, \ldots, 0) \qquad y'(t) = p(y(t))$$

satisfies $|f(x) - y_1(t)| \leqslant y_2(t)$ et $y_2(t) \xrightarrow[t\to\infty]{} 0$.



$y_1(t) \xrightarrow[t\to\infty]{} f(x)$

$y_2(t) =$ error bound

Theorem (Bournez et al, 2007)

$f : [a, b] \to \mathbb{R}$ computable[1] *$\Leftrightarrow$ f computable by GPAC*

1. In Computable Analysis, a standard model over reals built from Turing machines.

Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x

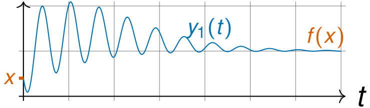
Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x
- GPAC :

Tentative definition

$T(x) = ??$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$



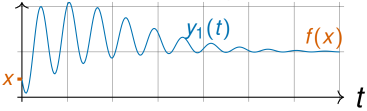
Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x
- GPAC :

Tentative definition

$T(x, \mu) =$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$



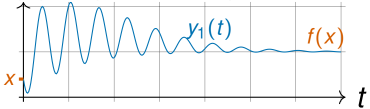
Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x
- GPAC :

Tentative definition

$T(x, \mu) =$ first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$



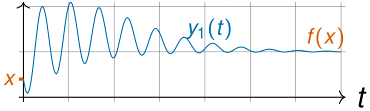
Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x
- GPAC :

Tentative definition

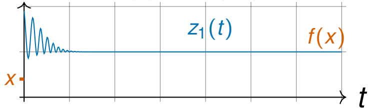
$T(x, \mu) =$ first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$



$\rightsquigarrow$

$z(t) = y(e^t)$



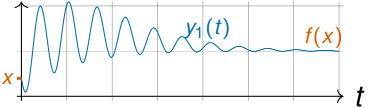
Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x
- GPAC :

Tentative definition

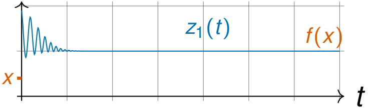
$T(x,\mu) =$ first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$

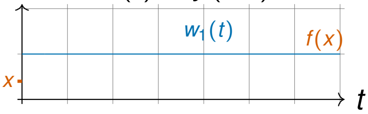


$\leadsto$

$z(t) = y(e^t)$



$w(t) = y(e^{e^t})$



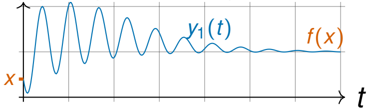
Complexity of analog systems

- Turing machines : $T(x)$ = number of steps to compute on x
- GPAC : time contraction problem $\rightarrow$ **open problem**

Tentative definition

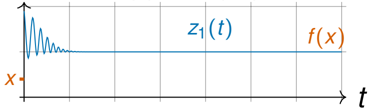
$T(x, \mu)$ = first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$

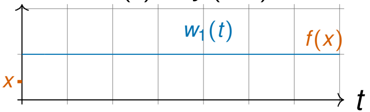


$\rightsquigarrow$

$z(t) = y(e^t)$



$w(t) = y(e^{e^t})$

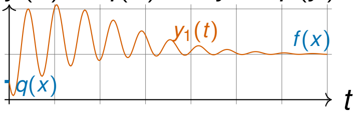


Something is wrong...

All functions have constant time complexity.

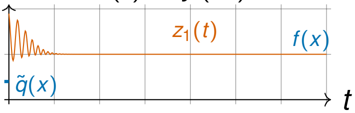
Time-space correlation of the GPAC

$y(0) = q(x) \qquad y' = p(y)$



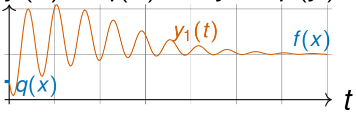
$\rightsquigarrow$

$z(t) = y(e^t)$



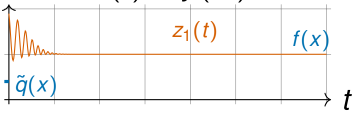
Time-space correlation of the GPAC

$y(0) = q(x) \qquad y' = p(y)$

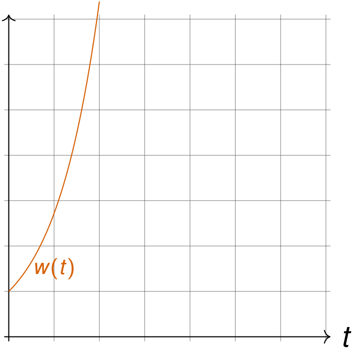


$\rightsquigarrow$

$z(t) = y(e^t)$

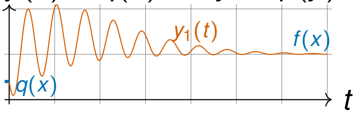


extra component : $w(t) = e^t$



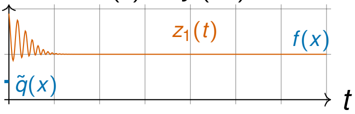
Time-space correlation of the GPAC

$y(0) = q(x) \qquad y' = p(y)$

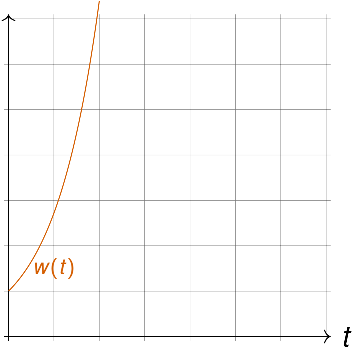


$\leadsto$

$z(t) = y(e^t)$



extra component : $w(t) = e^t$



Observation

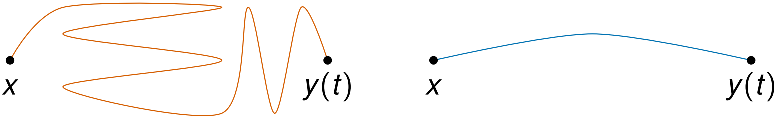
Time scaling costs “space”.

$\leadsto$

Time complexity for the GPAC must involve time and space !

Complexity of solving polynomial ODEs

$$y(0) = x \qquad y'(t) = p(y(t))$$



Complexity of solving polynomial ODEs

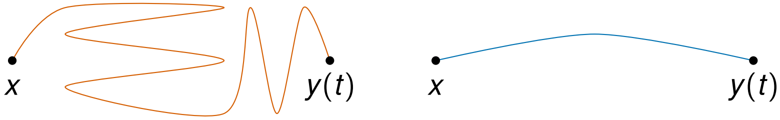
$$y(0) = x \qquad y'(t) = p(y(t))$$

Theorem

If $y(t)$ exists, one can compute p, q such that $\left|\frac{p}{q} - y(t)\right| \leqslant 2^{-n}$ in time

$$\operatorname{poly}(\textit{size of } x \textit{ and } p, n, \ell(t))$$

where $\ell(t) \approx$ length of the curve (between x and $y(t)$)

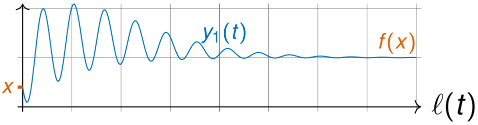


length of the curve = complexity = ressource

Characterization of real polynomial time

Definition : $f : [a,b] \to \mathbb{R}$ in ANALOG-P$_{\mathbb{R}}$ $\Leftrightarrow \exists p$ polynomial, $\forall x \in [a,b]$

$$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$$



Characterization of real polynomial time

Definition : $f : [a, b] \to \mathbb{R}$ in ANALOG-P$_{\mathbb{R}}$ $\Leftrightarrow$ $\exists p$ polynomial, $\forall x \in [a, b]$

$$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$$

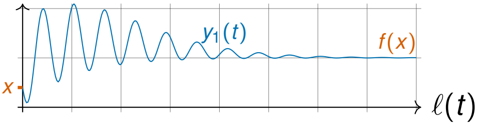
satisfies :

1. $|y_1(t) - f(x)| \leqslant 2^{-\ell(t)}$

   «greater length $\Rightarrow$ greater precision»

2. $\ell(t) \geqslant t$

   «length increases with time»



Characterization of real polynomial time

Definition : $f : [a,b] \to \mathbb{R}$ in ANALOG-P$_{\mathbb{R}}$ $\Leftrightarrow \exists p$ polynomial, $\forall x \in [a,b]$

$$y(0) = (x, 0, \dots, 0) \qquad y' = p(y)$$

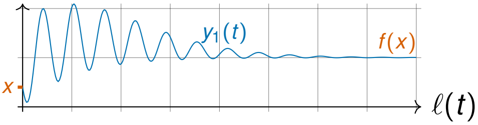
satisfies :

1. $|y_1(t) - f(x)| \leqslant 2^{-\ell(t)}$

   «greater length ⇒ greater precision»

2. $\ell(t) \geqslant t$

   «length increases with time»



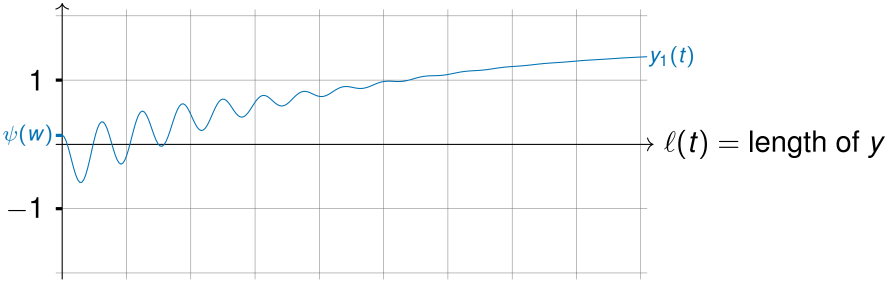
Theorem

$f : [a,b] \to \mathbb{R}$ computable in polynomial time $\Leftrightarrow f \in$ ANALOG-P$_{\mathbb{R}}$.

Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

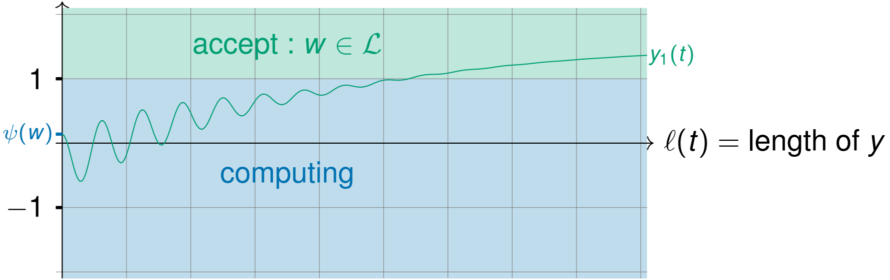
$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$



Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$



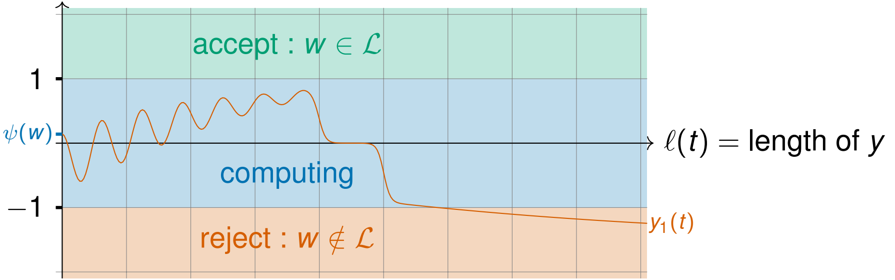
satisfies

1. if $y_1(t) \geqslant 1$ then $w \in \mathcal{L}$

Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$



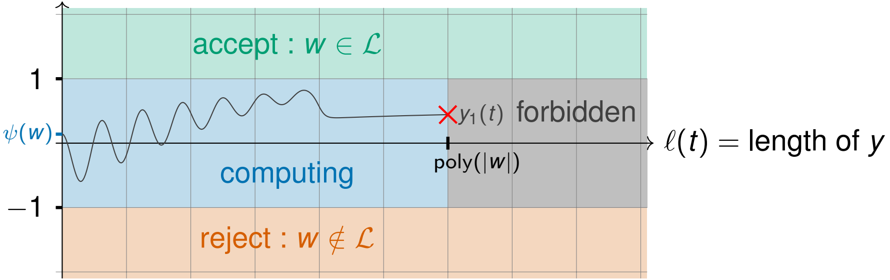
satisfies

2. if $y_1(t) \leqslant -1$ then $w \notin \mathcal{L}$

Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$



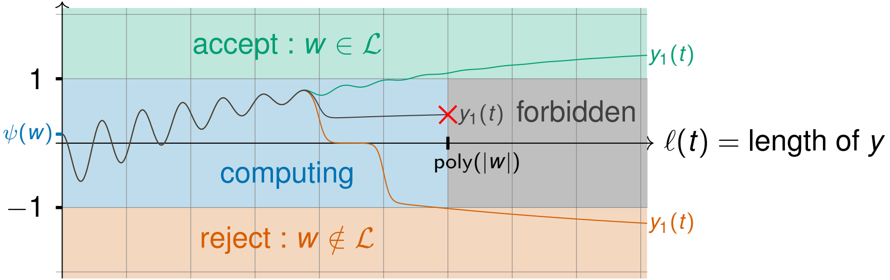
satisfies

3. if $\ell(t) \geqslant \text{poly}(|w|)$ then $|y_1(t)| \geqslant 1$

Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

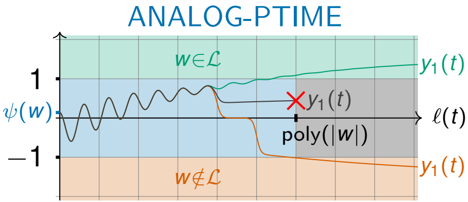
$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

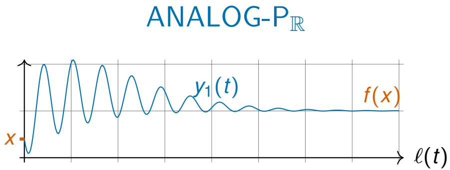


Theorem

PTIME = ANALOG-PTIME

Summary





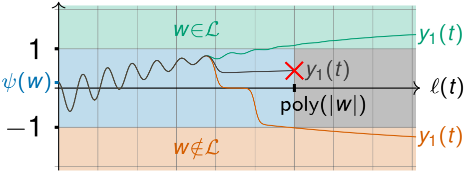
Theorem

- $\mathcal{L} \in$ PTIME *of and only if* $\mathcal{L} \in$ ANALOG-PTIME
- $f : [a, b] \to \mathbb{R}$ *computable in polynomial time* $\Leftrightarrow f \in$ ANALOG-$P_{\mathbb{R}}$

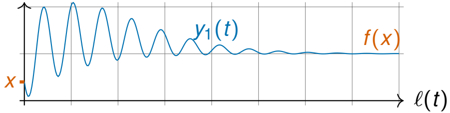
- Analog complexity theory based on length
- Time of Turing machine $\Leftrightarrow$ length of the GPAC
- Purely continuous characterization of PTIME

Summary

ANALOG-PTIME



ANALOG-$P_{\mathbb{R}}$



Theorem

- $\mathcal{L} \in$ PTIME *of and only if* $\mathcal{L} \in$ ANALOG-PTIME
- $f : [a, b] \to \mathbb{R}$ *computable in polynomial time* $\Leftrightarrow f \in$ ANALOG-$P_{\mathbb{R}}$

- Analog complexity theory based on length
- Time of Turing machine $\Leftrightarrow$ length of the GPAC
- Purely continuous characterization of PTIME
- Only rational coefficients needed

Back to Chemical Reaction Networks

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Is this really realistic ?

Back to Chemical Reaction Networks

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Is this really realistic ?

- we need precise reaction rates
- robust to noise ? $y' = p(y) + e$
- growth of the # molecules/volume

Back to Chemical Reaction Networks

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Is this really realistic ?

- we need precise reaction rates $\rightarrow$ no good answer (yet)
- robust to noise ? $y' = p(y) + e$
- growth of the # molecules/volume

Back to Chemical Reaction Networks

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Is this really realistic ?

- we need precise reaction rates $\rightarrow$ no good answer (yet)
- robust to noise ? $y' = p(y) + e$
- growth of the # molecules/volume

Two possible implementations/proof [2]

"Integer" encoding :

- exponential growth
- very robust : $|e| \leqslant 1$

Back to Chemical Reaction Networks

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Is this really realistic ?

- we need precise reaction rates $\rightarrow$ no good answer (yet)
- robust to noise ? $y' = p(y) + e$
- growth of the # molecules/volume

Two possible implementations/proof [2]

“Integer” encoding :

- exponential growth
- very robust : $|e| \leqslant 1$

“Rational” encoding :

- linear growth
- somewhat robust : if $|e| \leqslant e^{-N^c}$ can do $SPACE(\mathcal{O}(N))$

Back to Chemical Reaction Networks

Theorem (CMSB 2017)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Is this really realistic ?

- we need precise reaction rates $\rightarrow$ no good answer (yet)
- robust to noise ? $y' = p(y) + e$
- growth of the # molecules/volume

Two possible implementations/proof [2]

"Integer" encoding :

- exponential growth
- very robust : $|e| \leqslant 1$

"Rational" encoding :

- linear growth
- somewhat robust : if $|e| \leqslant e^{-N^c}$ can do $SPACE(\mathcal{O}(N))$

2. Disclaimer : not in the paper, I haven't checked the details.

Future work

Reaction networks :

- chemical
- enzymatic

$$y' = p(y)$$

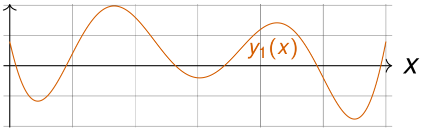
?

$$y' = p(y) + e(t)$$

- Finer time complexity (linear)
- Nondeterminism
- Robustness
- « Space» complexity
- Other models
- Stochastic

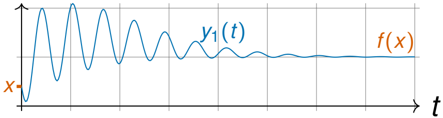
Universal differential equations

Generable functions



subclass of analytic functions

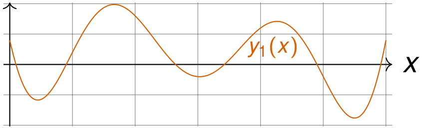
Computable functions



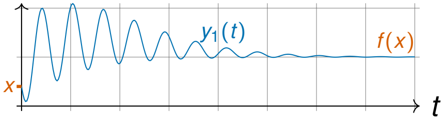
any computable function

Universal differential equations

Generable functions

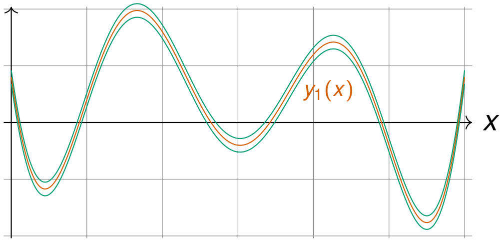


Computable functions

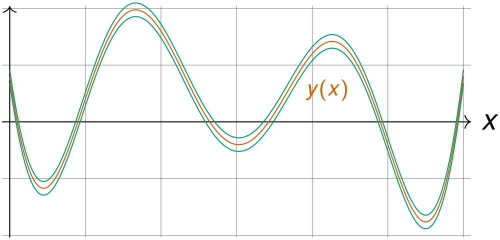


subclass of analytic functions

any computable function



Universal differential algebraic equation (DAE)



Theorem (Rubel, 1981)

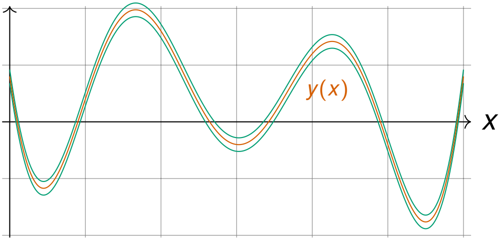
For any continuous functions f and ε, there exists $y : \mathbb{R} \to \mathbb{R}$ solution to

$$\begin{aligned} 3y'^4 y'' y''''^2 & -4y'^4 y'''^2 y'''' + 6y'^3 y''^2 y''' y'''' + 24y'^2 y''^4 y'''' \\ & -12y'^3 y'' y'''^3 - 29y'^2 y''^3 y'''^2 + 12y''^7 \qquad = 0 \end{aligned}$$

such that $\forall t \in \mathbb{R}$,

$$|y(t) - f(t)| \leqslant \varepsilon(t).$$

Universal differential algebraic equation (DAE)



Theorem (Rubel, 1981)

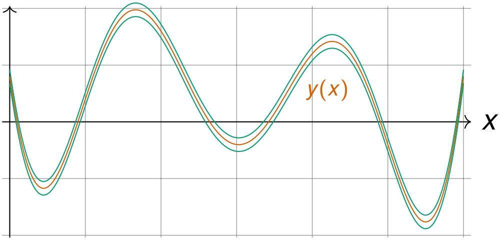
*There exists a **fixed** polynomial p and $k \in \mathbb{N}$ such that for any continuous functions f and ε, there exists a solution $y : \mathbb{R} \to \mathbb{R}$ to*

$$p(y, y', \ldots, y^{(k)}) = 0$$

such that $\forall t \in \mathbb{R}$,

$$|y(t) - f(t)| \leqslant \varepsilon(t).$$

Universal differential algebraic equation (DAE)



Theorem (Rubel, 1981)

*There exists a **fixed** polynomial p and $k \in \mathbb{N}$ such that for any continuous functions f and ε, there exists a solution $y : \mathbb{R} \to \mathbb{R}$ to*

$$p(y, y', \ldots, y^{(k)}) = 0$$

such that $\forall t \in \mathbb{R}$,

$$|y(t) - f(t)| \leqslant \varepsilon(t).$$

Problem : this is «weak» result.

The problem with Rubel's DAE

The solution y is not unique, **even with added initial conditions** :

$$p(y, y', \ldots, y^{(k)}) = 0, \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k$$

In fact, this is fundamental for Rubel's proof to work !

The problem with Rubel's DAE

The solution y is not unique, **even with added initial conditions** :

$$p(y, y', \ldots, y^{(k)}) = 0, \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k$$

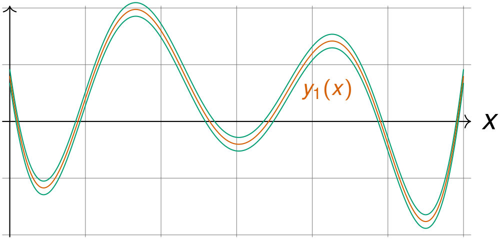
In fact, this is fundamental for Rubel's proof to work !

- Rubel's statement : this DAE is universal
- More realistic interpretation : this DAE allows almost anything

Open Problem (Rubel, 1981)

Is there a universal ODE $y' = p(y)$?
Note : explicit polynomial ODE $\Rightarrow$ unique solution

Universal initial value problem (IVP)



Theorem

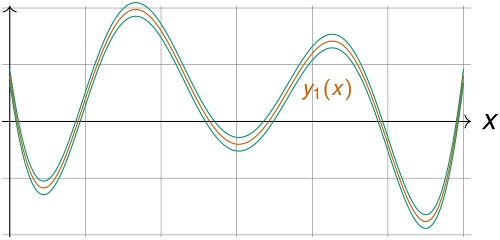
There exists a ***fixed*** *(vector of) polynomial p such that for any continuous functions f and* ε*, there exists* $\alpha \in \mathbb{R}^d$ *such that*

$$y(0) = \alpha, \qquad y'(t) = p(y(t))$$

has a ***unique solution*** $y : \mathbb{R} \to \mathbb{R}^d$ *and* $\forall t \in \mathbb{R}$,

$$|y_1(t) - f(t)| \leqslant \varepsilon(t).$$

Universal initial value problem (IVP)



Notes :

- **system** of ODEs,
- y is analytic,
- we need $d \approx 300$.

Theorem

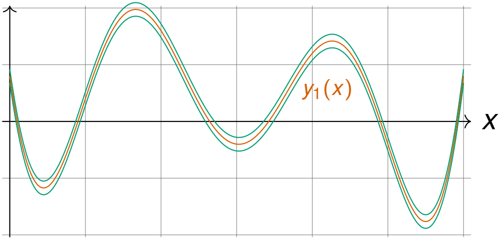
*There exists a **fixed** (vector of) polynomial p such that for any continuous functions f and* ε*, there exists* $\alpha \in \mathbb{R}^d$ *such that*

$$y(0) = \alpha, \qquad y'(t) = p(y(t))$$

*has a **unique solution** y* $: \mathbb{R} \to \mathbb{R}^d$ *and* $\forall t \in \mathbb{R}$,

$$|y_1(t) - f(t)| \leqslant \varepsilon(t).$$

Universal initial value problem (IVP)



Notes :

- **system** of ODEs,
- y is analytic,
- we need $d \approx 300$.

Theorem

*There exists a **fixed** (vector of) polynomial p such that for any continuous functions f and ε, there exists $\alpha \in \mathbb{R}^d$ such that*

$$y(0) = \alpha, \qquad y'(t) = p(y(t))$$

*has a **unique solution** $y : \mathbb{R} \to \mathbb{R}^d$ and $\forall t \in \mathbb{R}$,*

$$|y_1(t) - f(t)| \leqslant \varepsilon(t).$$

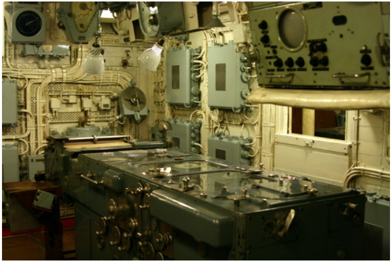
Remark : α is usually transcendental, but computable from f and ε

What is a computer?

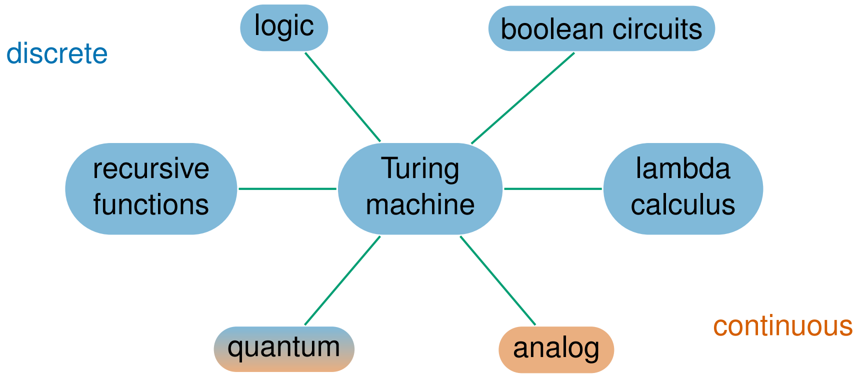
What is a computer ?

What is a computer ?

VS



Computability

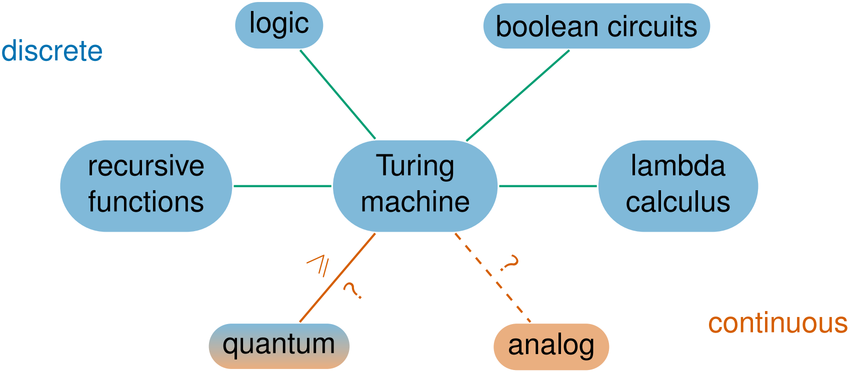


Church Thesis

All **reasonable** models of computation are equivalent.

Church Thesis

Complexity



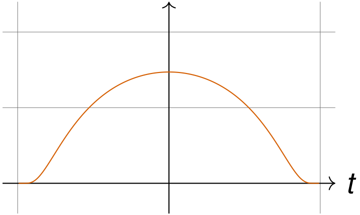
Effective Church Thesis

All **reasonable** models of computation are equivalent for complexity.

Rubel's proof in one slide

- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

It satisfies $\quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$



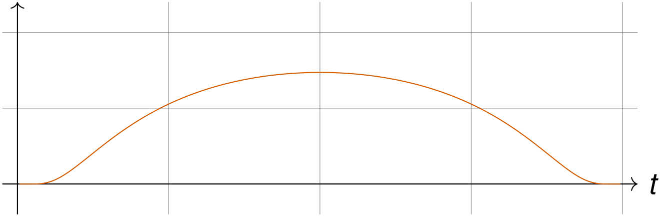
Rubel's proof in one slide

- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

  It satisfies $\quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$
- For any $a, b, c \in \mathbb{R}$, $y(t) = cf(at+b)$ satisfies

$$\begin{aligned} 3y'^4 y'' y''''^2 \quad & -4y'^4 y''^2 y'''' + 6y'^3 y''^2 y''' y'''' + 24 y'^2 y''^4 y'''' \\ & -12 y'^3 y'' y'''^3 - 29 y'^2 y''^3 y'''^2 + 12 y''^7 = 0 \end{aligned}$$

Translation and rescaling :

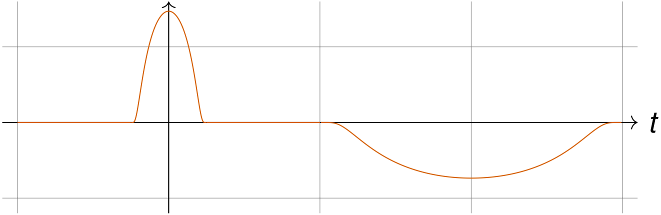


Rubel's proof in one slide

- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

  It satisfies $\quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$
- For any $a, b, c \in \mathbb{R}$, $y(t) = cf(at + b)$ satisfies

  $$3y'^4y''y''''^2 - 4y'^4y''^2y'''' + 6y'^3y''^2y'''y'''' + 24y'^2y''^4y'''' - 12y'^3y''y'''^3 - 29y'^2y''^3y'''^2 + 12y''^7 = 0$$
- Can glue together arbitrary many such pieces



Rubel's proof in one slide

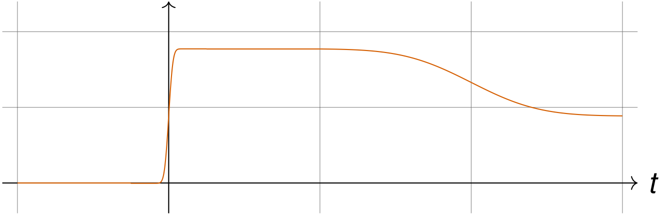
- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

  $$\text{It satisfies} \quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$$

- For any $a, b, c \in \mathbb{R}$, $y(t) = cf(at+b)$ satisfies

  $$3y'^4y''y''''^2 - 4y'^4y''^2y'''' + 6y'^3y''^2y'''y'''' + 24y'^2y''^4y'''' - 12y'^3y''y'''^3 - 29y'^2y''^3y'''^2 + 12y''^7 = 0$$

- Can glue together arbitrary many such pieces
- Can arrange so that $\int f$ is solution : piecewise pseudo-linear



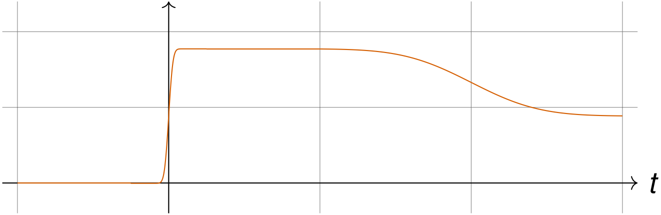
Rubel's proof in one slide

- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

  It satisfies $\quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$
- For any $a, b, c \in \mathbb{R}$, $y(t) = cf(at+b)$ satisfies

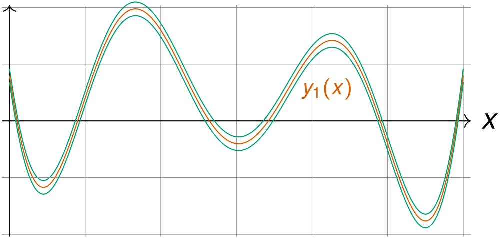
$$3y'^4y''y''''^2 - 4y'^4y''^2y'''' + 6y'^3y''^2y'''y'''' + 24y'^2y''^4y'''' - 12y'^3y''y'''^3 - 29y'^2y''^3y'''^2 + 12y''^7 = 0$$

- Can glue together arbitrary many such pieces
- Can arrange so that $\int f$ is solution : piecewise pseudo-linear



Conclusion : Rubel's equation allows any piecewise pseudo-linear functions, and those are **dense in** C^0

Universal DAE revisited



Theorem

*There exists a **fixed** polynomial p and* $k \in \mathbb{N}$ *such that for any continuous functions f and* ε*, there exists* $\alpha_0, \ldots, \alpha_k \in \mathbb{R}$ *such that*

$$p(y, y', \ldots, y^{(k)}) = 0, \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k$$

*has a **unique analytic solution** and this solution satisfies such that*

$$|y(t) - f(t)| \leqslant \varepsilon(t).$$