The Shannon machine

(Colloquium *"The Legacy of Claude Shannon"*)

Daniel Graça[1,2,*]

[1]Universidade do Algarve, Portugal
[2]Instituto de Telecomunicações, Portugal

13 December 2016

[*] Daniel Graça was partially supported by *Fundação para a Ciência e a Tecnologia* and EU FEDER POCTI/POCI via Instituto de Telecomunicações through the FCT project UID/EEA/50008/2013.

Claude Shannon

Claude Shannon (1916–2001) is well-known for his:

Claude Shannon

Claude Shannon (1916–2001) is well-known for his:

- Foundational work on information theory

Claude Shannon

Claude Shannon (1916–2001) is well-known for his:

- Foundational work on information theory
- Results connecting digital circuits with Boolean algebra

Claude Shannon

Claude Shannon (1916–2001) is well-known for his:

- Foundational work on information theory
- Results connecting digital circuits with Boolean algebra

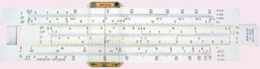
This talk discusses some less well-known results of Shannon about mathematical models for (analog) computation.

What is a computer?

Antikythera mechanism

What is a computer?

Antikythera mechanism

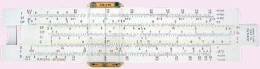


Slide Rule

What is a computer?

Antikythera mechanism

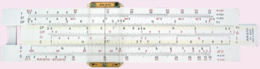
Differential Analyzer



Slide Rule

What is a computer?

Antikythera mechanism



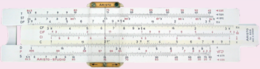
Slide Rule

Differential Analyzer

Laptop

What is a computer?

Antikythera mechanism



Slide Rule

Differential Analyzer

Laptop

Non-programmable

Programmable

Digital computers are the ultimate computing devices?

Digital computers are the ultimate computing devices?

Perhaps no!

- For example it is believed that quantum computers can efficiently solve problems which are too hard to solve with digital computers.

Digital computers are the ultimate computing devices?

Perhaps no!

- For example it is believed that quantum computers can efficiently solve problems which are too hard to solve with digital computers.

Question

Can we theoretically conceive other devices which might have more computational power than digital computers?

Digital computers are the ultimate computing devices?

Perhaps no!

- For example it is believed that quantum computers can efficiently solve problems which are too hard to solve with digital computers.

Question

Can we theoretically conceive other devices which might have more computational power than digital computers?

- If we are not yet able to build those devices (because of technological limitations, etc.), this question can only be answered with mathematical models.

Digital computers are the ultimate computing devices?

Perhaps no!

- For example it is believed that quantum computers can efficiently solve problems which are too hard to solve with digital computers.

Question

Can we theoretically conceive other devices which might have more computational power than digital computers?

- If we are not yet able to build those devices (because of technological limitations, etc.), this question can only be answered with mathematical models.
- We have a mathematical model for digital computers – the *Turing machine*.

Digital computers are the ultimate computing devices?

Perhaps no!

- For example it is believed that quantum computers can efficiently solve problems which are too hard to solve with digital computers.

Question

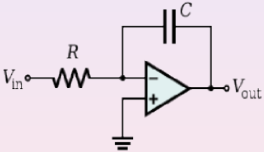
Can we theoretically conceive other devices which might have more computational power than digital computers?

- If we are not yet able to build those devices (because of technological limitations, etc.), this question can only be answered with mathematical models.
- We have a mathematical model for digital computers – the *Turing machine*.
- But what about for other computing devices?

- In particular, from a theoretical point of view, how do analog computers (differential analyzers) compare in terms of computational power with digital computers (Turing machines)?

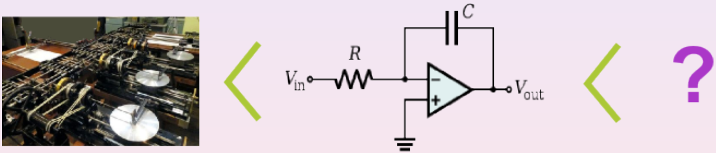
- In particular, from a theoretical point of view, how do analog computers (differential analyzers) compare in terms of computational power with digital computers (Turing machines)?
- For example, is it conceivable that one can implement a super analog computer, using some radically new technology not yet developed?

$<$



$<$?

- In particular, from a theoretical point of view, how do analog computers (differential analyzers) compare in terms of computational power with digital computers (Turing machines)?
- For example, is it conceivable that one can implement a super analog computer, using some radically new technology not yet developed?



- To answer that question we need a mathematical model for analog computers!

The Shannon machine (1941)

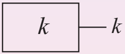
- Shannon worked as an operator on the (mechanical) differential analyzer of Vannevar Bush (better known for heading the U.S. OSRD during World War II, through which almost all wartime military R&D was carried out) to earn some money.

The Shannon machine (1941)

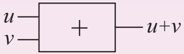
- Shannon worked as an operator on the (mechanical) differential analyzer of Vannevar Bush (better known for heading the U.S. OSRD during World War II, through which almost all wartime military R&D was carried out) to earn some money.
- In 1941 Claude Shannon introduced his *General Purpose Analog Computer (GPAC)* as a mathematical nodel for diferential analyzers.

The Shannon machine (1941)

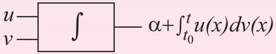
- Shannon worked as an operator on the (mechanical) differential analyzer of Vannevar Bush (better known for heading the U.S. OSRD during World War II, through which almost all wartime military R&D was carried out) to earn some money.
- In 1941 Claude Shannon introduced his *General Purpose Analog Computer (GPAC)* as a mathematical nodel for diferential analyzers.
- It consists of circuits built with four types of basic units:



A constant unit associated to the real value k



An adder unit



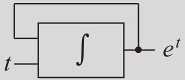
An integrator unit

A multiplier unit

Examples

Example

Compute $y(x) = e^x$ with a GPAC



$$\begin{cases} y' = y \\ y(0) = 1 \end{cases}$$

Theorem (Shannon, modern version)

A function is generated by a GPAC if and only if it is the solution of some (vectorial) polynomial initial-value problem

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

Theorem (Shannon, modern version)

A function is generated by a GPAC if and only if it is the solution of some (vectorial) polynomial initial-value problem

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

The class of solutions of polynomial initial-value problems (called PIVP functions) has some nice properties:

Theorem (Shannon, modern version)

A function is generated by a GPAC if and only if it is the solution of some (vectorial) polynomial initial-value problem

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

The class of solutions of polynomial initial-value problems (called PIVP functions) has some nice properties:

- It is closed under arithmetic operations $(+, -, \times, /)$ and composition

Theorem (Shannon, modern version)

A function is generated by a GPAC if and only if it is the solution of some (vectorial) polynomial initial-value problem

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

The class of solutions of polynomial initial-value problems (called PIVP functions) has some nice properties:

- It is closed under arithmetic operations $(+, -, \times, /)$ and composition
- All the elementary functions of Analysis are PIVP

Theorem (Shannon, modern version)

A function is generated by a GPAC if and only if it is the solution of some (vectorial) polynomial initial-value problem

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

The class of solutions of polynomial initial-value problems (called PIVP functions) has some nice properties:

- It is closed under arithmetic operations $(+, -, \times, /)$ and composition
- All the elementary functions of Analysis are PIVP
- Solutions of initial-value problems defined with PIVPs are themselves PIVPs (e.g. the solution of $y_1' = \sin(y_2) + y_1, y_2' = \cos(y_1) + 3$, $y_1(0) = y_2(0) = 1$ is constituted by PIVP functions)

Theorem (Shannon, modern version)

A function is generated by a GPAC if and only if it is the solution of some (vectorial) polynomial initial-value problem

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

The class of solutions of polynomial initial-value problems (called PIVP functions) has some nice properties:

- It is closed under arithmetic operations $(+, -, \times, /)$ and composition
- All the elementary functions of Analysis are PIVP
- Solutions of initial-value problems defined with PIVPs are themselves PIVPs (e.g. the solution of $y_1' = \sin(y_2) + y_1, y_2' = \cos(y_1) + 3$, $y_1(0) = y_2(0) = 1$ is constituted by PIVP functions)
- Almost all of Classical Physics can be modeled with PIVP functions

In summary

- Shannon introduced a mathematical model (GPAC/PIVPs) for analog computers.
- The PIVP model has nice mathematical properties and can simulate/capture a large class of physical systems.

Main takeaway of this talk

Shannon did for analog computers what Turing did for digital computers.

Some recent results

More recently there has been some research comparing the theoretical computing power of PIVP functions vs. Turing machines.

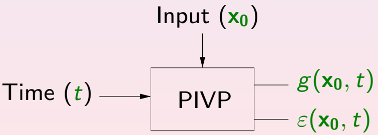
Some recent results

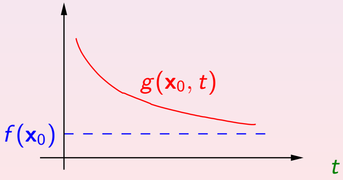
More recently there has been some research comparing the theoretical computing power of PIVP functions vs. Turing machines.

Theorem (Bournez, Campagnolo, Graça, Hainry)

PIVPs and Turing machines are equivalent from a computability perspective.

In the previous result it is assumed that we allow PIVPs to use some time for computation before giving the output.





Even if at a computability level we do not get more from PIVP functions, is it conceivable that we can theoretically compute the solution of some problems *faster*, like for quantum computing?

Even if at a computability level we do not get more from PIVP functions, is it conceivable that we can theoretically compute the solution of some problems *faster*, like for quantum computing?

Answer: No!

Even if at a computability level we do not get more from PIVP functions, is it conceivable that we can theoretically compute the solution of some problems *faster*, like for quantum computing?

Answer: No!

Theorem (Bournez, Graça, Pouly)

PIVPs and Turing machines (computable analysis) are (polynomially) equivalent from a computational complexity perspective.

Even if at a computability level we do not get more from PIVP functions, is it conceivable that we can theoretically compute the solution of some problems *faster*, like for quantum computing?

Answer: No!

Theorem (Bournez, Graça, Pouly)

PIVPs and Turing machines (computable analysis) are (polynomially) equivalent from a computational complexity perspective.

As an (unexpected) side effect, the previous result allows us to define the class P of functions computable in polynomial time by using polynomial IVPs and by (polynomially) bounding the length of the solution curve.

Thank you!

Definition (Discrete recognizability)

A language $\mathcal{L} \subseteq \Gamma^*$ is called *analog recognizable* if there exists a vector q of polynomials with two variables and a vector p of polynomials with d variables, both with polynomial-time computable coefficients, and a polynomial $\Omega : \mathbb{R}_0^+ \to \mathbb{R}_0^+$, such that for all $w \in \Gamma^*$ there is a (unique) $y : \mathbb{R}_0^+ \to \mathbb{R}^d$ such that for all $t \in \mathbb{R}_0^+$:

- $y(0) = q(\psi(w))$ and $y'(t) = p(y(t))$ ▶ y satisfies a differential equation
- if $|y_1(t)| \geqslant 1$ then $|y_1(u)| \geqslant 1$ for all $u \geqslant t$ ▶ the decision is stable
- if $w \in \mathcal{L}$ (resp. $\notin \mathcal{L}$) and $len_y(0, t) \geqslant \Omega(|w|)$ then $y_1(t) \geqslant 1$ (resp. $\leqslant -1$) ▶ decision
- $len_y(0, t) \geqslant t$ ▶ technical condition

Theorem (An implicit characterization of P - Bournez, Graça, Pouly)

A decision problem (language) $\mathcal{L}$ *belongs to the class P if and only if it is analog recognizable.*