Theories of Computation for Continuous Systems. Computing with Analog Models.

Olivier Bournez

Ecole Polytechnique
Laboratoire d'Informatique de l'X
Palaiseau, France

Colloquium Morgenstern
15 March 2012

Menu

Sub-menu

Overall objective

Main objective

Understand computation theories for CONTINUOUS systems.

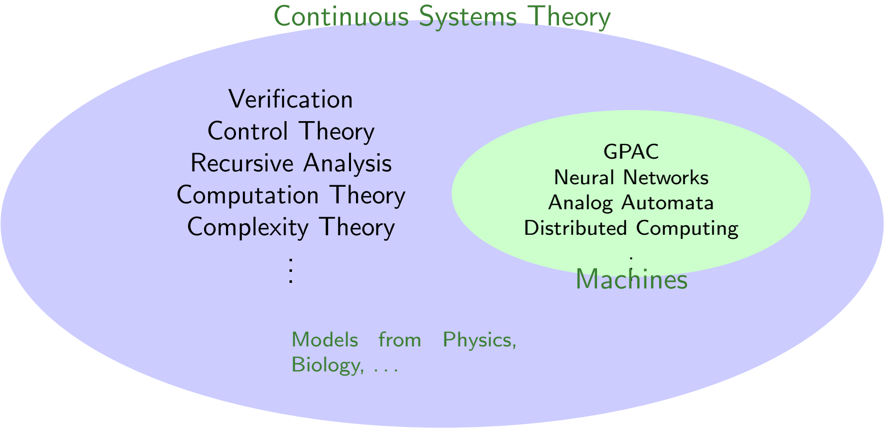
Dynamical Systems
over a continuous space
$\mathcal{H} = (\mathbb{R}^n, f)$

Discrete-Time

$$\left\{ \begin{array}{lcl} y(t+1) & = & f(y(t)) \\ y(0) & = & x \end{array} \right.$$

Continuous-Time

$$\left\{ \begin{array}{lcl} y' & = & f(y(t)) \\ y(0) & = & x \end{array} \right.$$


Continuous Systems Theory
Verification
Control Theory
Recursive Analysis
Computation Theory
Complexity Theory
GPAC
Neural Networks
Analog Automata
Distributed Computing
Machines
Models from Physics, Biology, ...

Continuous Systems Theory

Verification
Control Theory
Recursive Analysis
Computation Theory
Complexity Theory
⋮

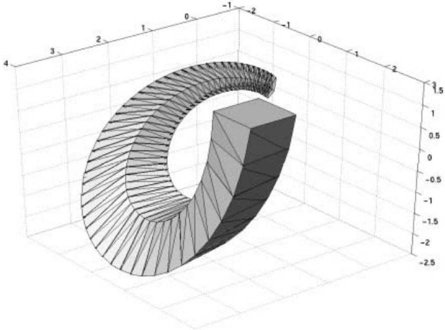
Models from Physics, Biology, ...

Sub-menu

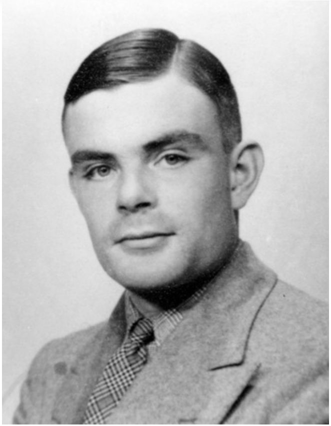
Main Focus

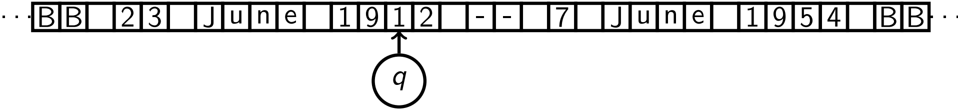
Verification and Control Theory

- **Reachability**. Given $\mathcal{H}$, x_0, $X \subset \mathbb{R}^n$, decide if there is a trajectory going from x_0 to X.
- **Stability**. Given $\mathcal{H}$, decide if all trajectories go to the origin.

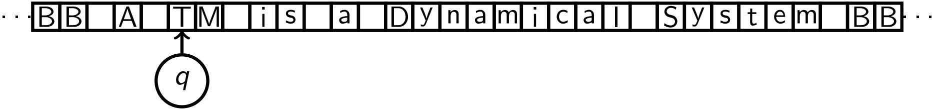


Alan M. Turing





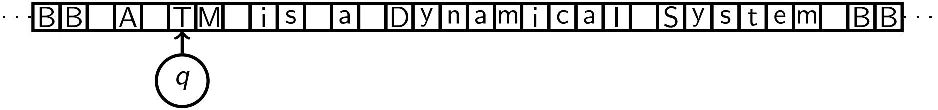
Preliminary: Digital World = Discrete Time and Space



A Turing machine is a particular discrete-time discrete-space dynamical systems.

- A Turing machine over alphabet Σ corresponds to a discrete time dynamical system

Preliminary: Digital World = Discrete Time and Space

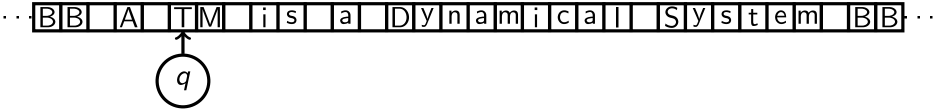


A Turing machine is a particular discrete-time discrete-space dynamical systems.

- A Turing machine over alphabet Σ corresponds to a discrete time dynamical system

$$(Q \times \mathbb{N} \times \Sigma^*, \vdash).$$

Preliminary: Digital World = Discrete Time and Space

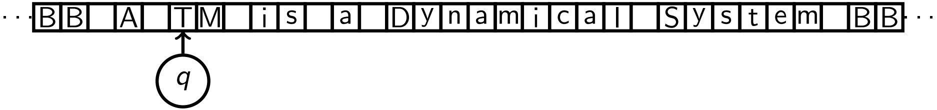


A Turing machine is a particular discrete-time discrete-space dynamical systems.

- A Turing machine over alphabet Σ corresponds to a discrete time dynamical system

$$(\mathbb{N}, \vdash).$$

Preliminary: Digital World = Discrete Time and Space



A Turing machine is a particular discrete-time discrete-space dynamical systems.

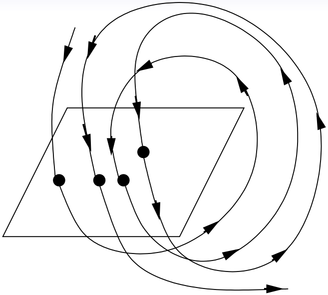
- A Turing machine over alphabet Σ corresponds to a discrete time dynamical system

$$(\mathbb{N}, \vdash).$$

- As $\mathbb{N} \subset \mathbb{R}$, it can be embedded into a continuous space dynamical system

$$(\mathbb{R}^m, f).$$

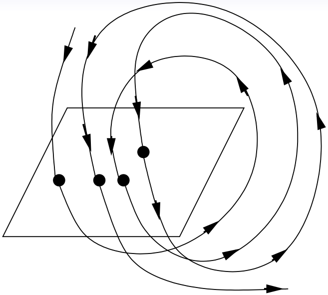
Dynamic Undecidability



Dynamic Undecidability Results:

- [Moore90]
- [Ruohonen93]
- [Siegelmann-Sontag94]
- [Asarin-Maler-Pnueli95]
- [Branicky95]
- [Graça-Campagnolo-Buescu2005]

Dynamic Undecidability

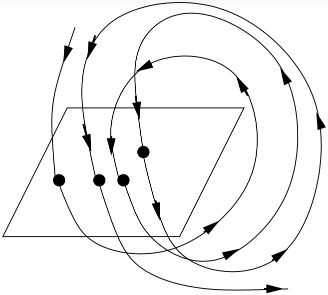


Dynamic Undecidability Results:

- [Moore90]
- [Ruohonen93]
- [Siegelmann-Sontag94]
- [Asarin-Maler-Pnueli95]
- [Branicky95]
- [Graça-Campagnolo-Buescu2005]

All non-trivial questions about dynamical systems are hard, from a computability and complexity point of view.

Dynamic Undecidability



Dynamic Undecidability Results:

- [Moore90]
- [Ruohonen93]
- [Siegelmann-Sontag94]
- [Asarin-Maler-Pnueli95]
- [Branicky95]
- [Graça-Campagnolo-Buescu2005]

All non-trivial questions about dynamical systems are hard, from a computability and complexity point of view.

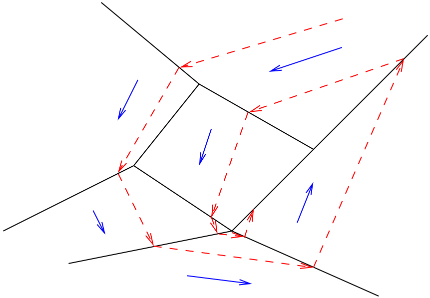
PROBLEMATIC !!

Amazing/EVEN MORE Problematics facts about continuous time systems

- Space contraction.
- Time contraction.
- Zeno's paradox phenomenon.

Piecewise Constant Derivative systems

PCD systems [Asarin-Maler-Pnueli94]:



$$dx/dt = f(x)$$

with $f : \mathbb{R}^d \to \mathbb{Q}^d$ piecewise constant:

1. $Range(f) = C$,

   $\#C < \infty$

2. for all $c \in C$, $f^{-1}(c)$ is a finite union of polyhedral convex subsets of $\mathbb{R}^d$.

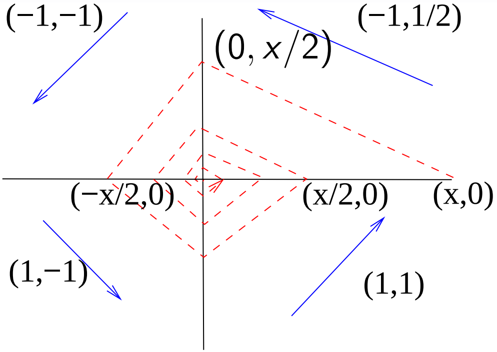
PCD systems in finite discrete time

Discrete time: number of regions crossed.

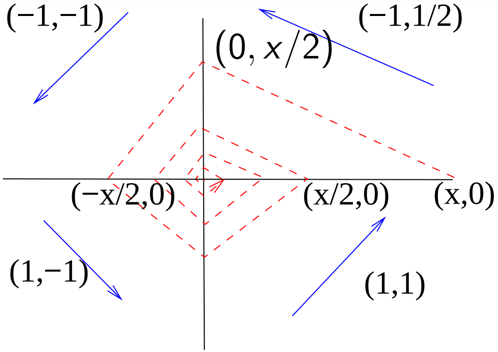
Theorem:

1. The reachability problem for PCD systems of dimension 2 is decidable [Asarin-Maler-Pnueli95].
2. PCD systems of dimension $d \geq 3$ can simulate Turing machines [Asarin-Maler-Pnueli95].

A trajectory of a PCD system

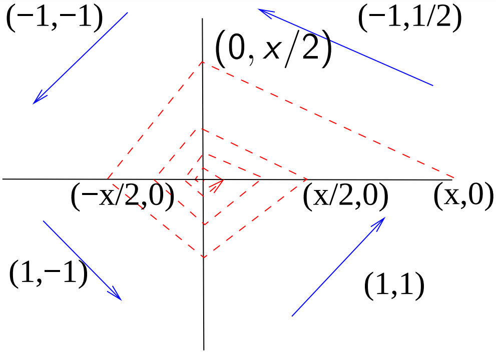


A trajectory of a PCD system



$$5/2(x + x/2 + x/4 + \dots) = 5$$

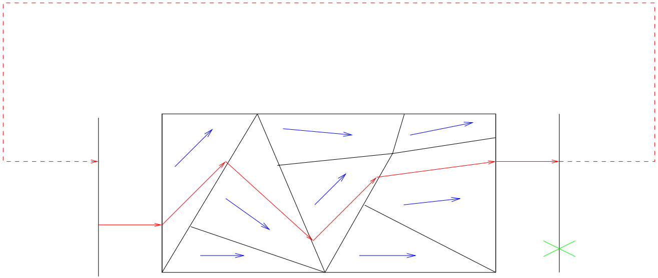
A trajectory of a PCD system



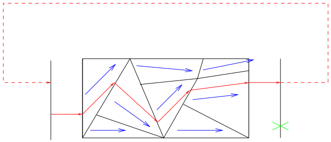
$$5/2(x + x/2 + x/4 + \dots) = 5$$

Observation [Zeno -490/-425]: to a finite continuous time can correspond a transfinite discrete time.

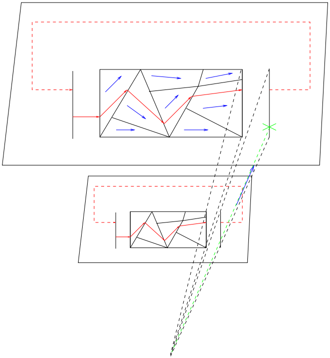
IDEA: Abstract 2-dimensional Representation of a 3-dimensional Turing Machine



IDEA: The Same But With Dimensions Divided by 2



IDEA: Recognizing the Halting Problem of a Turing Machine in dimension 4



In continuous time? [Bournez99]

Continuous time: time taken by the trajectory.

Dimension	Languages semi-recognized
2	$< \Sigma_1$
3	Σ_1
4	Σ_2
5	Σ_ω
6	$\Sigma_{\omega+1}$
7	Σ_{ω^2}
8	Σ_{ω^2+1}
...	...
2p+1	$\Sigma_{\omega^{p-1}}$
2p+2	$\Sigma_{\omega^{p-1}+1}$

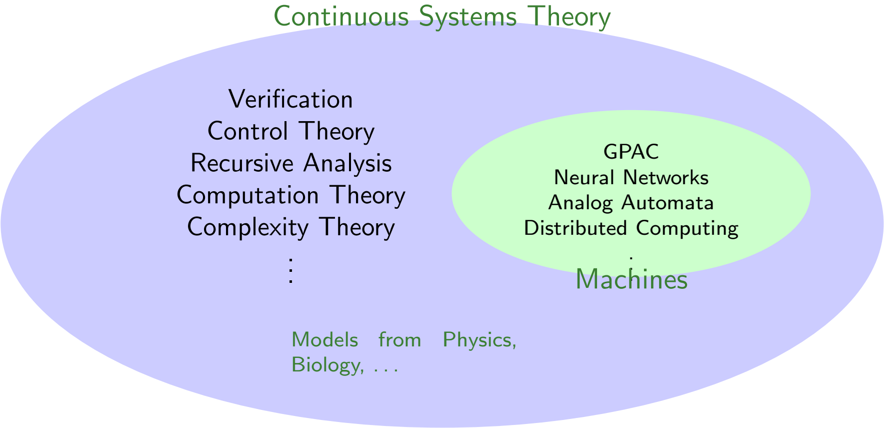
Extending [Asarin-Maler95].

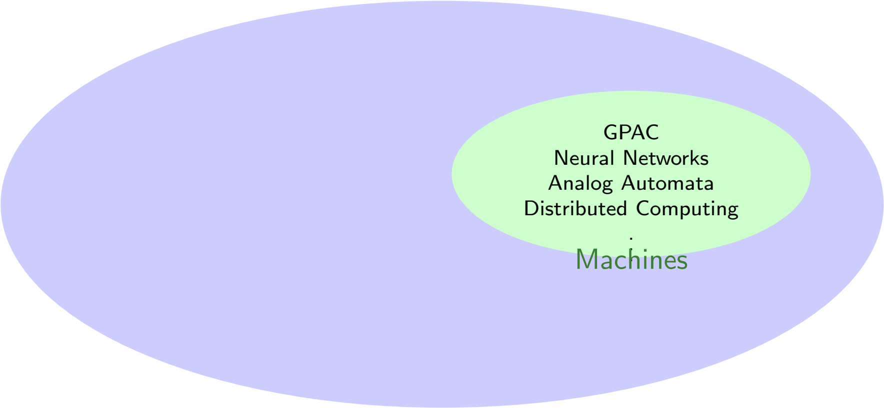
Smooth version

[Ruohonen97]: Space and time contractions can be used to prove that systems $(\mathbb{R}^m, f)$, with f smooth (i.e. $\mathcal{C}^\infty$), on a compact finite-dimensional domain, can simulate arbitrary Turing machines.

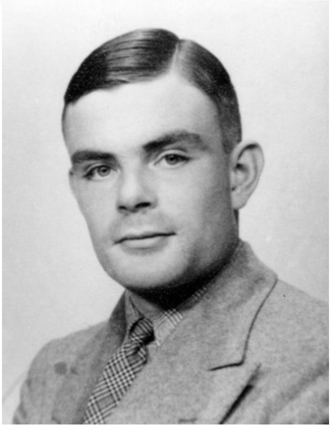
[Moore98] conjecture: No analytic function on a compact, finite-dimensional space, can simulate a Turing machine trough a reasonable input and output encoding.

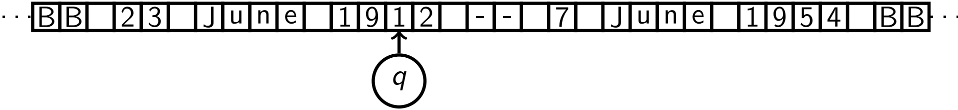
Sub-menu


Continuous Systems Theory
Verification
Control Theory
Recursive Analysis
Computation Theory
Complexity Theory
⋮
GPAC
Neural Networks
Analog Automata
Distributed Computing
⋮
Machines
Models from Physics, Biology, . . .


GPAC
Neural Networks
Analog Automata
Distributed Computing
Machines

Alan M. Turing





The digital world

- Many models of computations:
  - Recursive functions, Kurt Gödel, 1931-34.
  - Turing machines, Alan Turing, 1936.
  - λ-calculus, Alonzo Church, 1936.
  - Post systems
  - ...
- But, equivalent
  - at the computability level, through Church Turing's thesis
  - and also roughly equivalent at the complexity level: P, NP, ...

The digital world

- Many models of computations:
  - Recursive functions, Kurt Gödel, 1931-34.
  - Turing machines, Alan Turing, 1936.
  - λ-calculus, Alonzo Church, 1936.
  - Post systems
  - ...
- But, equivalent
  - at the computability level, through Church Turing's thesis
  - and also roughly equivalent at the complexity level: P, NP, ...
- These are digital models: time is discrete, space is discrete.

What about analog models?

The (digital) Picture

Church Thesis	“What is effectively calculable is computable”
Thesis M	“What can be calculated by a machine is computable”
Thesis?	“What can be calculated by a model is computable”

(following [Copeland2002])

Understanding computational power of models helps to understand

- limits of mechanical reasoning.
- limits of machines.
- limits of models.

Menu

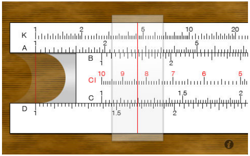
Sub-menu

Some Analog Computers

- Antikythera Mechanism

(-87, authors?)

- Slide Rule



(1620 - 1630, Napier, Gunter, Wingate)

- Planimeter

(1814, Hermann)

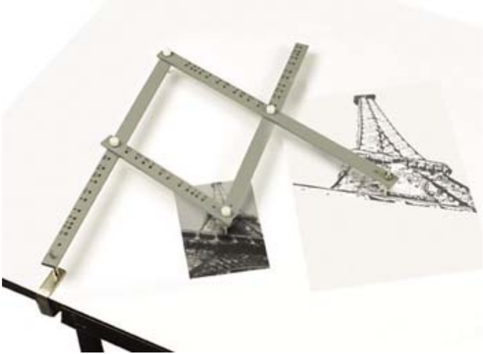
- MONIAC/Financephalograph

(1949, Phillips)

Sub-menu

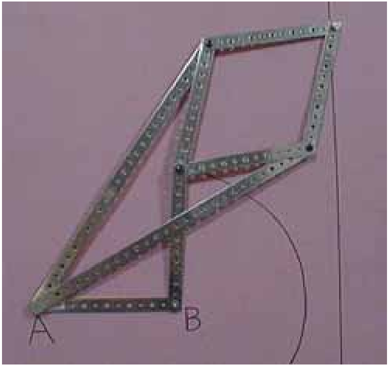
A model from 19th Century: Rivets' mechanisms

- Rivets' mechanisms.
  - How to realize an homothety: the pantograph.



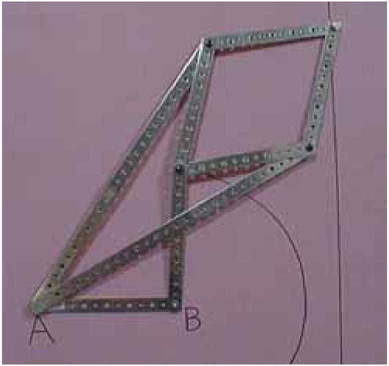
A model from 19th Century: Rivets' mechanisms

- How to transform a circular into a linear motion: Peaucellier's mechanism (1864 - 1871).



A model from 19th Century: Rivets' mechanisms

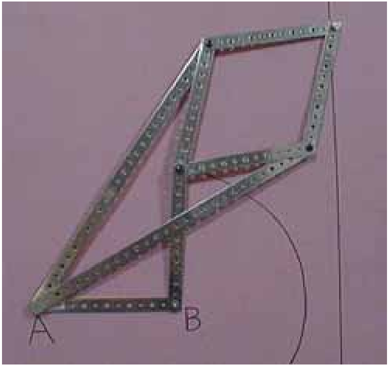
- How to transform a circular into a linear motion: Peaucellier's mechanism (1864 - 1871).



- Computational power of Rivets's mechanisms?

A model from 19th Century: Rivets' mechanisms

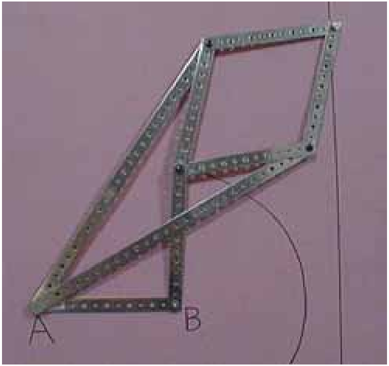
- How to transform a circular into a linear motion: Peaucellier's mechanism (1864 - 1871).



- Computational power of Rivets's mechanisms?
  - Theorem [Kempke]: computable iff semi-algebraic.

A model from 19th Century: Rivets' mechanisms

- How to transform a circular into a linear motion: Peaucellier's mechanism (1864 - 1871).



- Computational power of Rivets's mechanisms?
  - Theorem [Kempke]: computable iff semi-algebraic.

Voir aussi: "De la nécessité de tracer les droites au compas", Pierre Damphousse, Fête de la Science.

Formally

Theorem (Computational power of planar mechanisms)

- *For any non-empty semi-algebraic set S, there exists a mechanism with n points that move on linear segments, but that are free to move on these segments, and that forces the relation $(x_1, \ldots, x_n) \in S$, where x_i are the distances on the linear segments.*
- *Conversely, the domain of evolution of any finite planar mechanism is semi-algebraic.*

(theorem attributed to Kempke).

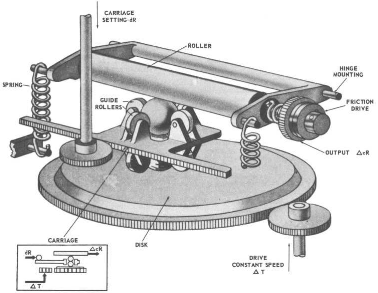
Sub-menu

A machine from 20th Century: Differential analyzers

Vannevar Bush's 1938 mechanical Differential Analyser

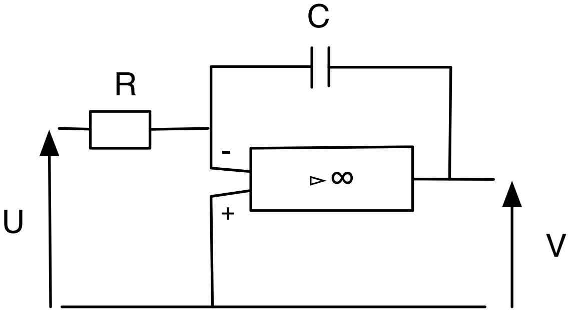
- Underlying principles: Lord Kelvin 1876.
- First ever built: V. Bush 1931 at MIT.
- Applications: from gunfire control up to aircraft design
- Intensively used during U.S. war effort.
- Electronic versions from late 40s, used until 70s

A Mechanical Integrator



Bureau of Naval Personnel, *Basic Machines and How They Work*, 1964

A Modern Electronic Integrator



$$V(t) = -1/RC \int_0^t U(t)dt$$

Electronic Differential Analyzer

Advertisements in Scientific American, March 1953.

Electronic Differential Analyzer

Advertisements in Scientific American, March 1953.

See also:

Doug Coward's
Analog Computer Museum

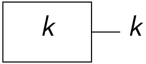
http://dcoward.best.vwh.net/analog/

The General Purpose Analog Computer

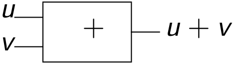
The GPAC

A mathematical abstraction from Claude Shannon (1941) of the Differential Analyzers.

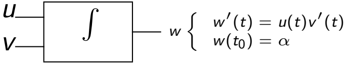
- Basic units:



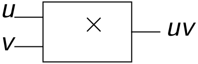
A constant unit



An adder unit

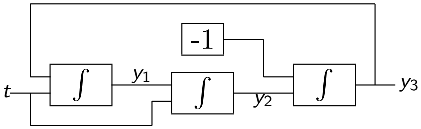


An integrator unit



A multiplier unit

Example: Generating cos and sin via a GPAC



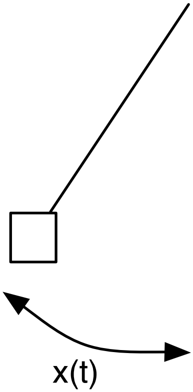
$$\begin{cases} y_1' = y_3 & \& \quad y_1(0) = 1 \\ y_2' = y_1 & \& \quad y_2(0) = 0 \\ y_3' = -y_2' & \& \quad y_3(0) = 0 \end{cases}$$

$$y_1 = \cos(t),\ y_2 = \sin(t),\ y_3 = -\sin(t).$$

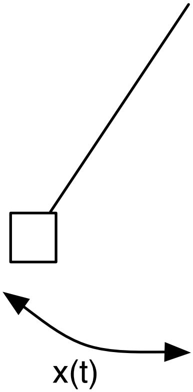
Programming with GPACs: Example. Pendulum

Suppose you want to solve

$$x'' + p^2 \sin(x) = 0.$$



Programming with GPACs: Example. Pendulum



Suppose you want to solve

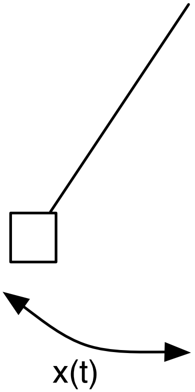
$$x'' + p^2 \sin(x) = 0.$$

Program:

Let's define

$$\begin{cases} y = x' \\ z = \sin(x) \\ u = \cos(x) \end{cases}$$

Programming with GPACs: Example. Pendulum



Suppose you want to solve

$$x'' + p^2 \sin(x) = 0.$$

Program:
Let's define

$$\begin{cases} y &= x' \\ z &= \sin(x) \\ u &= \cos(x) \end{cases}$$

To get

$$\begin{cases} x' &= y \\ y' &= -p^2 z \\ z' &= yu \\ u' &= -yz \end{cases}.$$

Sub-menu

From discrete to continuous models

$$
\begin{array}{ll}
Q = \{q_1, q_2, q_3, q_4\} & \\
\delta(q_1, q_2) = q_2 & [\beta] \\
\delta(q_2, q_1) = q_2 & [\beta] \\
\delta(q_4, q_2) = q_3 : 1/2, q_4 : 1/2 & [\nu] \\
\delta(q_2, q_4) = q_3 : 1/2, q_4 : 1/2 & [\nu]
\end{array}
$$

From discrete to continuous models

Microscopic Dynamic

$$Q = \{q_1, q_2, q_3, q_4\}$$

$$\begin{array}{ll} \delta(q_1, q_2) = q_2 & [\beta] \\ \delta(q_2, q_1) = q_2 & [\beta] \\ \delta(q_4, q_2) = q_3 : 1/2, q_4 : 1/2 & [\nu] \\ \delta(q_2, q_4) = q_3 : 1/2, q_4 : 1/2 & [\nu] \end{array}$$

Macroscopic Dynamic

Kermack-McKendrick SIR model.

$$\left\{ \begin{array}{ccc} S' & = & -\beta SI \\ I' & = & \beta SI - \nu I \\ R' & = & \nu I. \end{array} \right.$$

From discrete to continuous models

Microscopic Dynamic

$$
\begin{array}{ll}
\multicolumn{2}{c}{Q = \{q_1, q_2, q_3, q_4\}} \\
\delta(q_1, q_2) = q_2 & [\beta] \\
\delta(q_2, q_1) = q_2 & [\beta] \\
\delta(q_4, q_2) = q_3 : 1/2, q_4 : 1/2 & [\nu] \\
\delta(q_2, q_4) = q_3 : 1/2, q_4 : 1/2 & [\nu]
\end{array}
$$

Macroscopic Dynamic

Kermack-McKendrick
SIR model.

$$
\left\{
\begin{array}{lcc}
S' & = & -\beta S I \\
I' & = & \beta S I - \nu I \\
R' & = & \nu I.
\end{array}
\right.
$$

Epidemic Rate

$$R = \beta S_0 / \nu.$$

From discrete to continuous models

Microscopic Dynamic

$$Q = \{q_1, q_2, q_3, q_4\}$$

$$\begin{array}{ll} \delta(q_1, q_2) = q_2 & [\beta] \\ \delta(q_2, q_1) = q_2 & [\beta] \\ \delta(q_4, q_2) = q_3 : 1/2, q_4 : 1/2 & [\nu] \\ \delta(q_2, q_4) = q_3 : 1/2, q_4 : 1/2 & [\nu] \end{array}$$

Macroscopic Dynamic

Kermack-McKendrick
SIR model.

$$\left\{ \begin{array}{ccc} S' & = & -\beta SI \\ I' & = & \beta SI - \nu I \\ R' & = & \nu I. \end{array} \right.$$

Epidemic Rate

$$R = \beta S_0 / \nu.$$

Question

Programming/Computing with Such models?

A model from 21th century: Computing with Large Populations

- **My favourite example:**
  - States: $\{{\color{red}\bullet},{\color{blue}\bullet}\}$
  - Rules of interactions:

$$
\begin{aligned}
{\color{red}\bullet}\ {\color{red}\bullet} &\rightarrow {\color{red}\bullet}\ {\color{blue}\bullet}\\
{\color{red}\bullet}\ {\color{blue}\bullet} &\rightarrow {\color{red}\bullet}\ {\color{red}\bullet}\\
{\color{blue}\bullet}\ {\color{red}\bullet} &\rightarrow {\color{red}\bullet}\ {\color{red}\bullet}\\
{\color{blue}\bullet}\ {\color{blue}\bullet} &\rightarrow {\color{red}\bullet}\ {\color{blue}\bullet}
\end{aligned}
$$

  - What can we say about

$$p_{\color{red}\bullet} = \frac{number\ of\ {\color{red}\bullet}}{number\ of\ {\color{red}\bullet} + number\ of\ {\color{blue}\bullet}}?$$

- **This is a model**
  - inspired from [Angluin,Aspnes,Diamadi,Fischer,Peralta 2004]'s Population Protocoles introduced in the context of distributed systems / (anonymous) sensor networks.
  - but with a large population hypothesis.

Informal approach on this example

$$\textcolor{red}{\bullet}\ \textcolor{red}{\bullet} \rightarrow \textcolor{red}{\bullet}\ \textcolor{blue}{\bullet}$$
$$\textcolor{red}{\bullet}\ \textcolor{blue}{\bullet} \rightarrow \textcolor{red}{\bullet}\ \textcolor{red}{\bullet}$$
$$\textcolor{blue}{\bullet}\ \textcolor{red}{\bullet} \rightarrow \textcolor{red}{\bullet}\ \textcolor{red}{\bullet}$$
$$\textcolor{blue}{\bullet}\ \textcolor{blue}{\bullet} \rightarrow \textcolor{red}{\bullet}\ \textcolor{blue}{\bullet}$$

- The mean number of $\textcolor{red}{\bullet}$ created,

$$\begin{aligned} b(p_{\textcolor{red}{\bullet}}) &= -1 * p_{\textcolor{red}{\bullet}}^2 + 1 * p_{\textcolor{red}{\bullet}}(1 - p_{\textcolor{red}{\bullet}}) + 1 * p_{\textcolor{red}{\bullet}}(1 - p_{\textcolor{red}{\bullet}}) + 1 * (1 - p_{\textcolor{red}{\bullet}})^2 \\ &= 1 - 2p_{\textcolor{red}{\bullet}}^2 \end{aligned}$$

  must be equal, at the limit to 0,
  and hence

$$p_{\textcolor{red}{\bullet}} = \frac{\sqrt{2}}{2},$$

  at the limit.

- In other words,

  this protocol **computes** real number $\frac{\sqrt{2}}{2}$.

Main result

Theorem

ν is computable by an LPP

if and only if

$\nu \in [0, 1]$ is algebraic.

Menu

What can be generated by a GPAC?

- The purpose of Shannon's 41 paper is a characterization of GPAC generable functions.
- Shannon's 41 characterization is incomplete: Several problems, even about definitions, corrected by [PourEl-Richards74], [Lipshitz-Rubel87], [Graça-Costa03].
- For the better defined class considered in [Graça-Costa03].

Proposition (Graça-Costa03)

A scalar function $f : \mathbb{R} \to \mathbb{R}$ is generated by a GPAC iff it is a component of polynomial continuous time dynamical system.

What can be generated by a GPAC?

- The purpose of Shannon's 41 paper is a characterization of GPAC generable functions.
- Shannon's 41 characterization is incomplete: Several problems, even about definitions, corrected by [PourEl-Richards74], [Lipshitz-Rubel87], [Graça-Costa03].
- For the better defined class considered in [Graça-Costa03].

Proposition (Graça-Costa03)

A scalar function $f : \mathbb{R} \to \mathbb{R}$ is generated by a GPAC iff it is a component of polynomial continuous time dynamical system.

- These functions will be also called *pIVP* functions.

Formally:

- For the better defined class considered in [Graça-Costa03], a scalar function $f : \mathbb{R} \to \mathbb{R}$ is generated by a GPAC iff

$$f(t) = y_i(t)$$

for $y(t) \in \mathbb{R}^m$ solution of

$$\begin{cases} y' & = & p(t, y), \\ y(0) & = & x \end{cases} \tag{1}$$

where p is (a vector of) polynomials.

Formally:

- For the better defined class considered in [Graça-Costa03], a scalar function $f : \mathbb{R} \rightarrow \mathbb{R}$ is generated by a GPAC iff

$$f(t) = y_i(t)$$

for $y(t) \in \mathbb{R}^m$ solution of

$$\left\{ \begin{array}{lcl} y' & = & p(t, y), \\ y(0) & = & x \end{array} \right. \tag{1}$$

where p is (a vector of) polynomials.

- These functions will be also called *pIVP* functions.

Uncomputability (Ungenerability) Results

Consequence: A GPAC generated unary function $f : I \subset \mathbb{R} \to \mathbb{R}$ must be differentially algebraic (d.a.):

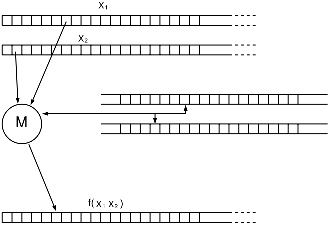
i.e. it satisfies some algebraic differential equation of the form $p\left(t, y, y', \ldots, y^{(n)}\right) = 0$, where p is a non-zero polynomial in all its variables.

Non-d.a. functions:

- Gamma function $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ [Hölder 1887].
- Riemann's Zeta function $\zeta(x) = \sum_{k=0}^\infty \frac{1}{k^x}$ [Hilbert].

Menu

Due to Turing, Grzegorczyk, Lacombe. Here presentation from Weihrauch.



A tape represents a real number

Each real number x is represented via an infinite sequence $(x_n)_n \in \mathbb{Q}$,

$$||x_n - x|| \leq 2^{-n}.$$

M behaves like a Turing Machine

Read-only one-way input tapes
Write-only one-way output tape.
M outputs a representation of $f(x_1, x_2)$ from representations of x_1, x_2.

Solving ODEs and computability

- Pour-El Richards 79:
  - There exists some computable $f : [0,1] \times [-1,1] \to \mathbb{R}$ such that ordinary differential equation

    $$y' = f(t, y),$$

    has no computable solution over any closed domain.

- Graça Zhong Buescu 2007:
  - If $f : [0,1] \times [-1,1] \to \mathbb{R}$ is computable and ordinary differential equation

    $$y' = f(t, y),$$

    has a **unique** solution, then it must be computable.

Moral on Analog Models of Computations

- **Summary:**

$$\text{GPAC generable} \subsetneq \text{Computable}$$

- **With more details:**

  - [Graça Zhong Buescu 2007] Let $f : (\alpha, \beta) \subset \mathbb{R} \to \mathbb{R}^k$ be some pIVP function with computable parameters.
    Then f is computable on (α, β).
  - pIVP functions must be analytic.
  - Computable functions include some non-analytic functions (ex: $\min(x, 0)$).
  - Gamma function and Riemann's Zeta function are computable.

Criticisms

We stated

$$\text{GPAC generable} \subsetneq \text{Computable}.$$

- However, the notion of GPAC generated function assumes computation in "real time" - a very restrictive form of computation.

Criticisms

We stated

$$\text{GPAC generable} \subsetneq \text{Computable.}$$

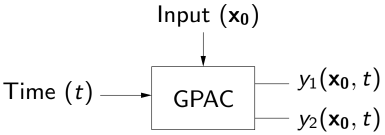
- However, the notion of GPAC generated function assumes computation in "real time" - a very restrictive form of computation.
- What happen if we change this notion of computability to the kind of "converging computation" used in recursive analysis,

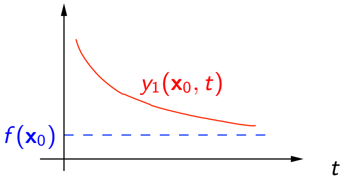
GPAC Computability vs GPAC Generation

Definition

A function $f : [a, b] \to \mathbb{R}$ is GPAC-computable iff there exist some computable polynomials $p : \mathbb{R}^{n+1} \to \mathbb{R}^n$, $p_0 : \mathbb{R} \to \mathbb{R}$, and $n-1$ computable real values $\alpha_1, ..., \alpha_{n-1}$ such that:

1. $(y_1, ..., y_n)$ is the solution of the Cauchy problem $y' = p(y, t)$ with initial condition $(\alpha_1, ..., \alpha_{n-1}, p_0(x))$ set at time $t_0 = 0$
2. $\lim_{t\to\infty} y_2(t) = 0$
3. $|f(x) - y_1(t)| \leq y_2(t)$ for all $x \in [a, b]$ and all $t \in [0, +\infty)$.





Graça 04's Result

Proposition (Graça 04)

The Gamma function Γ *is GPAC-computable.*

(so is the ζ function)

Bournez, Campagnolo, Graça, Hainry's result

Theorem

Let a and b be computable reals. A function $f : [a, b] \to \mathbb{R}$ is computable iff it is GPAC-computable.

In a provocative way:

- GPAC is not weaker than modern machines, from a computability point of view.

Menu

An Other Morality & A BIG question?

- $TIME_{TM}(t) \subseteq TIME_{GPAC}(t)$.
- **Important question:**
  - Formulation 1: Can GPAC compute faster than Turing machines?
  - Formulation 2: $TIME_{GPAC}(t) \subseteq TIME_{TM}(t)$?

An Other Morality & A BIG question?

- $TIME_{TM}(t) \subseteq TIME_{GPAC}(t)$.
- **Important question:**
  - Formulation 1: Can GPAC compute faster than Turing machines?
  - Formulation 2: $TIME_{GPAC}(t) \subseteq TIME_{TM}(t)$?
  - Formulation 3: Can (at least polynomial) ordinary differential equations be solved in polynomial time?

Problems and solutions

Usual methods problems:

- Finite order method

Problems and solutions

Usual methods problems:

- Finite order method
  $\Rightarrow$ Not polynomial

Problems and solutions

Usual methods problems:

- Finite order method
  $\Rightarrow$ Not polynomial
- Assume compact domain or Lipschitz constant:
  $||p(a) - p(b)|| \leqslant L||a - b||$

Problems and solutions

Usual methods problems:

- Finite order method
  $\Rightarrow$ Not polynomial
- Assume compact domain or Lipschitz constant:
  $||p(a) - p(b)|| \leqslant L||a - b||$
  $\Rightarrow$ Useless algorithms for our theoretical analysis

Problems and solutions

Usual methods problems:

- Finite order method
  $\Rightarrow$ Not polynomial
- Assume compact domain or Lipschitz constant:
  $||p(a) - p(b)|| \leqslant L||a - b||$
  $\Rightarrow$ Useless algorithms for our theoretical analysis

Solutions

- Unbounded order method

Problems and solutions

Usual methods problems:

- Finite order method
  $\Rightarrow$ Not polynomial
- Assume compact domain or Lipschitz constant:
  $||p(a) - p(b)|| \leqslant L||a - b||$
  $\Rightarrow$ Useless algorithms for our theoretical analysis

Solutions

- Unbounded order method
- No assumptions on the domain

Problems and solutions

Usual methods problems:

- Finite order method
  $\Rightarrow$ Not polynomial
- Assume compact domain or Lipschitz constant:
  $||p(a) - p(b)|| \leqslant L||a - b||$
  $\Rightarrow$ Useless algorithms for our theoretical analysis

Solutions

- Unbounded order method
- No assumptions on the domain
- Do not assume Lipschitz functions

Problems and solutions

Usual methods problems:

- Finite order method
  $\Rightarrow$ Not polynomial
- Assume compact domain or Lipschitz constant:
  $||p(a) - p(b)|| \leqslant L||a - b||$
  $\Rightarrow$ Useless algorithms for our theoretical analysis

Solutions

- Unbounded order method
- No assumptions on the domain
- Do not assume Lipschitz functions

$\Rightarrow$ New problems !

A solution

We want to solve:

$$\begin{cases} y' &= p(y) \\ y(t_0) &= y_0 \end{cases}$$

A result (submitted):

- A simple algorithm: variable order multi-step Taylor method
- Tricky proof: error analysis and parameter choices

Main benefits: The proposed method is indeed polynomial !!

The algorithm

Algorithm 1: SolvePIVP

input : The initial condition $(t_0, y_0) \in \mathbb{Q} \times \mathbb{Q}^d$
input : The polynomial p of the PIVP
input : The total time step $T \in \mathbb{Q}$
input : The precision ξ requested
input : The number of steps N
input : The order of the method ω
output: $x \in \mathbb{Q}^d$

begin
  $\Delta \leftarrow \frac{T}{N}$
  $x \leftarrow y_0$
  for $n \leftarrow 1$ **to** N **do**
    $x \leftarrow \sum_{i=0}^{\omega-1} \frac{\Delta^i}{i!}\texttt{NthDeriv}(p, t_0 + n\Delta, x, \omega, \xi + \Delta)$

Main Result: Technical View

Theorem

Let $k = \deg(p)$, $\mu \geqslant 2$, $T \in \mathbb{Q}_+$, $Y \in \mathbb{Q}$ *such that*

$$Y \geqslant \sup_{t_0 \leqslant u \leqslant t_0 + T} \|y(u)\|_\infty$$

Then Previous Algorithm guarantees

$$\|y(t_0 + T) - \mathrm{SolvePIVP}(t_0, \tilde{y}_0, p, T, \omega, N, \omega)\|_\infty \leqslant e^{-\mu}$$

with the following parameters

$$\Delta = \frac{T}{N} \quad M = (2 + Y)^k \quad A = d(1 + k!\Sigma p M) \quad N = \lceil TeA \rceil$$

$$B = k4^k \Sigma p \Delta M \quad \omega = 2+\mu+\ln(N)+NB \quad \|y_0 - \tilde{y}_0\|_\infty \leqslant e^{-NB-\mu-1}$$

Morality on Analog Computations

- **Morality:** $TIME_{GPAC}(t) \subseteq TIME_{TM}(t)$ for pIVP functions that stay bounded or polynomially bounded.

$$Y \geqslant \sup_{t_0 \leqslant u \leqslant t_0+T} \|y(u)\|_\infty$$

- **Next question:** Is $TIME_{TM}(t) \subseteq TIME_{GPAC}(t)$ true for such functions.

- I.e. Derive a class of ODEs such that

$$Poly - Time_{TM} = Poly - Time_{ODE}.$$

Menu

Conclusions

- We saw various analog models of computations.
- No possible unification of all analog models:
  - Computability: Several models are provably different.
  - Complexity:
    - Even defining the time of a computation is problematic for some of them.

For the GPAC model

- **Important notice:** the GPAC is the only "general purpose" "physically motivated" model that we presented.

- **Analog Computability / Complexity:**
  - GPAC generable $\subsetneq$ Computable.
  - Computability: GPAC computable $=$ Computable.
  - Close to a notion of complexity for GPAC.

- **Promising perspective:**
  - Towards a complexity theory for analog models of computations.

The (digital) Picture

Church Thesis	"What is effectively calculable is computable"
Thesis M	"What can be calculated by a machine is computable"
Thesis?	"What can be calculated by a model is computable"

(following [Copeland2002])

Understanding computational power of models helps to understand

- limits of mechanical reasoning.
- limits of machines.
- limits of models.