Analogue models of computation: computing with real numbers

Manon BLANC

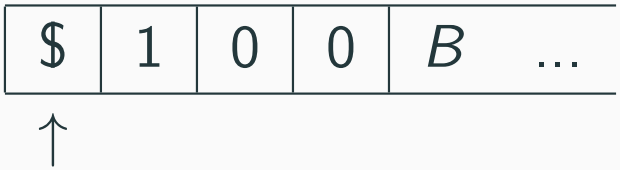
Practices around Computation

We need to talk about a discrete model

An example of non-analogue model

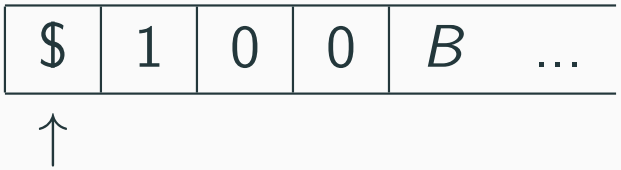
A mathematical representation: the Turing machine

The model:



A mathematical representation: the Turing machine

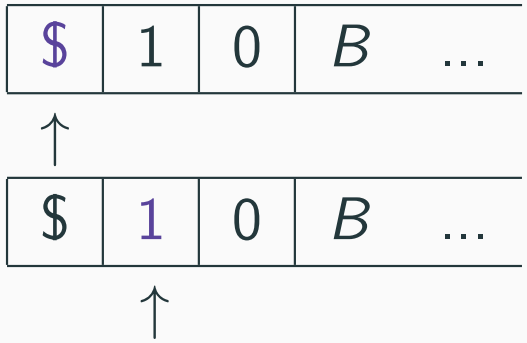
The model:



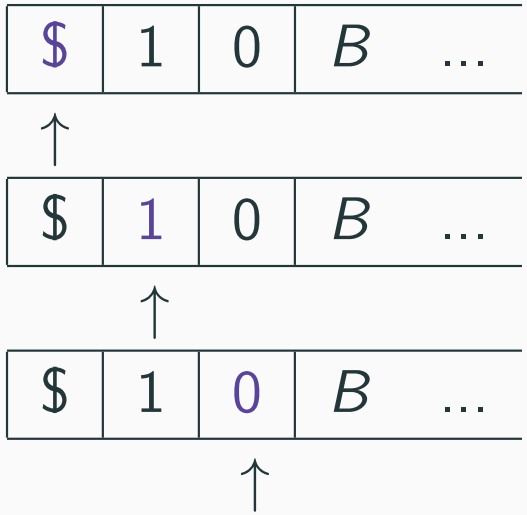
Physical World	**Mathematical Model**
Laptop	Turing machines

Example: the addition +1

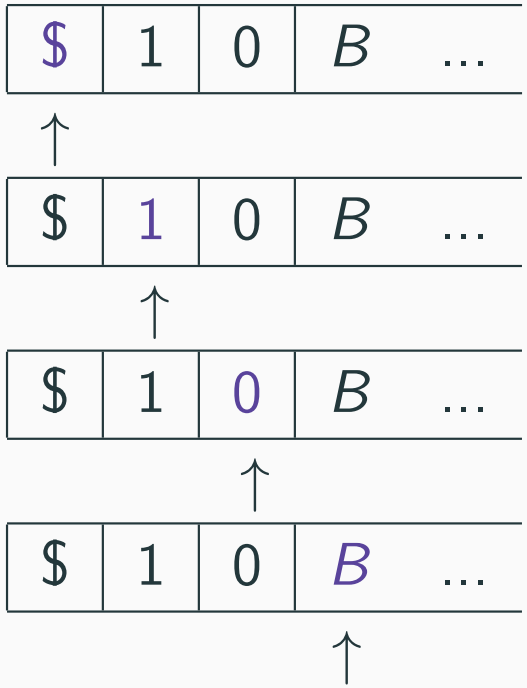
Example: the addition +1



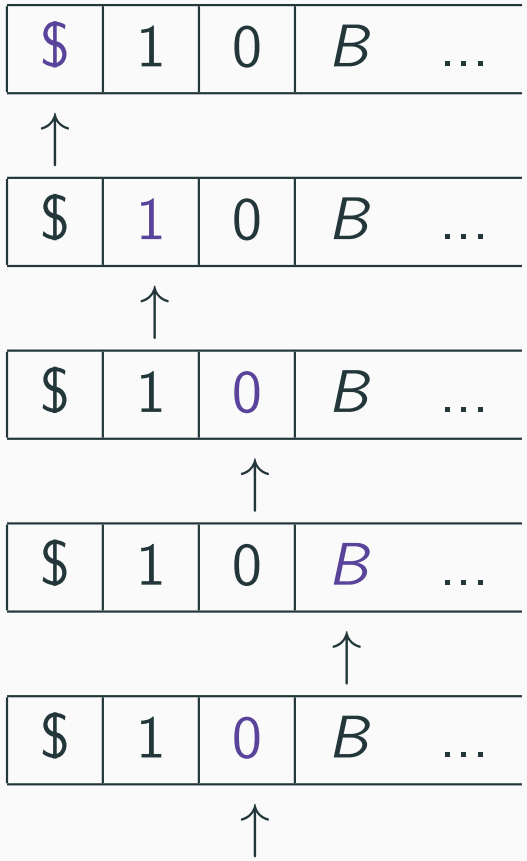
Example: the addition +1



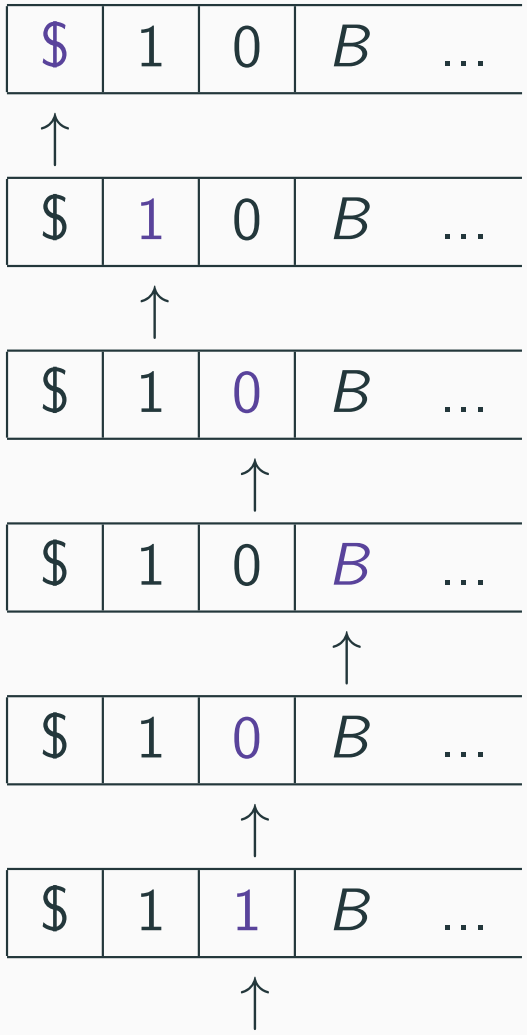
Example: the addition +1



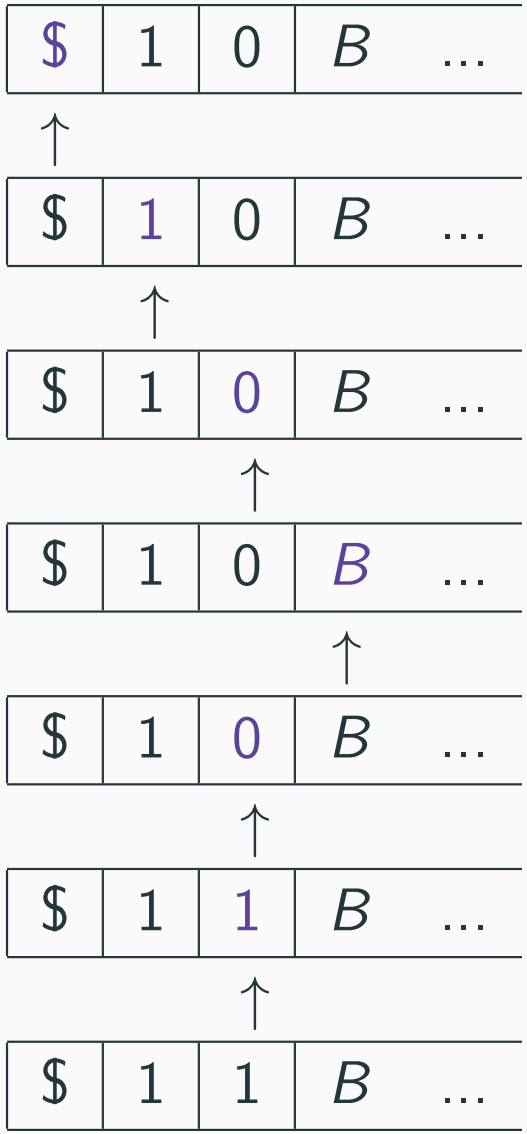
Example: the addition +1



Example: the addition +1



Example: the addition +1



A bit more formally

(we assume the tape is non-empty)

	\$	0	1	B
q_{init}	$q_{init}, \$, \rightarrow$	$q_{init}, 0, \rightarrow$	$q_{init}, 1, \rightarrow$	$q_1, B, \leftarrow$
q_1		$accept, 1$	$q_2, 0, \leftarrow$	
q_2	$q_3, \$, \rightarrow$	$accept, 1$	$q_2, 1, \leftarrow$	
q_3		$q_4^0, 1, \rightarrow$		
q_4^0		$q_4^0, 0, \rightarrow$		$accept, 0$

A notion of cost for TM

- Time = number of steps

A notion of cost for TM

- Time = number of steps
- Space = number of needed cells

On our example

Time : 6 steps $\rightarrow O(n)$, n the size of the input

Space : 2 cells $\rightarrow O(n)$, n the size of the input

Both are linear (in the size of the input), so *polynomial* in time and space.

Robust correspondance

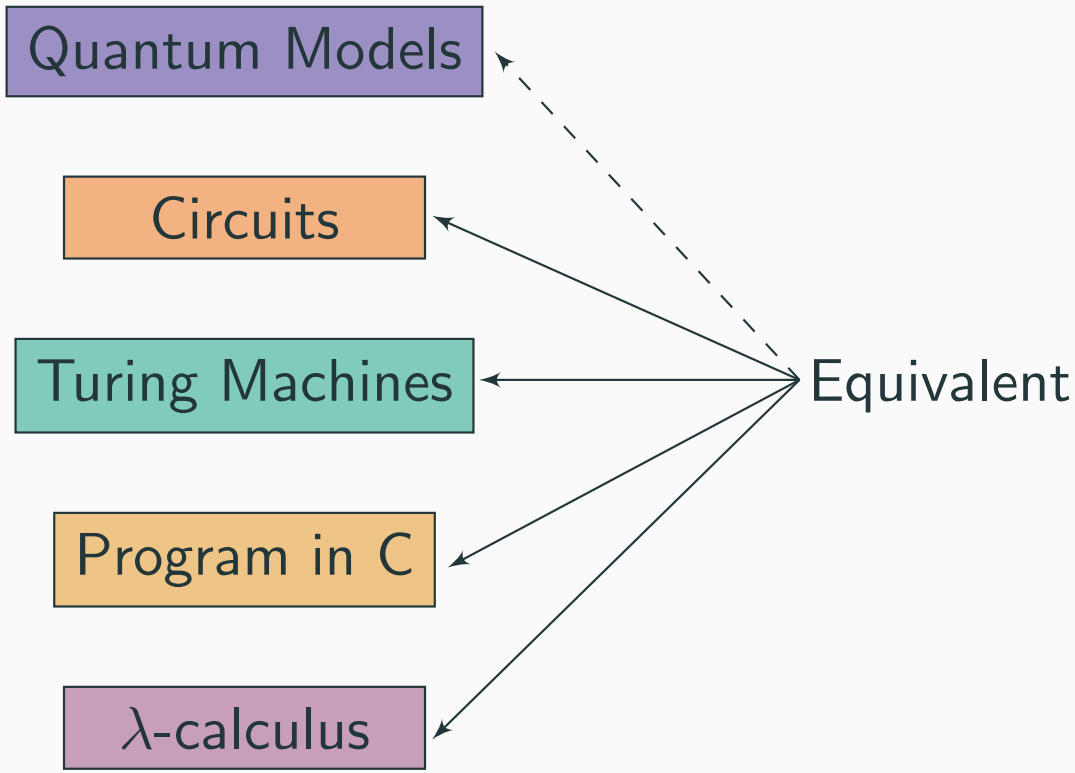
- Time of a TM $\sim$ Time on the laptop...

$\rightarrow$ Notions of time and space does not depend on the model

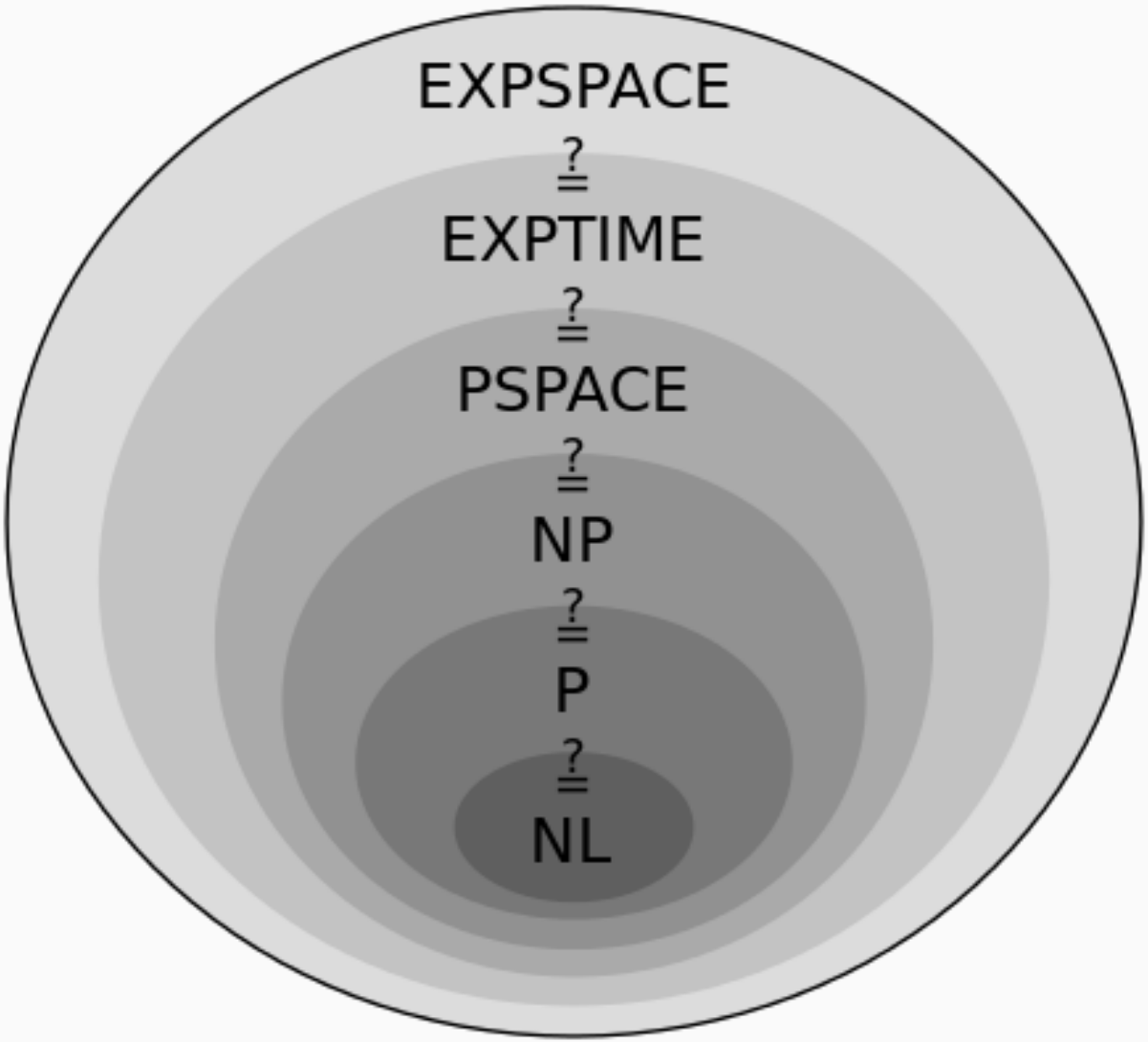
Computational Power

Church-Turing thesis: All discrete sufficiently powerful and reasonable models of computation have the same computational power as a Turing machine.

Power: boxes


Quantum Models
Circuits
Turing Machines
Program in C
λ-calculus
Equivalent

Putting boxes into boxes (but they are very nice boxes)



What an analogue world

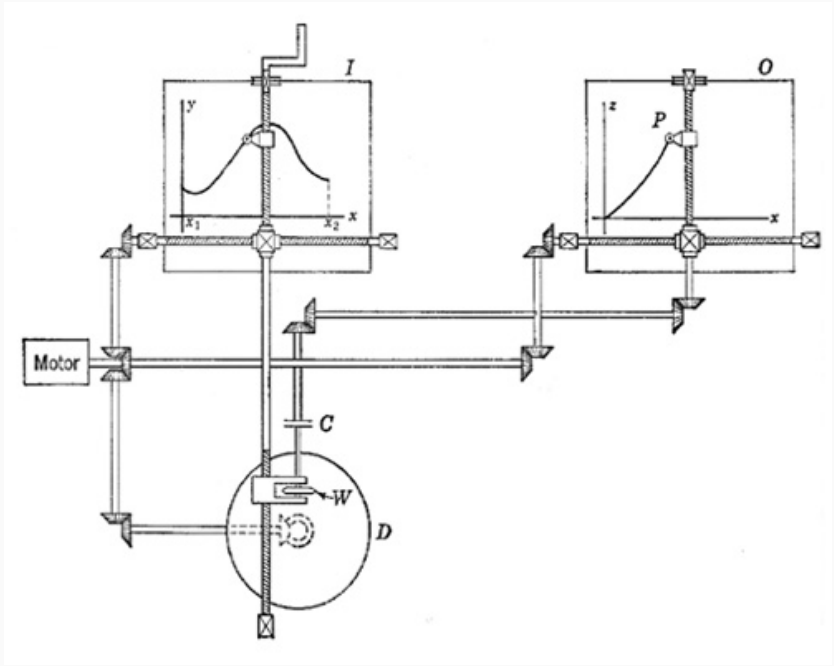
First

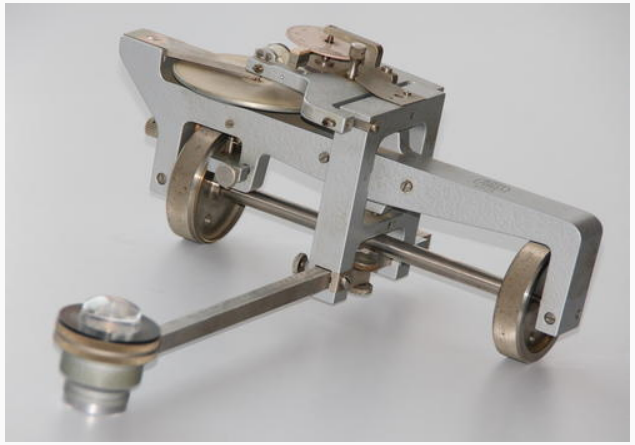
Historicaly, they are the first computational machines and certainely the most accurate

$\rightarrow$ Nothing is lost in translation

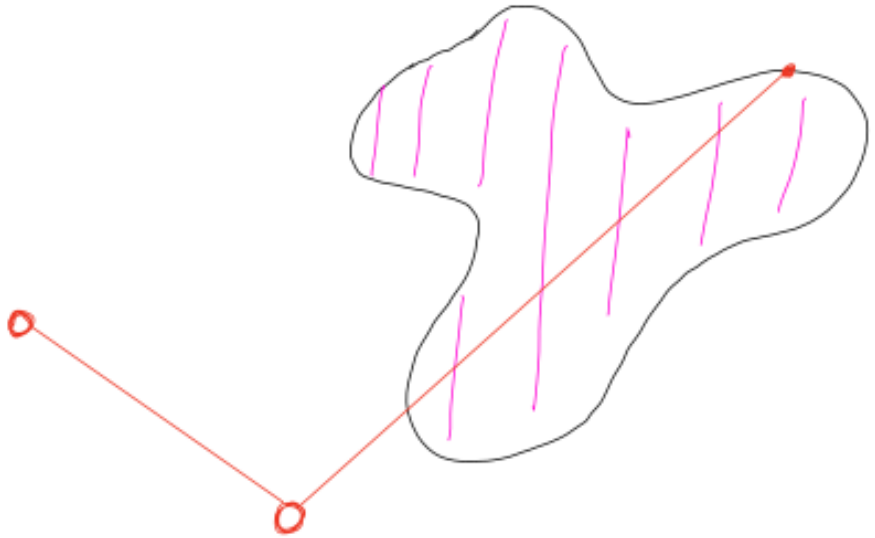
Well, actually... An example of analogue model

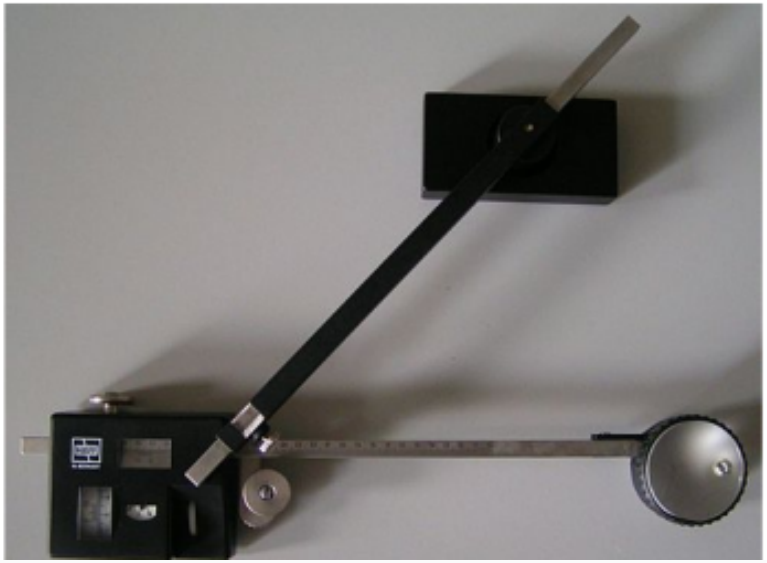
We live in an analogue world

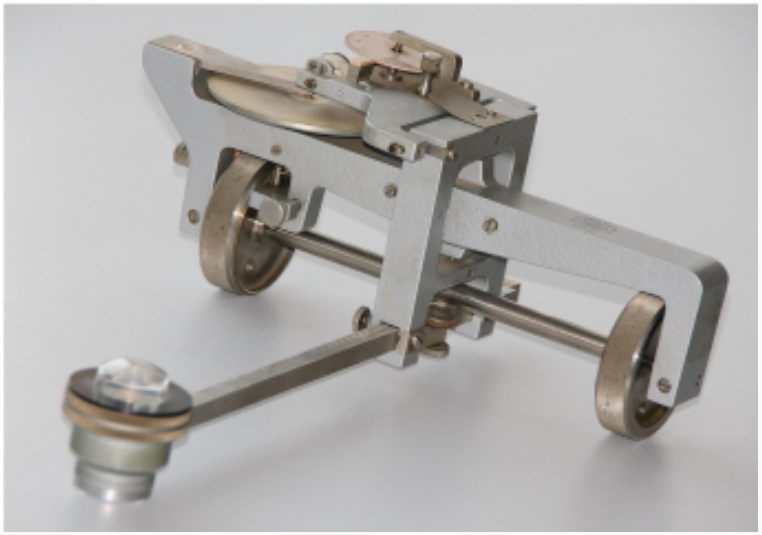




We live in an analogue world: planimeter







We live in an analogue world:

How does it work?

- Making: convert the sound into a print on a support in a spiraling groove;

How does it work?

- Making: convert the sound into a print on a support in a spiraling groove;
- Playing:
  - A head is placed on the groove;

How does it work?

- Making: convert the sound into a print on a support in a spiraling groove;
- Playing:
  - A head is placed on the groove;
  - It transmit the deviations of the groove to an electromagnetic transductor;

How does it work?

- Making: convert the sound into a print on a support in a spiraling groove;

- Playing:
  - A head is placed on the groove;
  - It transmit the deviations of the groove to an electromagnetic transductor;
  - An audio power amplifier allows un to actually hear the sound

Better or worse?

Advantages:

- We capture the real sound
- Less mixing needed

Drawbacks:

- Worn out faster
- More fragile
- It is big

Too specific use

One device for one use: can we construct an "equivalent" of the physical computer, but for analogue model?

$\rightarrow$ is there a *general purpose* device?

The GPAC

Where are we now?

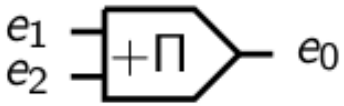
Physical World	Mathematical Model
Laptop	Turing machines
Differential Analyzer	?

The General Purpose Analog Computer (GPAC)

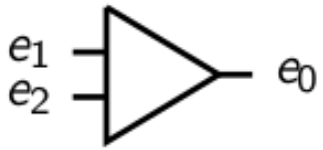
- Introduced by Claude Shannon in 1941;
- Mathematical model of the *Differential Analizer*;
- Model based on circuits, with several interconnected units doing basic operations

GPAC

constante: $e_0 = ke_1$

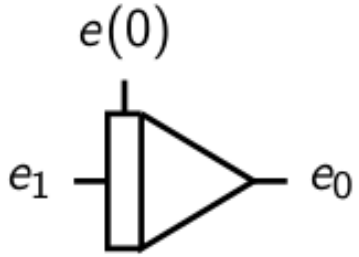


produit: $e_0 = e_1 e_2$



additionneur:

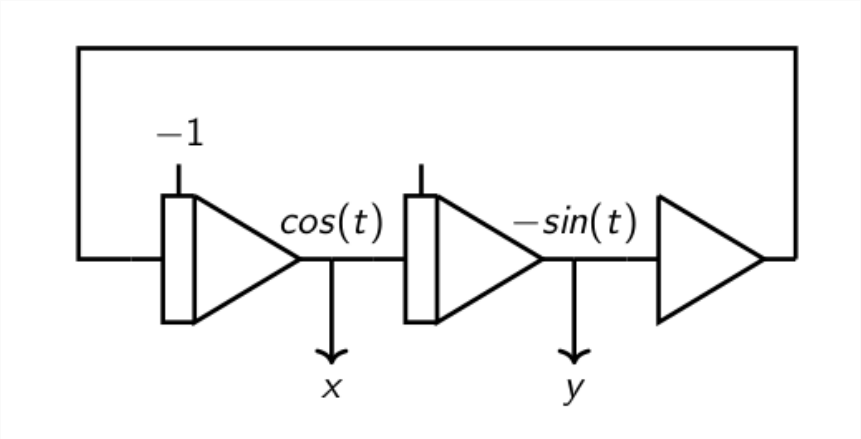
$e_0 = -(e_1 + e_2)$



intégrateur:

$e_0 = e(0) - \int_0^t e_1(u)du$

GPAC: how does it work?



$$\begin{cases} x'(t) &= y(t) \\ y'(t) &= -x(t) \\ x(0) &= -1 \\ y(0) &= 0 \end{cases} \Rightarrow \begin{cases} x(t) &= \cos(t) \\ y(t) &= -\sin(t) \end{cases}$$

Computational Power

Church-Turing thesis: All discrete sufficiently powerful and reasonable models of computation have the same computational power as a Turing machine.

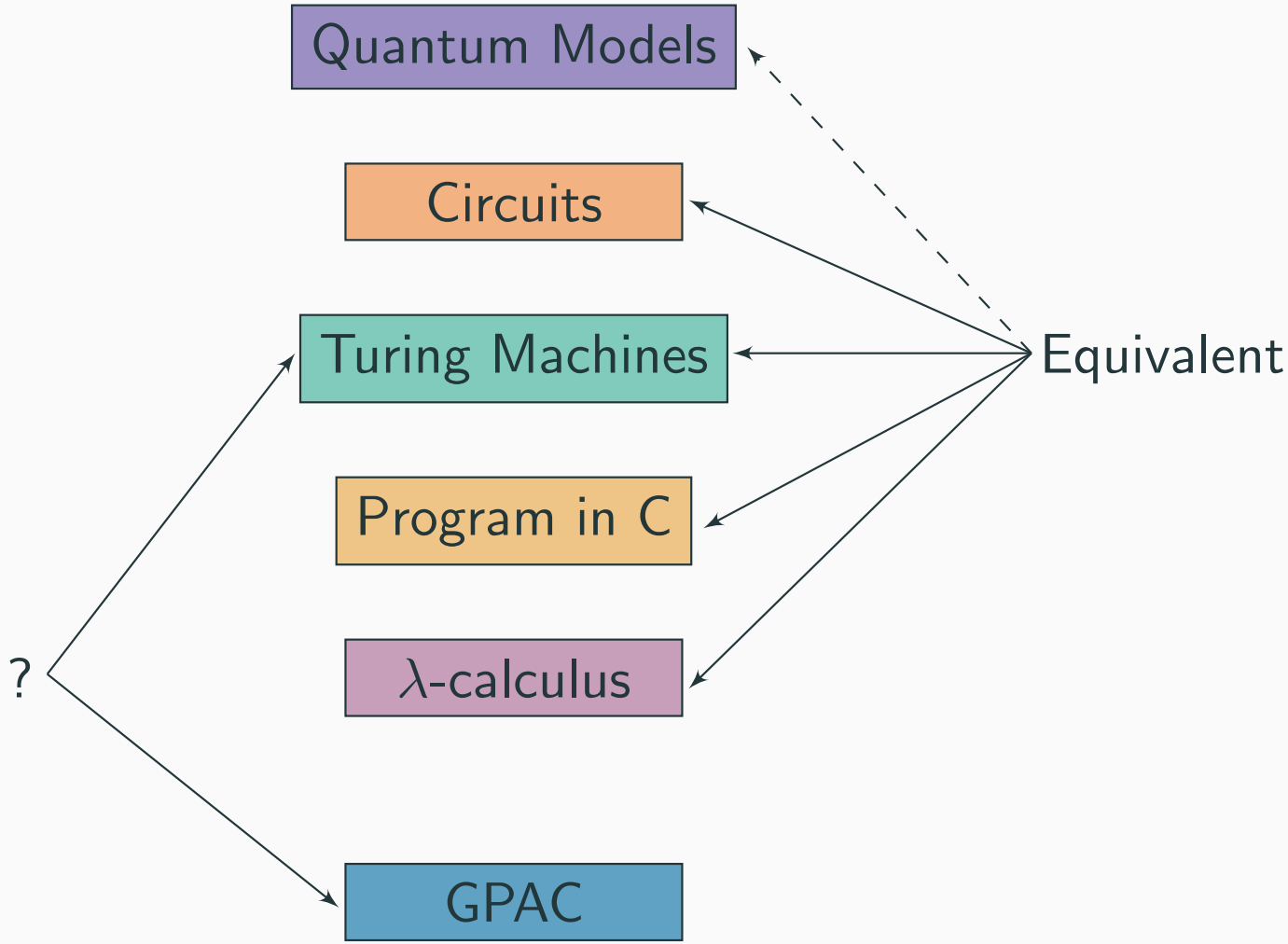
Computational Power

Church-Turing thesis: All discrete sufficiently powerful and reasonable models of computation have the same computational power as a Turing machine.

Is GPAC a sufficiently powerful?

Is GPAC reasonable?

Powerful: Other boxes



Computational power: is GPAC weaker?

- Shannon: GPAC $\leq$ Turing Machines: A GPAC corresponds to (a projection of) a *polynomial* differential equation, which can be computed by a TM.

- Shannon: GPAC $<$ Turing Machines: There exist computable functions, which are not projections of differential equations: e.g. Γ, ζ.

Well, he was wrong

We can actually program with polynomial ODE (Bournez, Graça, Pouly, Campagnolo, Hainry)

Little exercise

Transform this system of ODEs:

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

Little exercise

Transform this system of ODEs:

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a *polynomial* ODE:

$$\begin{cases} \\ \\ \\ \\ \\ \\ \end{cases} \qquad \begin{cases} \\ \\ \\ \\ \\ \\ \end{cases}$$

Little exercise

Transform this system of ODEs:

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a *polynomial* ODE:

$$\begin{cases} y_1' &= {y_3}^2 \\ & \\ & \\ & \\ & \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ & \\ & \\ & \\ & \end{cases}$$

considering $y_3 = \sin y_2$

Little exercise

Transform this system of ODEs:

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a *polynomial* ODE:

$$\begin{cases} y_1' &= {y_3}^2 \\ y_2' &= y_1 y_4 - y_5 \\ & \\ & \\ & \\ & \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ & \\ & \\ & \\ & \end{cases}$$

considering $y_3 = \sin y_2, y_4 = \cos y_2$, $y_5 = e^{e^{y_1}+t}$

Little exercise

Transform this system of ODEs:

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a *polynomial* ODE:

$$\begin{cases} y_1' &= {y_3}^2 \\ y_2' &= y_1 y_4 - y_5 \\ y_3' &= y_4(y_1 y_4 - y_5) \\ & \\ & \\ & \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ y_3(0) &= 0 \\ & \\ & \\ & \end{cases}$$

considering $y_3 = \sin y_2, y_4 = \cos y_2$, $y_5 = e^{e^{y_1}+t}$

Little exercise

Transform this system of ODEs:

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a *polynomial* ODE:

$$\begin{cases} y_1' &= {y_3}^2 \\ y_2' &= y_1 y_4 - y_5 \\ y_3' &= y_4(y_1 y_4 - y_5) \\ y_4' &= -y_3(y_1 y_4 - y_5) \\ & \\ & \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ y_3(0) &= 0 \\ y_4(0) &= 1 \\ & \\ & \end{cases}$$

considering $y_3 = \sin y_2, y_4 = \cos y_2$, $y_5 = e^{e^{y_1}+t}$

Little exercise

Transform this system of ODEs:

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a *polynomial* ODE:

$$\begin{cases} y_1' &= {y_3}^2 \\ y_2' &= y_1 y_4 - y_5 \\ y_3' &= y_4(y_1 y_4 - y_5) \\ y_4' &= -y_3(y_1 y_4 - y_5) \\ y_5' &= y_5(y_6 y_3^2 + 1) \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ y_3(0) &= 0 \\ y_4(0) &= 1 \\ y_5(0) &= e \end{cases}$$

considering $y_3 = \sin y_2, y_4 = \cos y_2$, $y_5 = e^{e^{y_1}+t}, y_6 = e^{y_1}$

Little exercise

Transform this system of ODEs:

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a *polynomial* ODE:

$$\begin{cases} y_1' &= {y_3}^2 \\ y_2' &= y_1 y_4 - y_5 \\ y_3' &= y_4(y_1 y_4 - y_5) \\ y_4' &= -y_3(y_1 y_4 - y_5) \\ y_5' &= y_5(y_6 y_3^2 + 1) \\ y_6' &= y_6 y_3^2 \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ y_3(0) &= 0 \\ y_4(0) &= 1 \\ y_5(0) &= e \\ y_6(0) &= 1 \end{cases}$$

considering $y_3 = \sin y_2, y_4 = \cos y_2$, $y_5 = e^{e^{y_1}+t}, y_6 = e^{y_1}$

The big theorem: GPAC is sufficiently powerful

Theorem
Every computable functions can be computed by polynomial ODE, and conversely

Thus : Shannon: GPAC $\not\leq$ Turing Machines

Turing Machines $\leq$ GPAC

Is GPAC more powerful? [1]

[1] Fun fact: that's a part of my PhD subject 😅

How much does it cost?

How to redefine the cost: Time

Fig. 5. A continuous system before and after an exponential speed-up.

How to redefine the cost: Time

Fig. 5. A continuous system before and after an exponential speed-up.

θ solution of:

$$y' = f(y)$$

with $f : \mathbb{R} \to \mathbb{R}$

ϕ solution of:

$$z = z'$$

$$y' = f(y)z$$

How to redefine the cost: Time

Fig. 5. A continuous system before and after an exponential speed-up.

θ solution of:

$$y' = f(y)$$

with $f : \mathbb{R} \to \mathbb{R}$

ϕ solution of:

$$z = z'$$

$$y' = f(y)z$$

Re-scaling: $\phi_1(t) = \theta(e^t)$

How to redefine the cost: Time

Fig. 5. A continuous system before and after an exponential speed-up.

θ solution of:

$$y' = f(y)$$

with $f : \mathbb{R} \to \mathbb{R}$

ϕ solution of:

$$z = z'$$

$$y' = f(y)z$$

With this definition of time, all computation can cost 1 time unit...

How to redefine the cost: Time

Fig. 5. A continuous system before and after an exponential speed-up.

Robust notion of time $\rightarrow$ a quantity invariant by rescaling

The length remains the same

Time = Length of the curve

Now space: Why it is not easy to solve ODEs

$$\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \textit{and} \quad \frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x}),$$

- Non-computable in the general case, for $\mathbf{u}$ computable (Pour-El, Richards)

Now space: Why it is not easy to solve ODEs

$$\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \textit{and} \quad \frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x}),$$

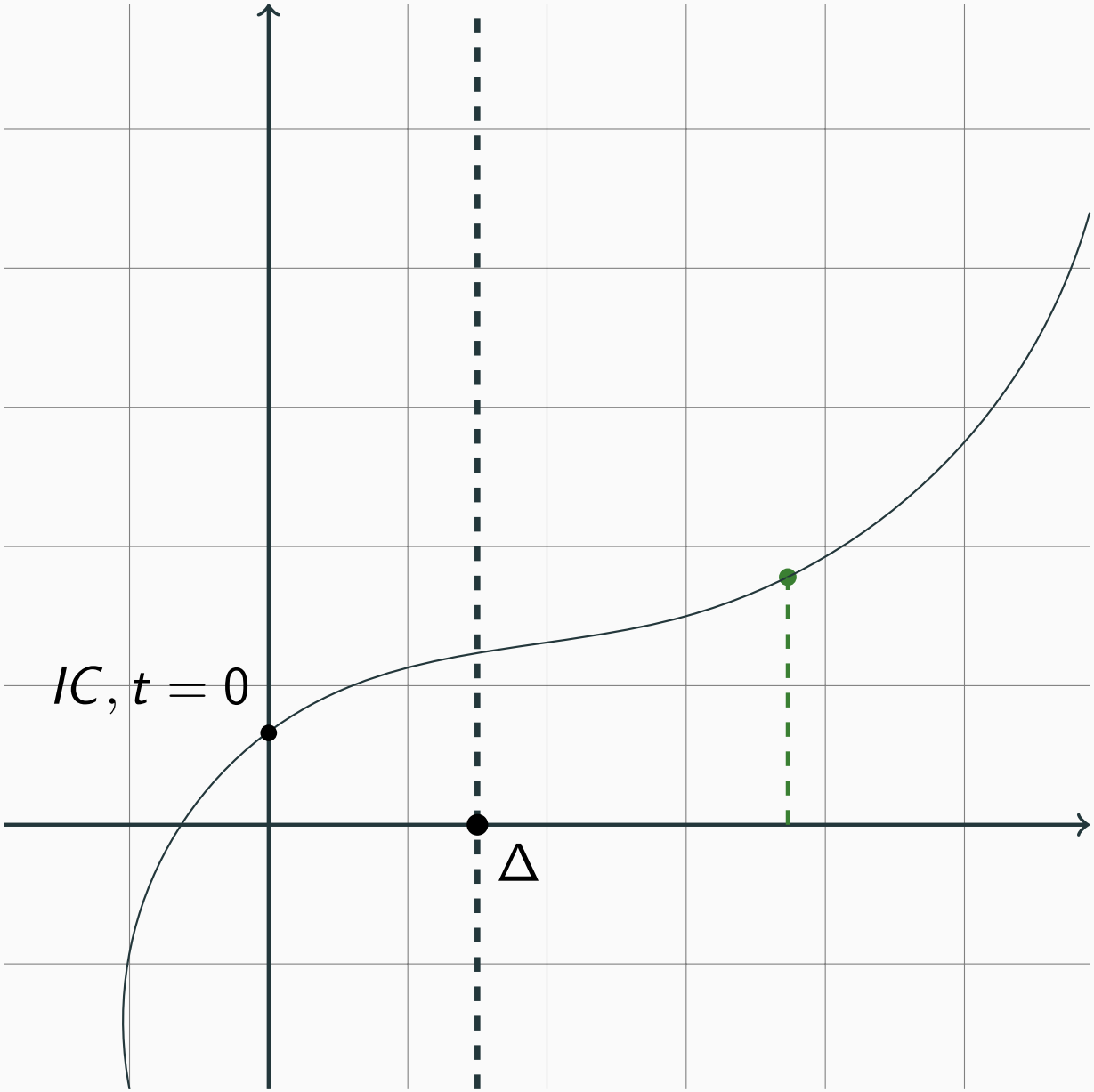
- Non-computable in the general case, for $\mathbf{u}$ computable (Pour-El, Richards)
- Computable for $\mathbf{u}$ computable and unique solution (Collins, Graça, Ruohonen)

Now space: Why it is not easy to solve ODEs

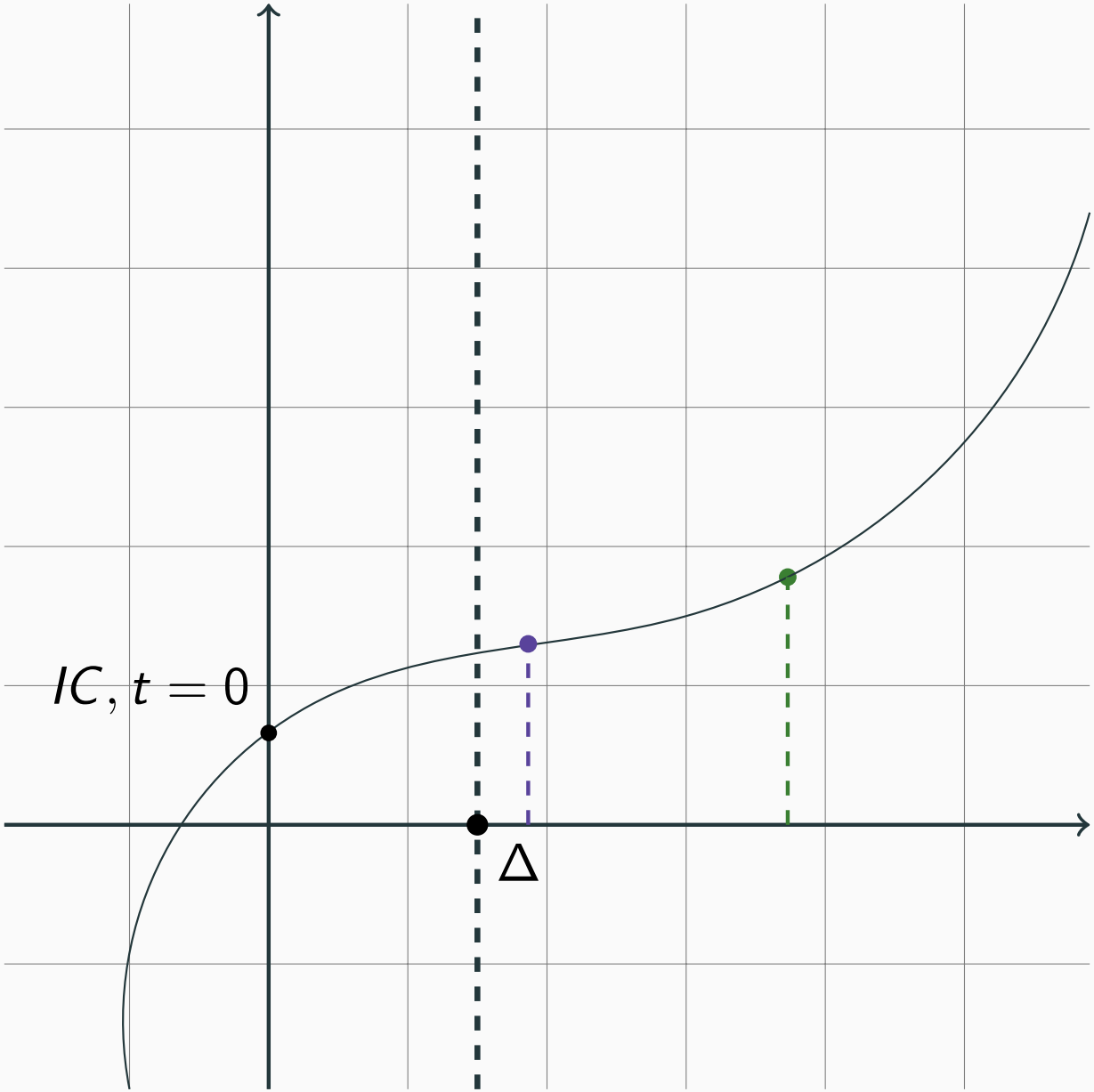
$$\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \textit{and} \quad \frac{\partial \mathbf{f}(t, \mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x}),$$

- Non-computable in the general case, for **u** computable (Pour-El, Richards)
- Computable for **u** computable and unique solution (Collins, Graça, Ruohonen)
- PSPACE-completeness on a bounded domain for **u** computable in polynomial-time (Kawamura, Ko)

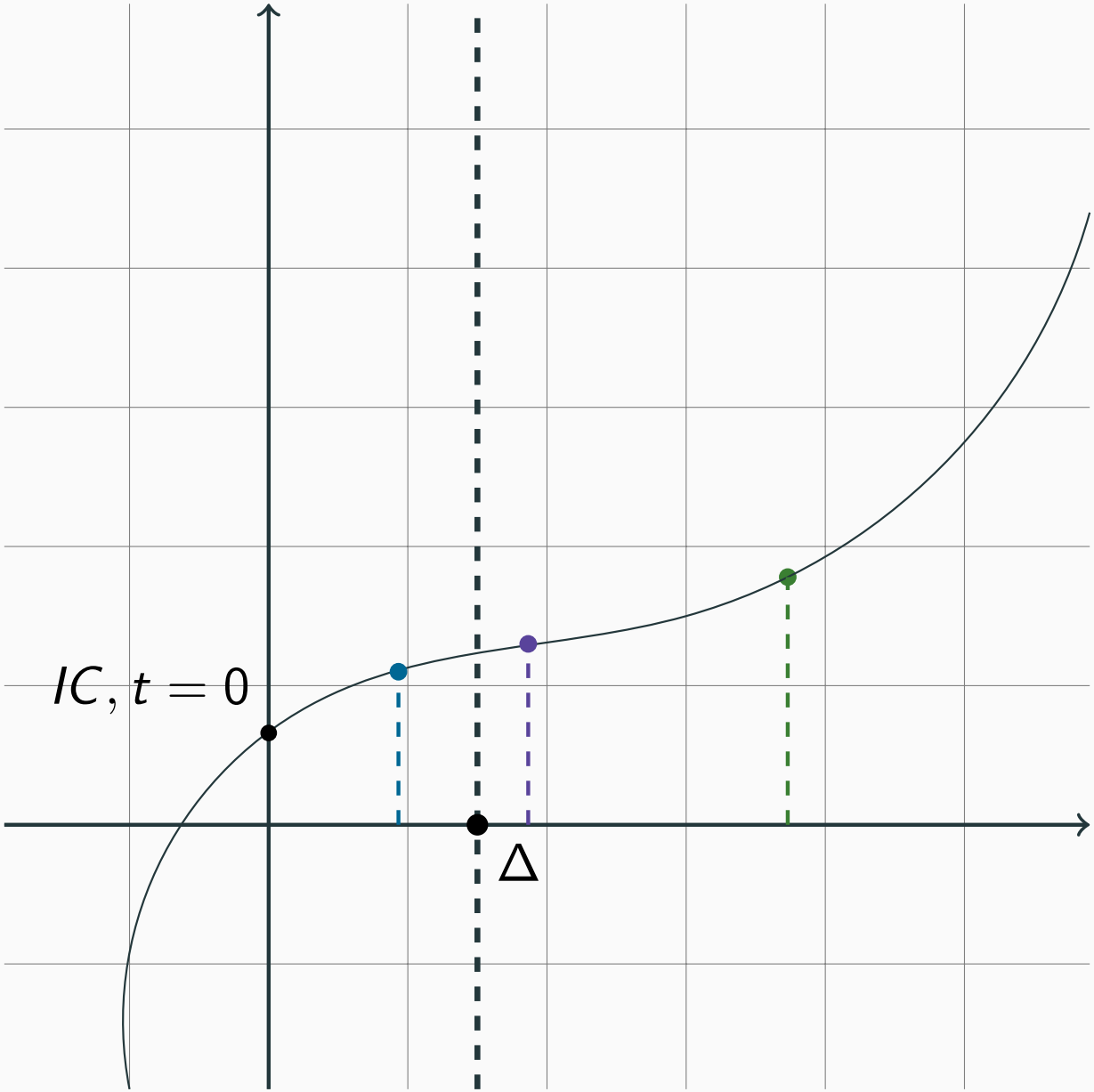
An unusual way of solving an ODE



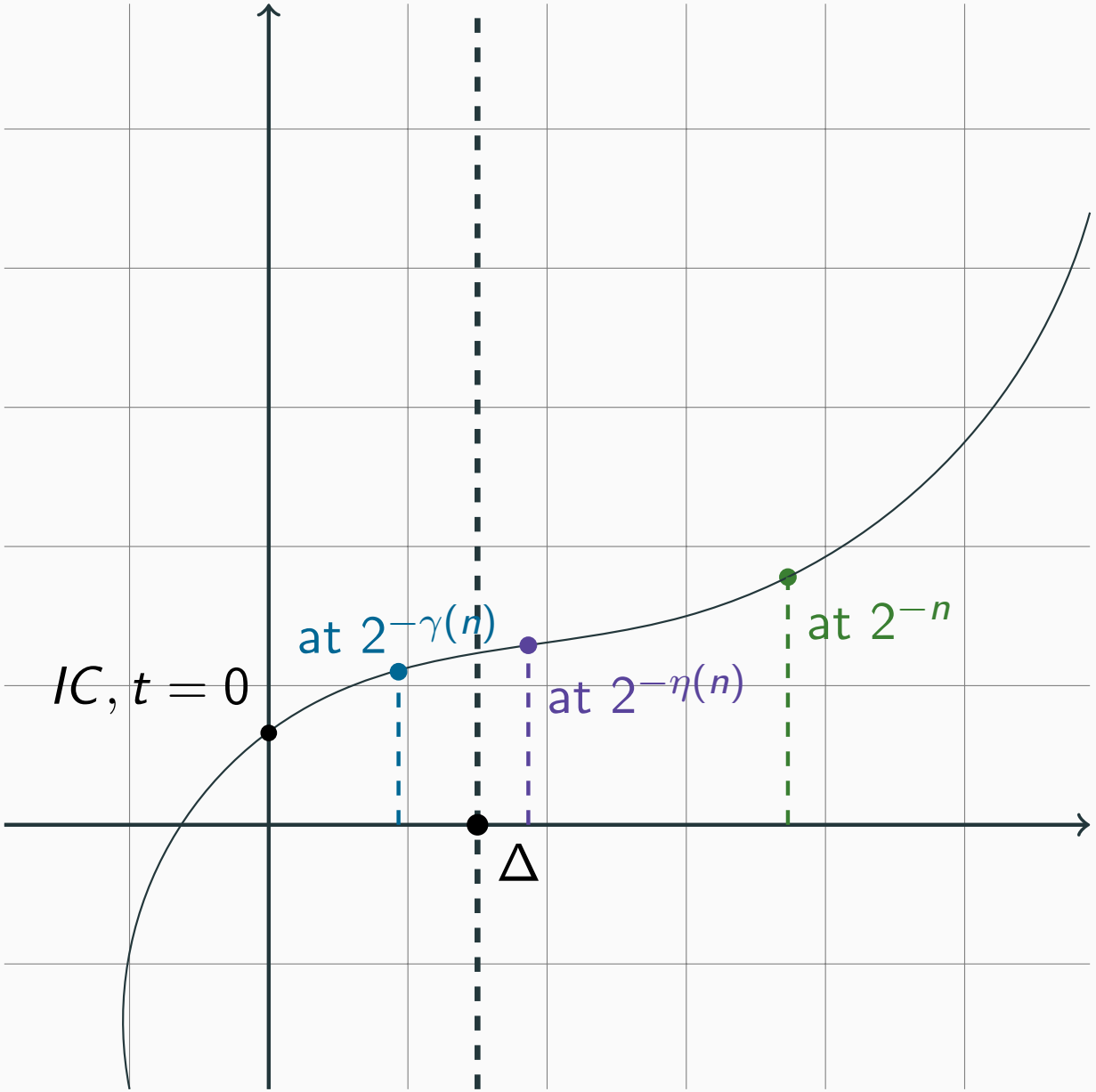
An unusual way of solving an ODE



An unusual way of solving an ODE



An unusual way of solving an ODE



Space=Precision

- Computation at $t = x$ at precision 2^{-n}, with initial condition $t = 0$
- Computation at $t = t_0 + \frac{x}{2}$ at precision $2^{-\eta(n)}$, with initial condition $t_0 = 0$ and $t_0 = \frac{x}{2}$
- Computation at $t = t_0 + \frac{x}{4}$ at precision $2^{-\gamma(n)}$, with initial condition $t_0 = 0$ and $t_0 = \frac{x}{4}$ and $t_0 = \frac{x}{2}$ and $t_0 = \frac{3x}{4}$

Space = Precision of the computation

Why computing with ODEs is interesting

Representations

- Differential analizers might use less energy
- Better representation of "real" systems: ODEs in mechanic, biology, chemistry...

Biochemistry: context

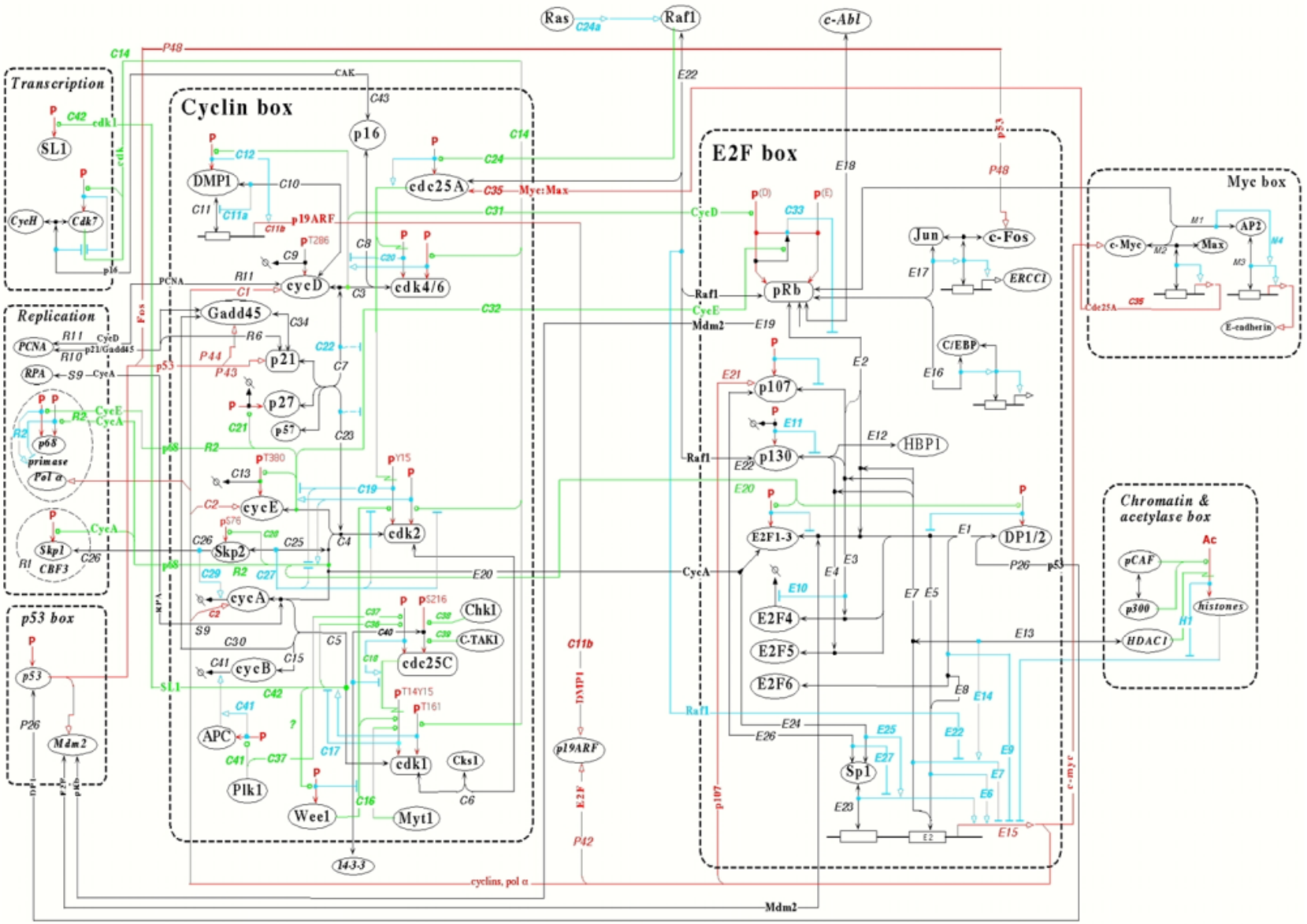
Your own cells do computation: treating signal, regulation of the metabolism

- Division
- Differentiation
- Migration

A major part of those are *analogue computations* with proteins

Once again, we live in an analogue world

Biochemistry: Kohn's map of the mammalian cell cycle



Biochemistry: into a GPAC

Assuming at most binary reactions with mass action law kinetics:

Theorem (Fages, Le Guludec, Bournez, Pouly)

- *Any polynomial ODE can be encoded in a chemically feasible positive system of double dimension.*
- *I.e. The systems of elementary biochemical reactions on finite universes of molecules are Turing-complete in differential semantics.*

Biochemistry: into a GPAC

Assuming at most binary reactions with mass action law kinetics:

Theorem (Fages, Le Guludec, Bournez, Pouly)

- *Any polynomial ODE can be encoded in a chemically feasible positive system of double dimension.*
- *I.e. The systems of elementary biochemical reactions on finite universes of molecules are Turing-complete in differential semantics.*

Furthermore: If the initial GPAC has a polynomial computational *complexity*, then the new system will also have a polynomial *complexity*.

So?

- Can you compute faster with GPACs than with TMs?

So?

- Can you compute faster with GPACs than with TMs? YES

So?

- Can you compute faster with GPACs than with TMs? YES
- Can you compute faster with Differential Analizer than with computers?

So?

- Can you compute faster with GPACs than with TMs? YES
- Can you compute faster with Differential Analizer than with computers? MAYBE

So?

- Can you compute faster with GPACs than with TMs? YES
- Can you compute faster with Differential Analizer than with computers? MAYBE
- Are all ODEs representable by a GPAC?

So?

- Can you compute faster with GPACs than with TMs? YES
- Can you compute faster with Differential Analizer than with computers? MAYBE
- Are all ODEs representable by a GPAC? YES IF POLYNOMIAL

So?

- Can you compute faster with GPACs than with TMs? YES
- Can you compute faster with Differential Analizer than with computers? MAYBE
- Are all ODEs representable by a GPAC? YES IF POLYNOMIAL
- Are all (poly) ODEs corresponding to a Differential Analizer?

So?

- Can you compute faster with GPACs than with TMs? YES
- Can you compute faster with Differential Analizer than with computers? MAYBE
- Are all ODEs representable by a GPAC? YES IF POLYNOMIAL
- Are all (poly) ODEs corresponding to a Differential Analizer? DON'T KNOW

Conclusion

Conclusion

- We saw and define analogue models of computation
- We saw a mathematical description of those models
- We linked the notion of time and memory costs for TMs with that framework