A Survey on Analog Models of Computation

Amaury Pouly
Joint work with Olivier Bournez

Université de Paris, IRIF, CNRS, F-75013 Paris, France

30 june 2020

Survey: `https://arxiv.org/abs/1805.05729`

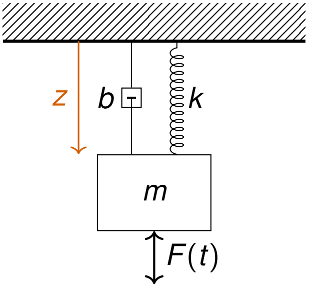
INSTITUT DE RECHERCHE EN INFORMATIQUE FONDAMENTALE

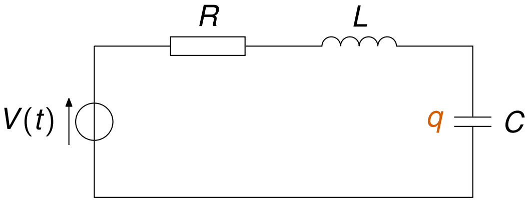
université PARIS DIDEROT

Université de Paris

The meaning of “analog”

Historically: “analog” = by analogy, *i.e.* same evolution

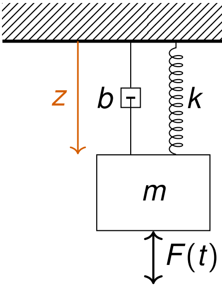


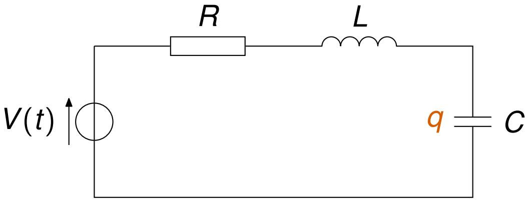


$$F = m\ddot{z} + b\dot{z} + kz \iff V = L\ddot{q} + R\dot{q} + \frac{1}{C}q$$

The meaning of “analog”

Historically: “analog” = by analogy, *i.e.* same evolution



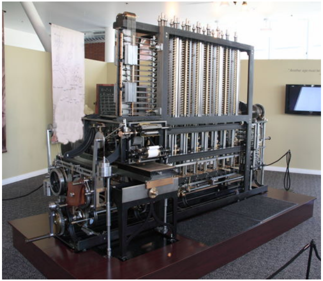


$$F = m\ddot{z} + b\dot{z} + kz \Leftrightarrow V = L\ddot{q} + R\dot{q} + \frac{1}{C}q$$

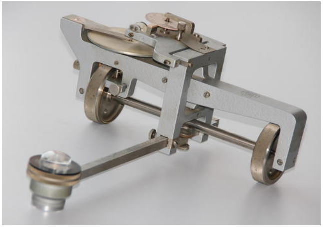
Nowadays: “analog” = continuous/opposite of digital

- $\Rightarrow$ orthogonal concepts
- $\Rightarrow$ even continuous/discrete unclear: hybrid exists

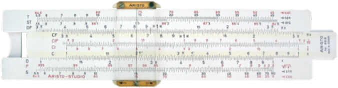
Some analog machines



Difference Engine



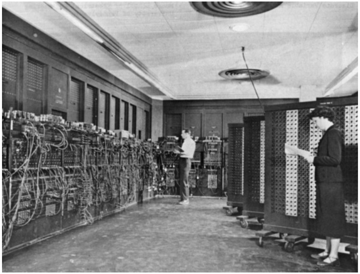
Linear Planimeter



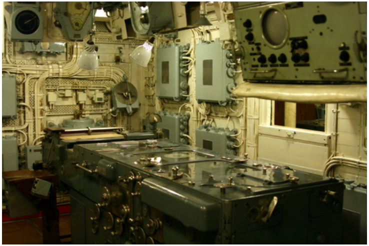
Slide Rule

Antikythera mechanism

Some analog machines



ENIAC

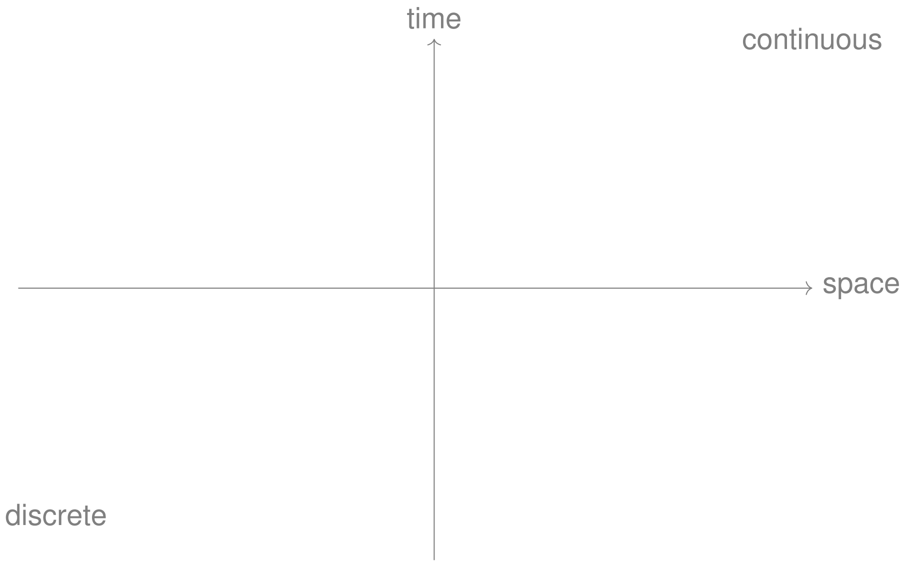


Admiralty Fire Control Table

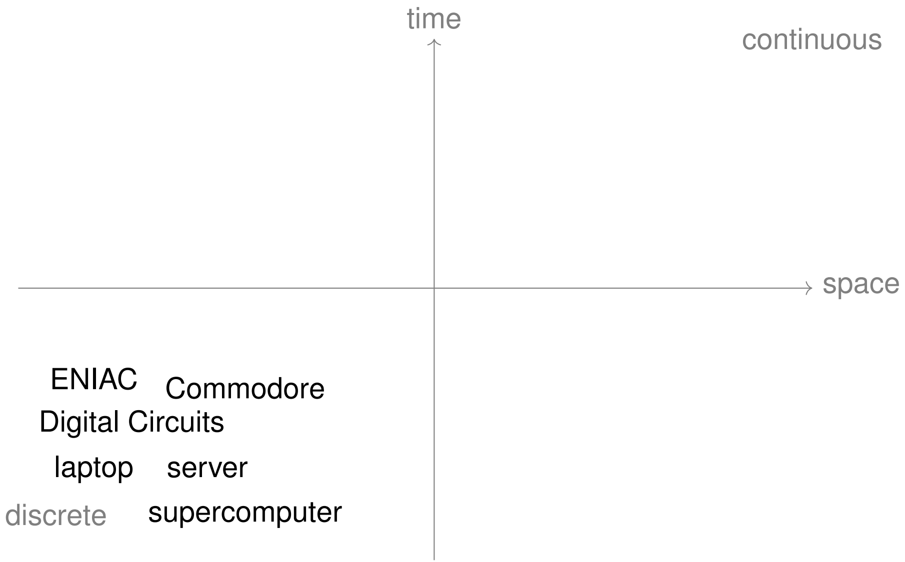
Differential Analyzer

Kelvin's Tide Predicter

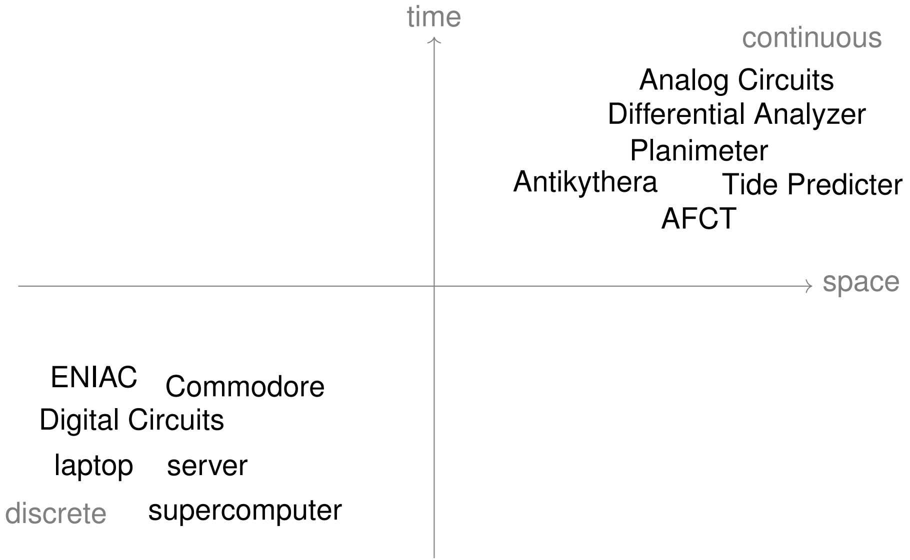
Classifying machines/models



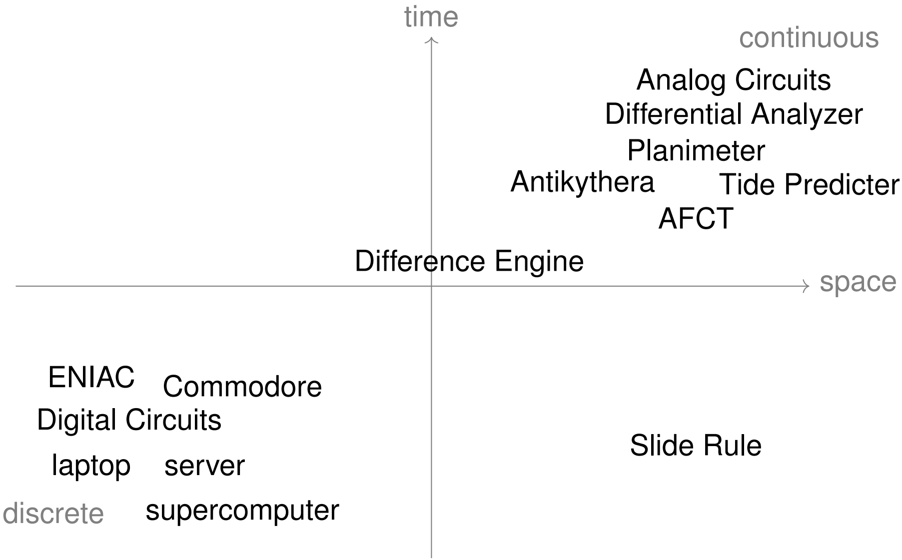
Classifying machines/models



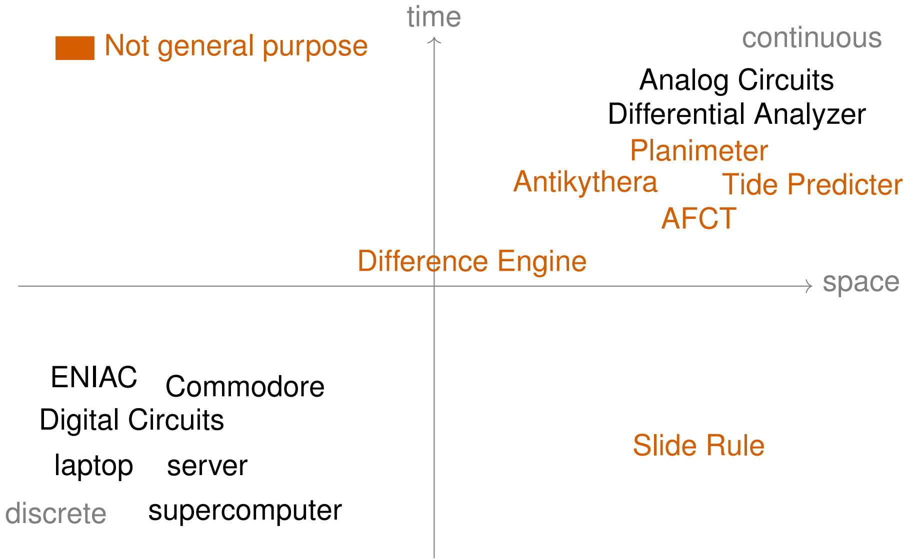
Classifying machines/models



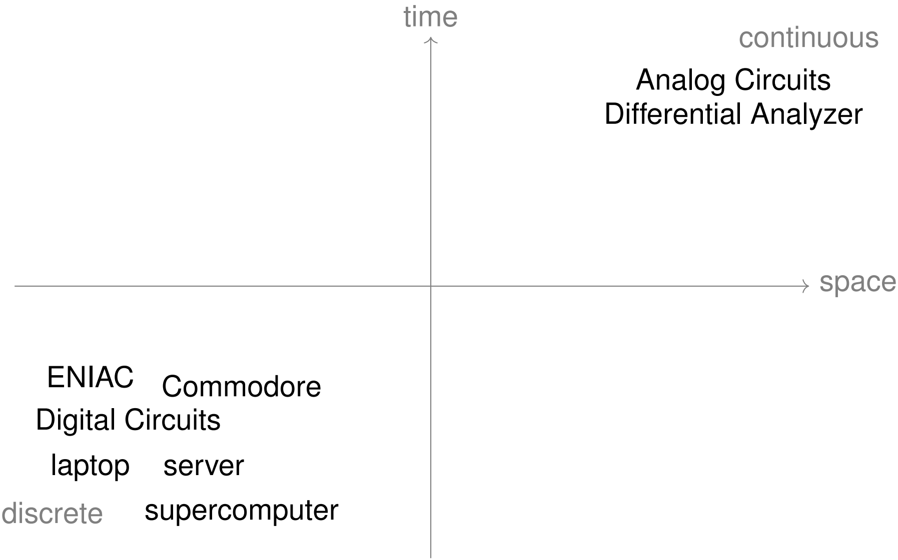
Classifying machines/models



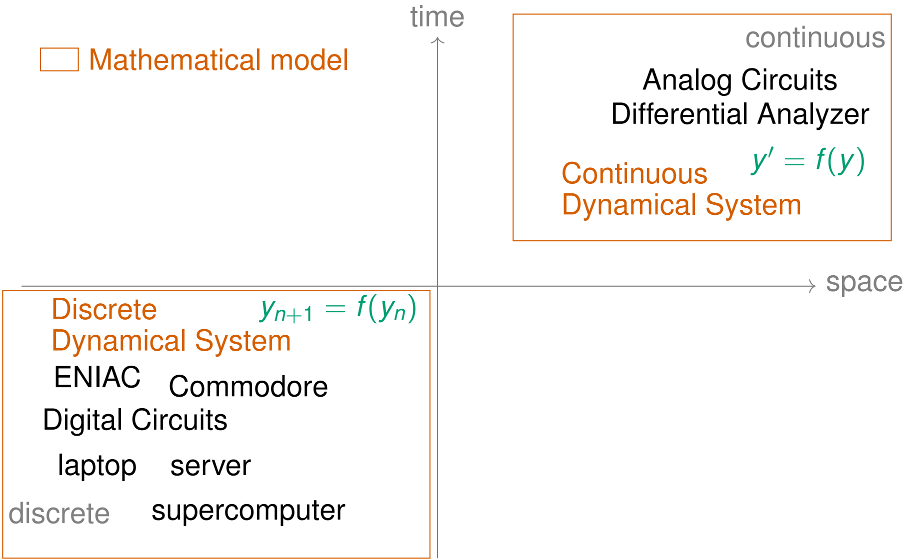
Classifying machines/models



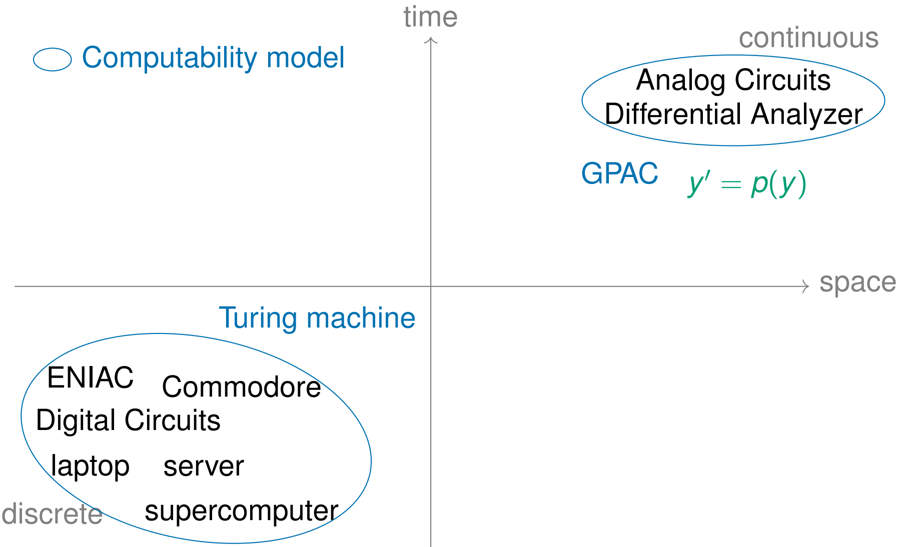
Classifying machines/models



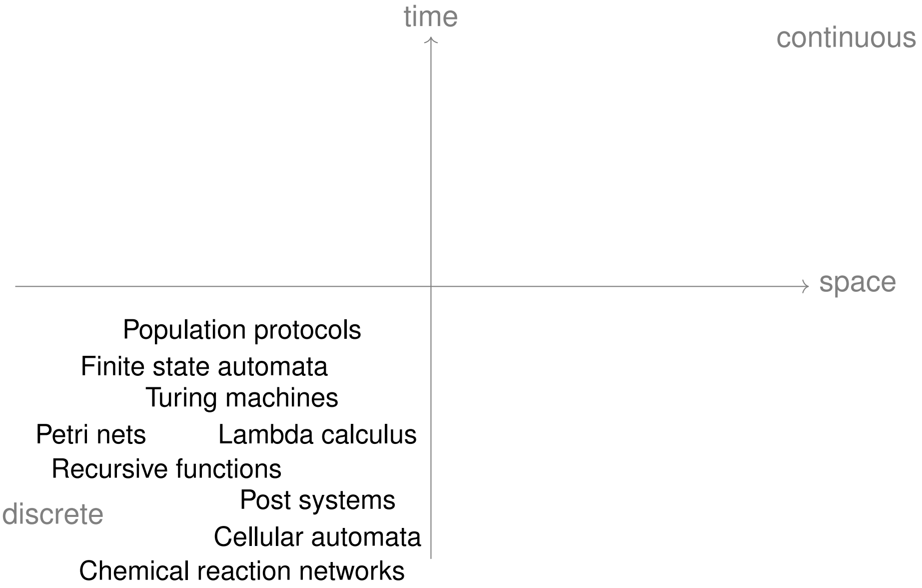
Classifying machines/models



Classifying machines/models



The many many models



The many many models

time

continuous

space

discrete

Population protocols
Finite state automata
Turing machines
Petri nets
Lambda calculus
Recursive functions
Post systems
Cellular automata
Chemical reaction networks

Continuous Automata
Deep learning models
Neural networks
Blum Shub Smale machines
Hybrid systems
Natural computing influence dynamics
Signal machines

The many many models

time

continuous

$\mathbb{R}-$recursive functions
Large population protocols
Hybrid Systems
Timed automata
Physarum computing
Shannon's GPAC
Black hole models
Hopfield's neural networks
Reaction-Diffusion Systems
Chemical reaction networks

space

Continuous Automata
Deep learning models
Neural networks
Blum Shub Smale machines
Hybrid systems
Natural computing influence dynamics
Signal machines

Population protocols
Finite state automata
Turing machines
Petri nets
Lambda calculus
Recursive functions
Post systems
Cellular automata
Chemical reaction networks

discrete

The many many models

time

$\mathbb{R}-$recursive functions

continuous

Large population protocols

Hybrid Systems

Timed automata

Physarum computing

Boolean difference equation models

Shannon's GPAC

Black hole models

Hopfield's neural networks

Reaction-Diffusion Systems

Chemical reaction networks

space

Population protocols

Continuous Automata

Finite state automata

Deep learning models

Turing machines

Neural networks

Petri nets

Lambda calculus

Blum Shub Smale machines

Recursive functions

Hybrid systems

Post systems

Natural computing influence dynamics

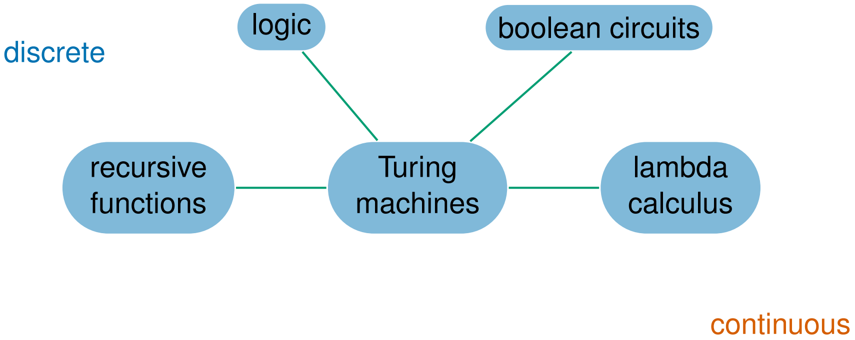
discrete

Cellular automata

Signal machines

Chemical reaction networks

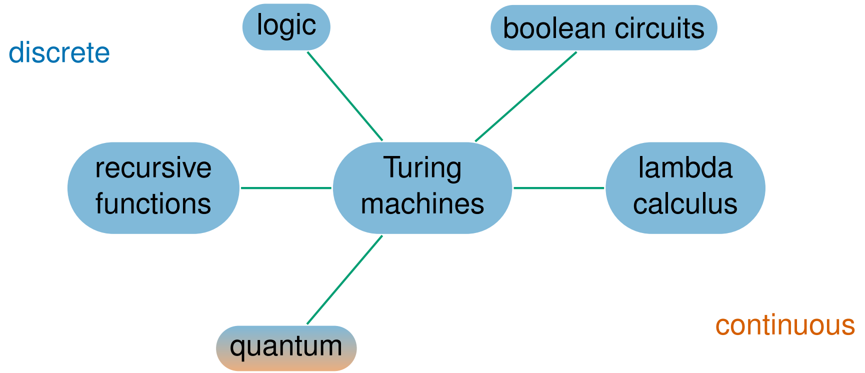
Making sense of all these models



“Church” thesis

All discrete models are Turing machine-computable.

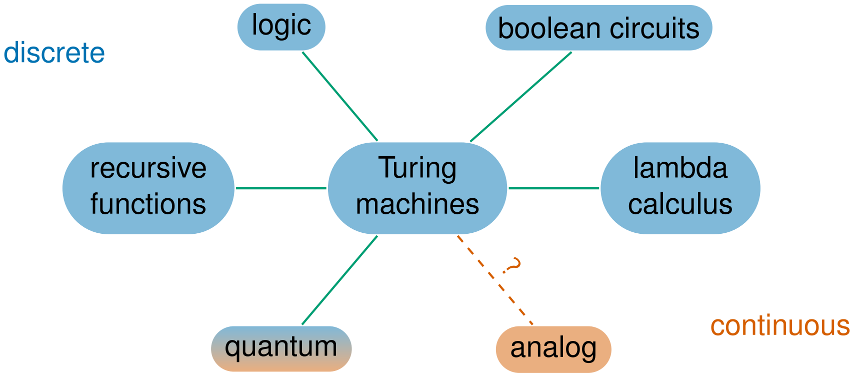
Making sense of all these models



“Church” thesis

All discrete models are Turing machine-computable.

Making sense of all these models

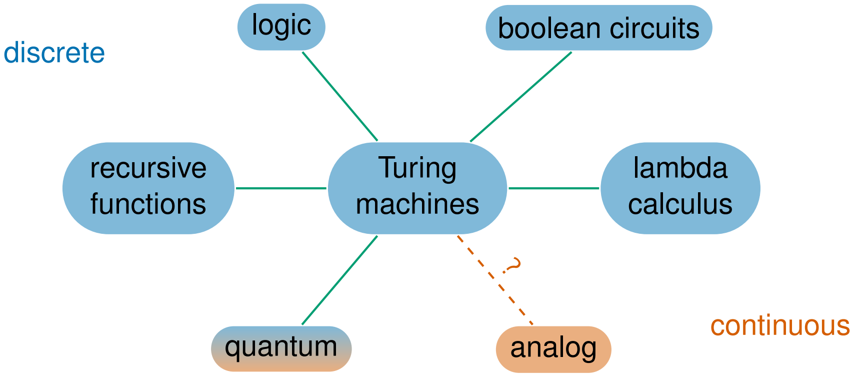


“Church” thesis ?

All models are Turing machine-computable.

Clearly **wrong**: a single real number (Ω of Chaitin) is super-Turing powerful.

Making sense of all these models



"Church" thesis ?

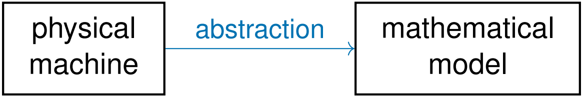
All **physical machine-based** models are Turing machine-computable.

Several issues with that statement.

Machine vs mathematical model

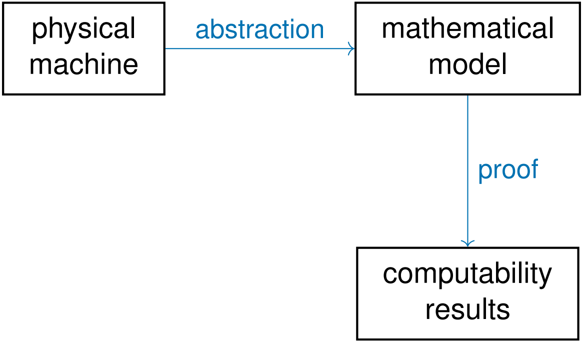
physical
machine

Machine vs mathematical model



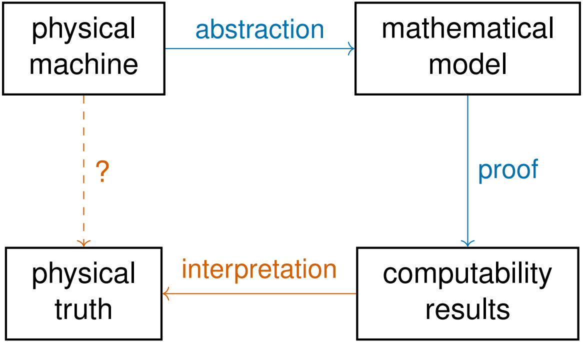
- mathematical model = abstraction of a system

Machine vs mathematical model



- mathematical model = abstraction of a system
- properties of model ≠ properties of system

Machine vs mathematical model



- mathematical model = abstraction of a system
- properties of model ≠ properties of system
- conclusion might be quantitatively or qualitatively wrong

Black hole model and hypercomputations

- machine: the universe
- model: general relativity

Picture black hole

Black hole model and hypercomputations

- machine: the universe
- model: general relativity

Picture black hole

Informal theorem

If slowly rotating Kerr black holes exists, one can check consistency of ZFC or solve the Turing halting problem in finite time.

- conclusion: hypercomputations are possible ?

Black hole model and hypercomputations

- machine: the universe
- model: general relativity

Picture black hole

Informal theorem

If slowly rotating Kerr black holes exists, one can check consistency of ZFC or solve the Turing halting problem in finite time.

- conclusion: hypercomputations are possible ?

Common occurrence in analog models: non-computable reals, Zeno phenomena, ...

Back to the Church thesis

Distinguish machines from models:

Actual Church thesis

Every effective computation can be carried out by a Turing machine, and vice versa.

$\Rightarrow$ effective = systematic method in logic/mathematics/CS

Back to the Church thesis

Distinguish machines from models:

Actual Church thesis

Every effective computation can be carried out by a Turing machine, and vice versa.

$\Rightarrow$ effective = systematic method in logic/mathematics/CS

Physical Church Turing thesis/Thesis M

Whatever can be calculated by a machine (with finite data/instructions) is Turing machine-computable.

$\Rightarrow$ machine that conforms to the physical laws

Back to the Church thesis

Distinguish machines from models:

Actual Church thesis

Every effective computation can be carried out by a Turing machine, and vice versa.

$\Rightarrow$ effective = systematic method in logic/mathematics/CS

Physical Church Turing thesis/Thesis M

Whatever can be calculated by a machine (with finite data/instructions) is Turing machine-computable.

$\Rightarrow$ machine that conforms to the physical laws

Alternative thesis

All **reasonable** models of computations are equivalent to Turing machines.

Chemical Reaction Networks (CRNs)

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 (any resemblance to chemistry is purely coincidental):

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

Chemical Reaction Networks (CRNs)

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 (any resemblance to chemistry is purely coincidental):

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

Semantics (assuming law of mass action):

- discrete
- differential
- stochastic

Chemical Reaction Networks (CRNs)

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 (any resemblance to chemistry is purely coincidental):

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

Semantics (assuming law of mass action):

- discrete $\rightarrow$
- differential
- stochastic

$y_i =$ molecule count

close to population protocols

Chemical Reaction Networks (CRNs)

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 (any resemblance to chemistry is purely coincidental):

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

Semantics (assuming law of mass action):

- discrete
- differential $\rightarrow$
- stochastic

$y_i =$ concentration

polynomial ODEs

Chemical Reaction Networks (CRNs)

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 (any resemblance to chemistry is purely coincidental):

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

Semantics (assuming law of mass action):

- discrete
- differential
- stochastic $\rightarrow$

y_i = probability distribution

stochastic ODEs

Chemical Reaction Networks (CRNs)

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 (any resemblance to chemistry is purely coincidental):

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

Semantics (assuming law of mass action):

- discrete
- differential
- stochastic

Observation

A system/machine can have several models, all useful, depending on the level of abstraction.

Are analog systems capable of hypercomputations?

Examples: Black holes, signal machines, hybrid systems

Are analog systems capable of hypercomputations?

Examples: Black holes, signal machines, hybrid systems

Let's do something useful with it!

Are analog systems capable of hypercomputations?

Examples: Black holes, signal machines, hybrid systems

Let's do something useful with it!

Something is wrong, change the model.

Are analog systems capable of hypercomputations?

Examples: Black holes, signal machines, hybrid systems

Let’s do something useful with it!

Something is wrong, change the model.

Let’s study it! *Especially* if it doesn’t exist.

Are analog systems capable of hypercomputations?

Examples: Black holes, signal machines, hybrid systems

Let's do something useful with it!

Something is wrong, change the model.

Even if it exists, it cannot be verified.

Let's study it! *Especially* if it doesn't exist.

Are analog systems capable of hypercomputations?

Examples: Black holes, signal machines, hybrid systems

Let's do something useful with it!

Something is wrong, change the model.

Even if it exists, it cannot be verified.

Let's study it! *Especially* if it doesn't exist.

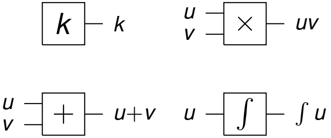
Possible conclusion

All **reasonable** models of computations are equivalent to Turing machines. Hypercomputability results can help us correct models.

General Purpose Analog Computer (GPAC)

Differential analyzer

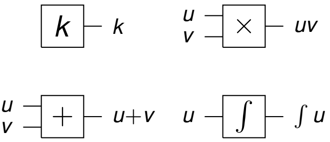
General Purpose Analog Computer (GPAC)



General Purpose Analog Computer, Shannon 1936

Differential analyzer

General Purpose Analog Computer (GPAC)

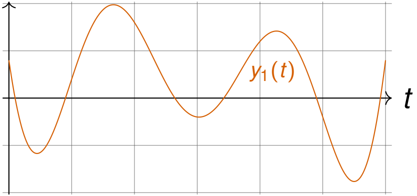


General Purpose Analog Computer, Shannon 1936

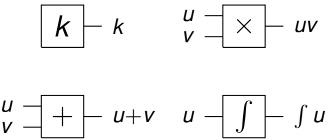
Differential analyzer

$$y(0) = y_0, \quad y'(t) = p(y(t))$$

Polynomial Differential Equation, Graça 2004



General Purpose Analog Computer (GPAC)

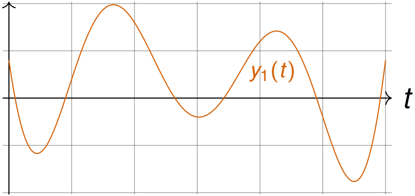


General Purpose Analog Computer, Shannon 1936

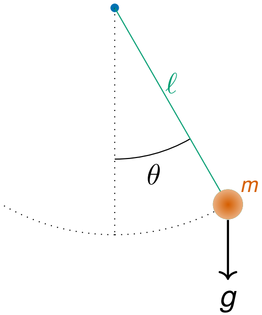
Differential analyzer

$$y(0) = y_0, \quad y'(t) = p(y(t))$$

Polynomial Differential Equation, Graça 2004

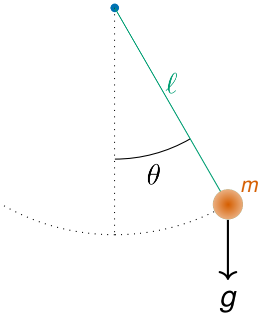


Example of dynamical system



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

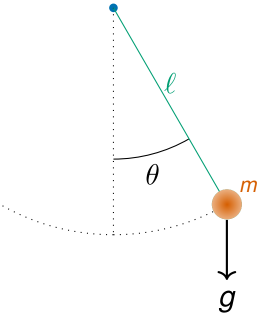
Example of dynamical system

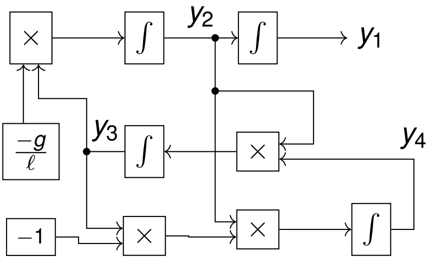


$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$

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

Example of dynamical system

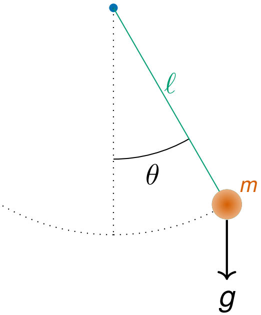


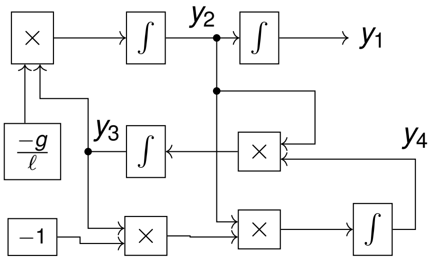


$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$

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

Example of dynamical system





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

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

Remark on “analog”

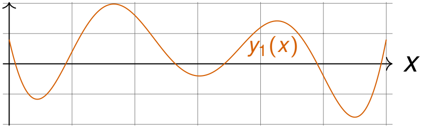
Continuous and analogy between circuits/mechanics/ODEs.

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

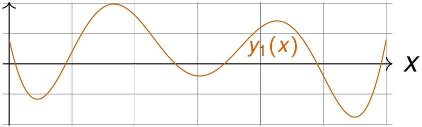
sin, cos, exp, log, ...

Computing with differential equations

Generable functions

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

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



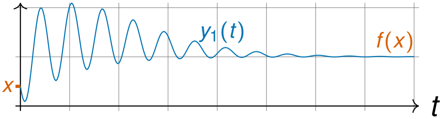
Shannon's notion

sin, cos, exp, log, ...

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

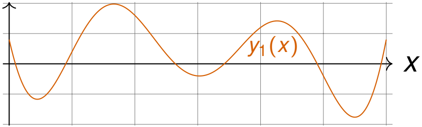
sin, cos, exp, log, Γ, ζ, ...

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

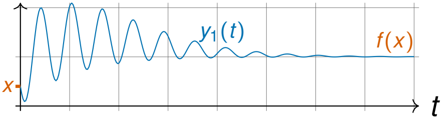
sin, cos, exp, log, ...

Considered "weak": not Γ and ζ
Only analytic functions

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

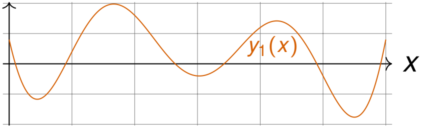
sin, cos, exp, log, Γ, ζ, ...

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

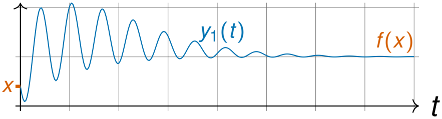
sin, cos, exp, log, ...

Considered "weak": not Γ and ζ
Only analytic functions

Computable

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

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

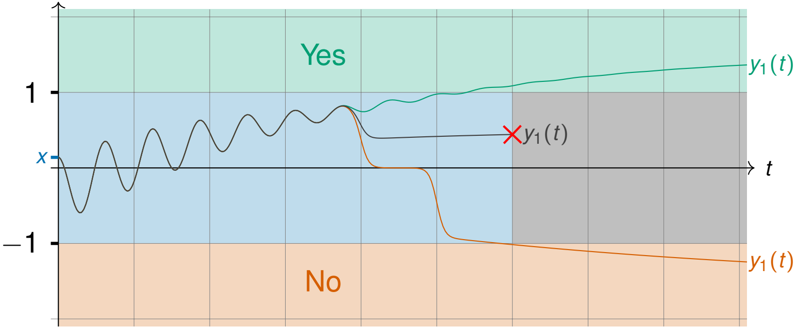


Modern notion

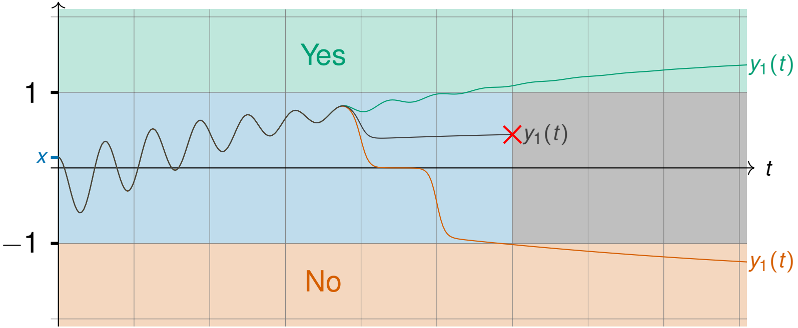
sin, cos, exp, log, Γ, ζ, ...

Turing powerful
[Bournez et al., 2007]

More formally



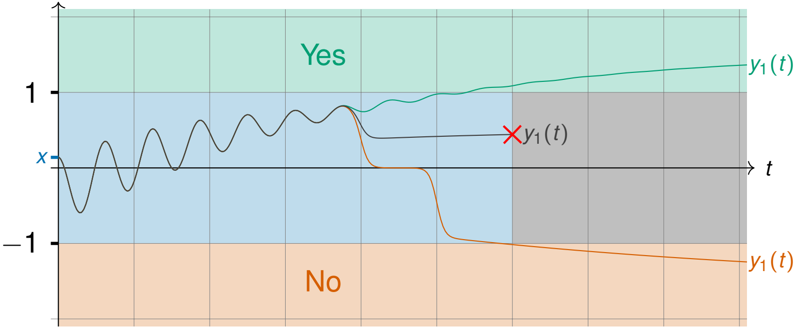
More formally



Theorem (Bournez et al, 2010)

This is equivalent to a Turing machine.

More formally



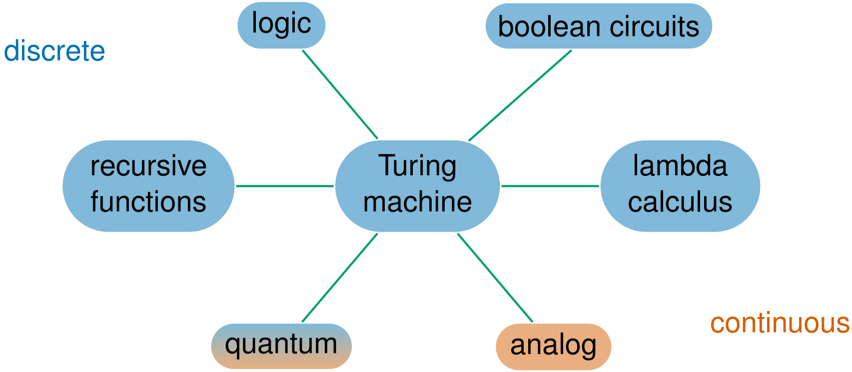
Theorem (Bournez et al, 2010)

This is equivalent to a Turing machine.

- analog computability theory
- purely continuous characterization of classical computability

Can Analog Machines Compute Faster?

Computability

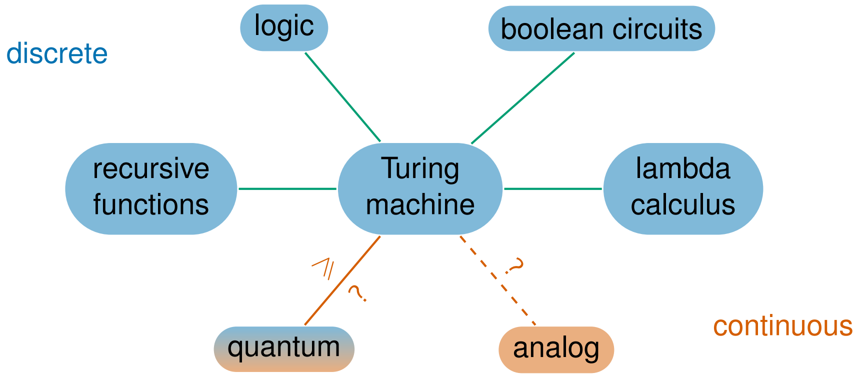


Church Thesis

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

Can Analog Machines Compute Faster?

Complexity



Effective Church Thesis

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

Complexity of analog systems

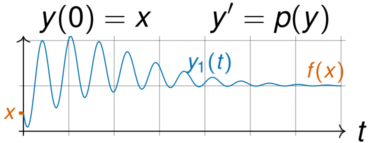
- Turing machines: $T(x) =$ number of steps to compute on x

Complexity of analog systems

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

Tentative definition

$T(x) = ??$

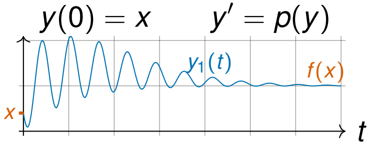


Complexity of analog systems

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

Tentative definition

$T(x, \mu) =$

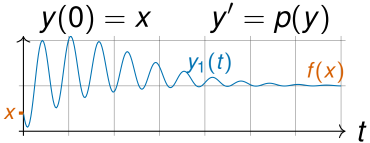


Complexity of analog systems

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

Tentative definition

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

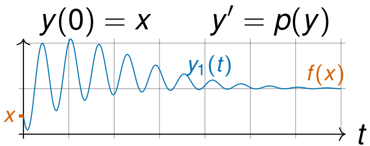


Complexity of analog systems

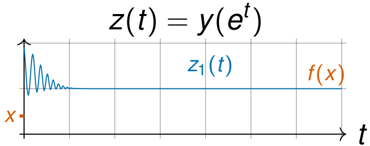
- Turing machines: $T(x)$ = number of steps to compute on x
- GPAC:

Tentative definition

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



$\rightsquigarrow$



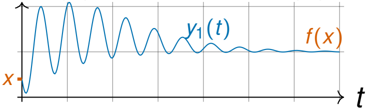
Complexity of analog systems

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

Tentative definition

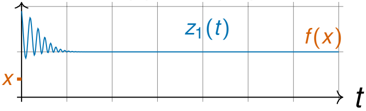
$T(x, \mu)$ = first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

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

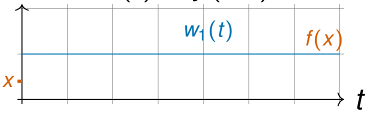


$\rightsquigarrow$

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



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



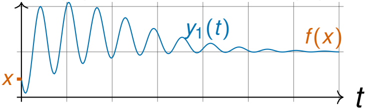
Complexity of analog systems

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

Tentative definition

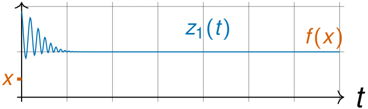
$T(x, \mu)$ = first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

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

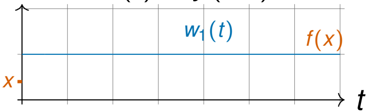


$\rightsquigarrow$

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



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

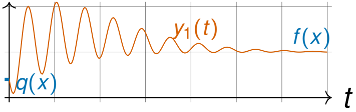


Something is wrong...

All functions have constant time complexity.

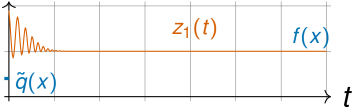
Time-space correlation of the GPAC

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



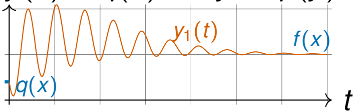
$\rightsquigarrow$

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



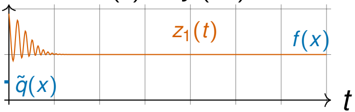
Time-space correlation of the GPAC

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

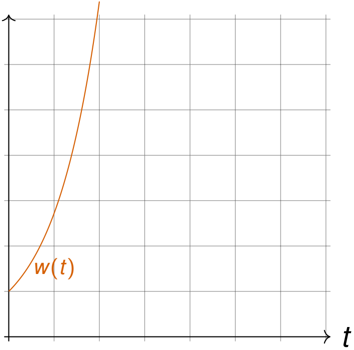


$\rightsquigarrow$

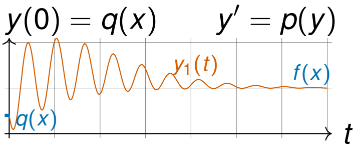
$z(t) = y(e^t)$



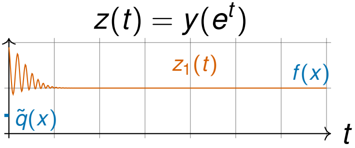
extra component: $w(t) = e^t$



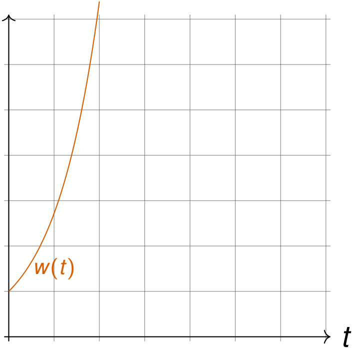
Time-space correlation of the GPAC



$\rightsquigarrow$



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



Observation

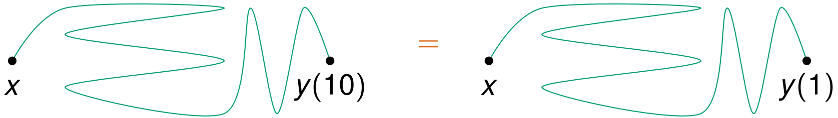
Time scaling costs "space".

$\rightsquigarrow$

Time complexity for the GPAC must involve time and space !

Complexity in the analog world

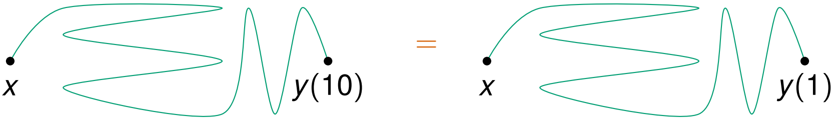
Complexity measure: length of the curve



Time acceleration: same curve = same complexity !

Complexity in the analog world

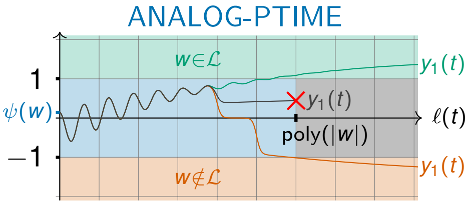
Complexity measure: length of the curve

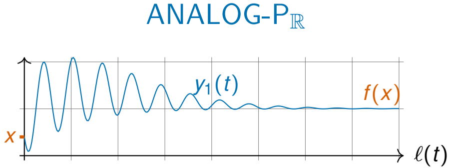


Time acceleration: same curve = same complexity !

Same time, different curves: different complexity !

Analog complexity



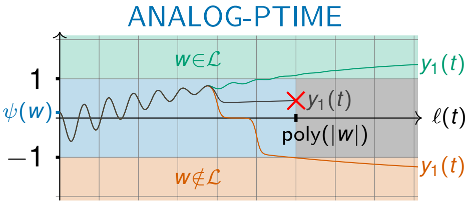


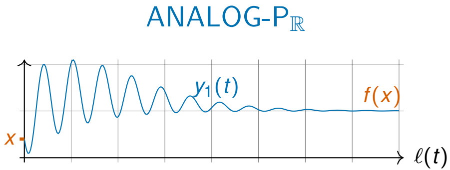
Theorem

- $\mathcal{L} \in$ 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

Analog complexity





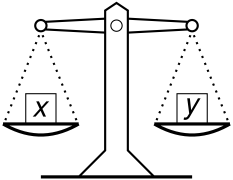
Theorem

- $\mathcal{L} \in$ 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 ⇔ length of the GPAC
- Purely continuous characterization of PTIME
- Only rational coefficients needed

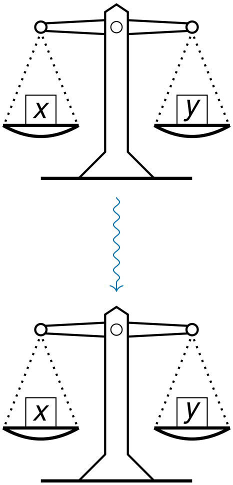
Does a balance scale compute a function?

Inputs: $x, y \in [0, +\infty)$



Does a balance scale compute a function?

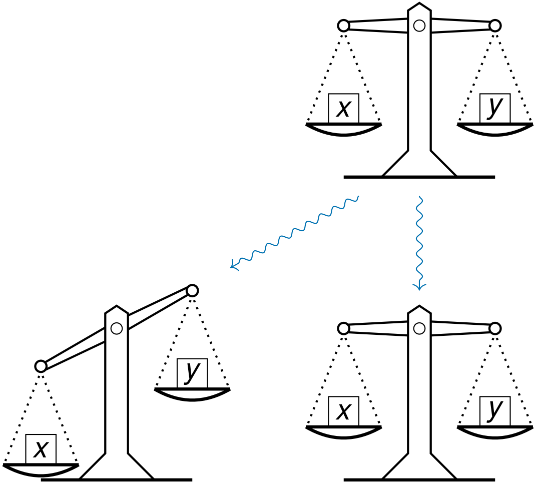
Inputs: $x, y \in [0, +\infty)$



$x = y$

Does a balance scale compute a function?

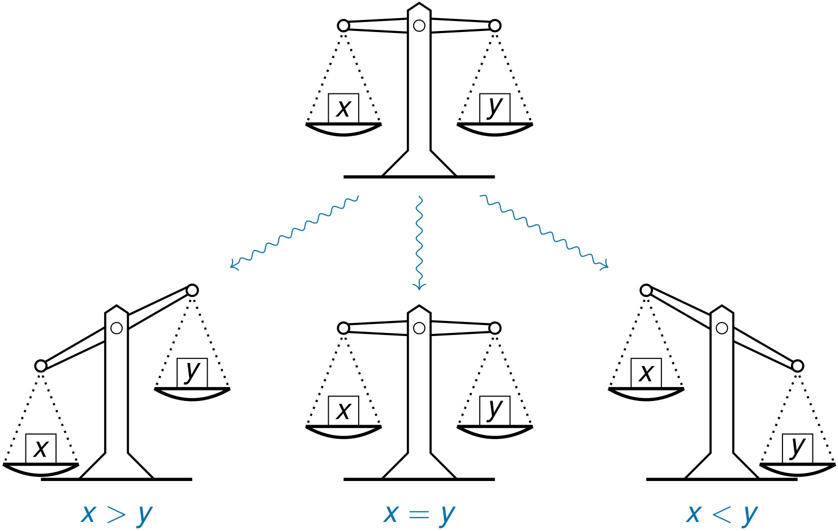
Inputs: $x, y \in [0, +\infty)$



$x > y$ $\qquad$ $x = y$

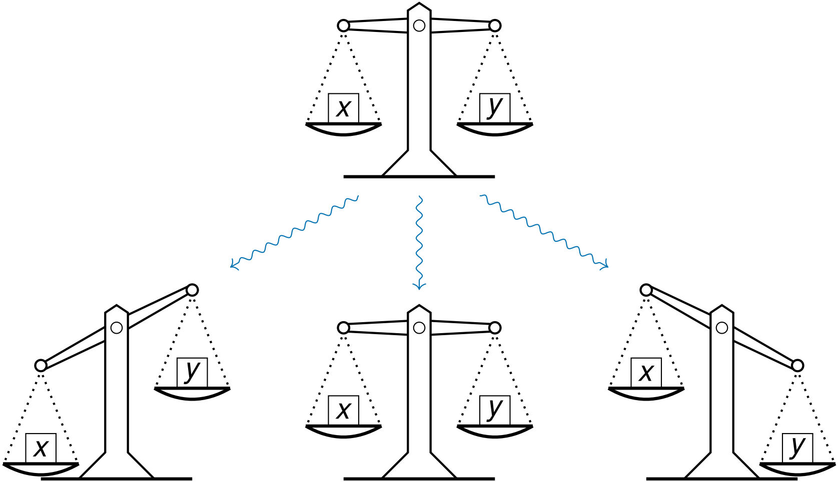
Does a balance scale compute a function?

Inputs: $x, y \in [0, +\infty)$



Does a balance scale compute a function?

Inputs: $x, y \in [0, +\infty)$

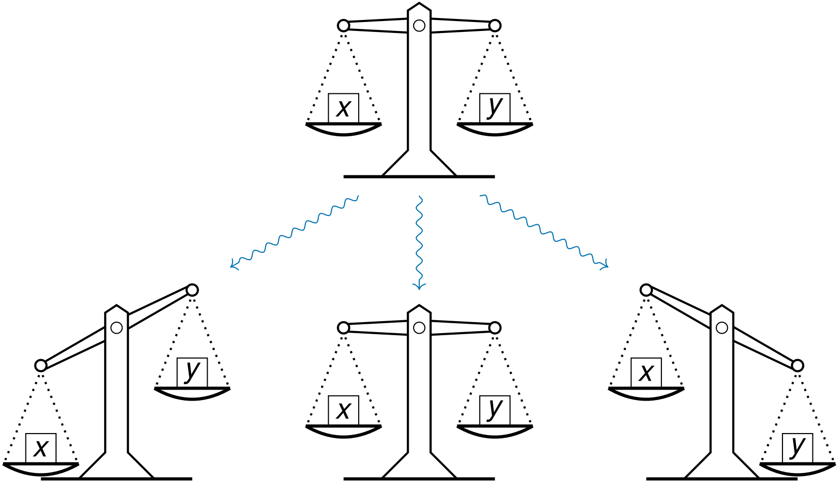


$x > y$ $x = y$ $x < y$

Output: $\text{sign}(x - y)$?

Does a balance scale compute a function?

Inputs: $x, y \in [0, +\infty)$



$x > y$ $x = y$ $x < y$

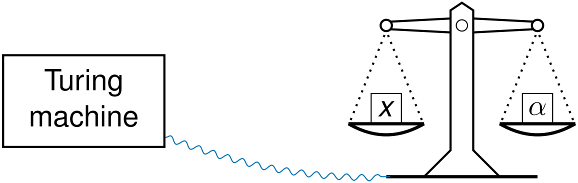
Output: $\text{sign}(x - y)$?

Complexity: ???

Physical Oracles (Beggs, Costa, Poças and Tucker)

Model: Turing Machine with “physical oracle”
Oracle: performs physical experiments with time limit
Outcomes: Yes, No, Timeout

Example: $\alpha \in [0, 1]$ unknown, x programmable

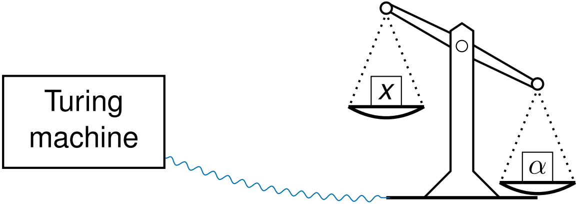


Queries:

Physical Oracles (Beggs, Costa, Poças and Tucker)

Model: Turing Machine with "physical oracle"
Oracle: performs physical experiments with time limit
Outcomes: Yes, No, Timeout

Example: $\alpha \in [0, 1]$ unknown, x programmable



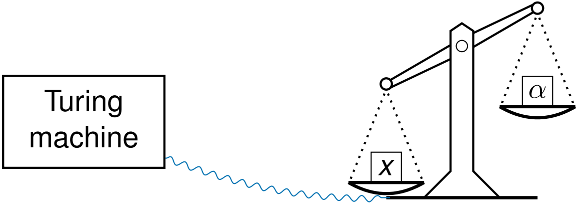
Queries:

- $x = \frac{1}{2}$, $T = 1$ second $\rightsquigarrow$ Yes

Physical Oracles (Beggs, Costa, Poças and Tucker)

Model: Turing Machine with “physical oracle”
Oracle: performs physical experiments with time limit
Outcomes: Yes, No, Timeout

Example: $\alpha \in [0,1]$ unknown, x programmable



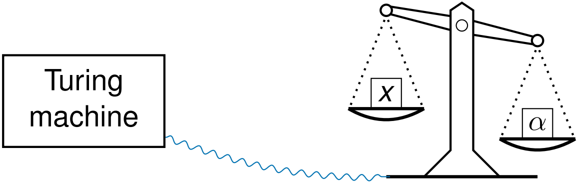
Queries:

- $x = \frac{1}{2}$, $T = 1$ second $\rightsquigarrow$ Yes
- $x = \frac{3}{4}$, $T = 1$ second $\rightsquigarrow$ No

Physical Oracles (Beggs, Costa, Poças and Tucker)

Model: Turing Machine with "physical oracle"
Oracle: performs physical experiments with time limit
Outcomes: Yes, No, Timeout

Example: $\alpha \in [0,1]$ unknown, x programmable



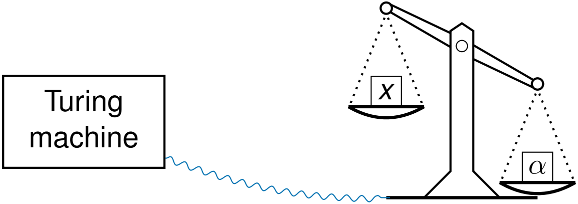
Queries:

- $x = \frac{1}{2}$, $T = 1$ second $\rightsquigarrow$ Yes
- $x = \frac{3}{4}$, $T = 1$ second $\rightsquigarrow$ No
- $x = \frac{5}{8}$, $T = 1$ second $\rightsquigarrow$ Timeout

Physical Oracles (Beggs, Costa, Poças and Tucker)

Model: Turing Machine with “physical oracle”
Oracle: performs physical experiments with time limit
Outcomes: Yes, No, Timeout

Example: $\alpha \in [0, 1]$ unknown, x programmable

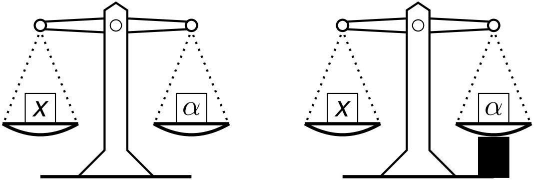


Queries:

- $x = \frac{1}{2}$, $T = 1$ second $\rightsquigarrow$ Yes
- $x = \frac{3}{4}$, $T = 1$ second $\rightsquigarrow$ No
- $x = \frac{5}{8}$, $T = 1$ second $\rightsquigarrow$ Timeout
- $x = \frac{5}{8}$, $T = 2$ seconds $\rightsquigarrow$ Yes

Physical Oracles (Beggs, Costa, Poças and Tucker)

Model: Turing Machine with “physical oracle”
Oracle: performs physical experiments with time limit
Outcomes: Yes, No, Timeout



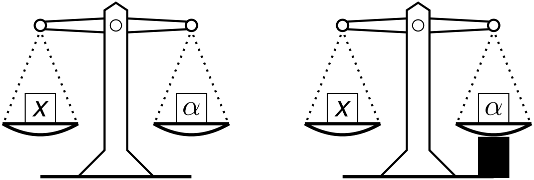
Wheatstone bridge
or
Brewster’s angle
experiment

Experiments types:

- Two-sided: time $\frac{C}{|x-\alpha|^d}$, Yes/No/Timeout
- One-sided: time $\frac{C}{|x-\alpha|^d}$, Yes/Timeout
- Vanishing: Yes (if $x \neq \alpha$)/Timeout, can test if $|x - \alpha| \leqslant |x' - \alpha|$

Physical Oracles (Beggs, Costa, Poças and Tucker)

Model: Turing Machine with "physical oracle"
Oracle: performs physical experiments with time limit
Outcomes: Yes, No, Timeout



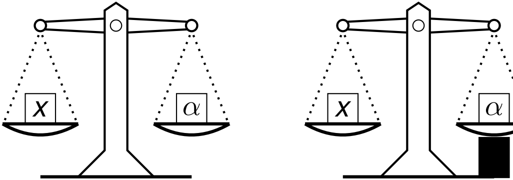
Wheatstone bridge
or
Brewster's angle
experiment

Experiments types:

- Two-sided: time $\frac{C}{|x-\alpha|^d}$, Yes/No/Timeout
- One-sided: time $\frac{C}{|x-\alpha|^d}$, Yes/Timeout
- Vanishing: Yes (if $x \neq \alpha$)/Timeout, can test if $|x-\alpha| \leqslant |x'-\alpha|$

Precision: infinite, unbounded, fixed

Physical Oracles (Beggs, Costa, Poças and Tucker)



Wheatstone bridge
or
Brewster's angle experiment

Experiments types:

- Two-sided: time $\frac{C}{|x-\alpha|^d}$, Yes/No/Timeout
- One-sided: time $\frac{C}{|x-\alpha|^d}$, Yes/Timeout
- Vanishing: Yes (if $x \neq \alpha$)/Timeout, can test if $|x-\alpha| \leqslant |x'-\alpha|$

Precision: infinite, unbounded, fixed

Theorem (Beggs, Costa, Poças and Tucker)

For a broad class of oracles + PTIME machine, complexity bounded by[1] BPP//log$\star$, or P/poly if non-computable analog-digital interface.

[1] BPP + non-uniform log advise.

Summary

- analog: analogy/continuous, orthogonal meanings
- machines vs models: need to distinguish concepts
- “Church” thesis: subtle, several variants
- hypercomputability: various interpretations
- some reasonable models exists: GPAC, equivalent to TM
- complexity: difficult to define in general, several approaches