Minds and Machines (Lecture 4): Computationalism

Prof. Ioannis Votsis

Philosophy Faculty
ioannis.votsis@nulondon.ac.uk
www.votsis.org

Introduction

- We have thus far considered the question whether computers are intelligent and even whether they can think.

are like

- One can also reverse this type of question and ask whether to reason intelligently or to think is to compute.

are like

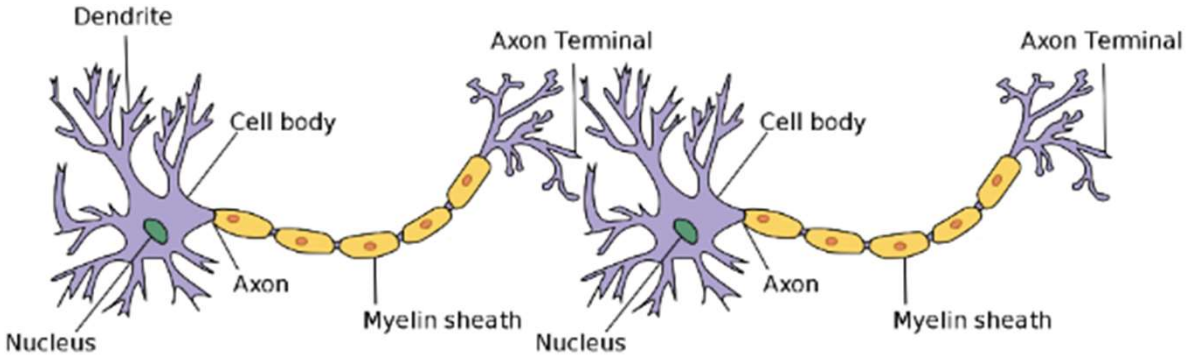
The brain as an ideal for computing

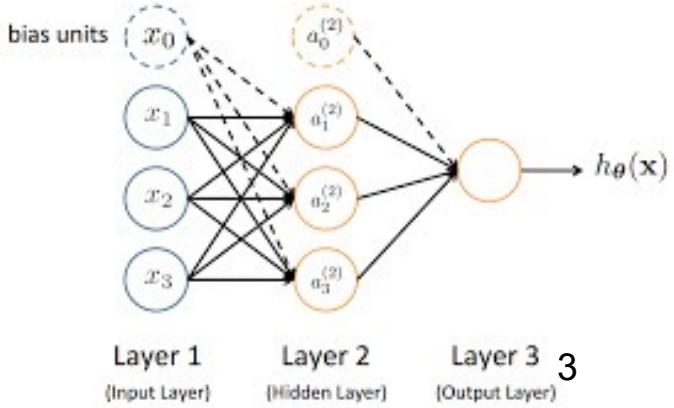
- Brains are composed of (roughly 86 billion) neurons that are interconnected and communicate via synapses.

- McCulloch and Pitts (1943): The brain can serve as an ideal for computer design; they devised the perceptron concept.

NB: First ANN implemented by Farley & Clark at MIT in 1950s.

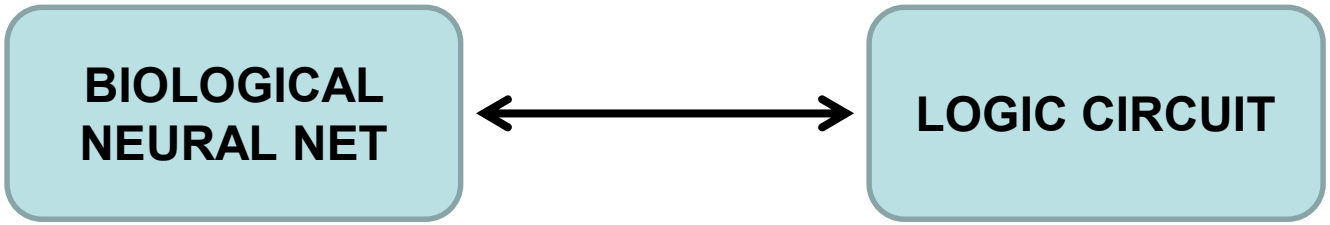
- ANNs mimic biological neural nets by storing and processing data in interconnected layers of nodes.





Computing as an ideal for the brain

- Conversely, some scholars started thinking of computers as models for what is going on in the mind/brain.
- This view also appears in McCulloch and Pitts (1943), where it's suggested that the brain may work like a Turning machine.



- It is now known as 'computationalism' or 'the computational theory of mind'.

Turing machines

- Informally, an algorithm is a finite sequence of instructions that transforms an input into an output.

- Turing (1936) sets out to formalise algorithms and in doing so comes up w/the abstract device we call ‘Turing machine’:

  * **memory**: infinitely long tape divided up into discrete cells
  * **states**: each cell empty or contains symbol ‘0’ or ‘1’
  * **head**: contains a finite # of internal states and moves sideways, one cell at a time, reading/writing symbols
  * **table**: contains a finite # of instructions that command the head based on its internal state + contents of current cell.

- Note that the table is effectively specifying an algorithm.

Different implementations

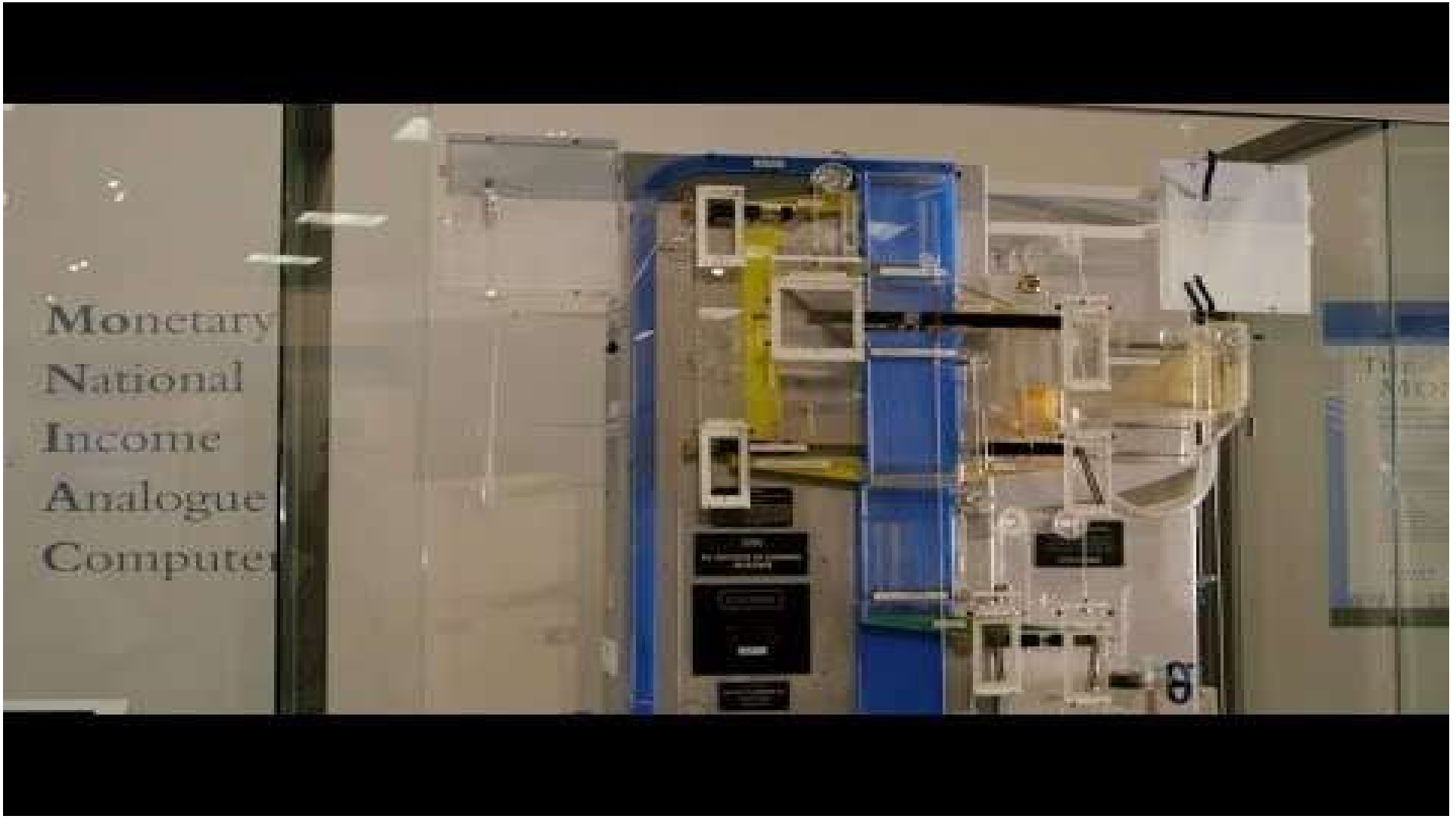
- In the same paper, Turing also proved the existence of what is now known as a Universal Turing Machine.

- In effect, such a machine is able to mimic any other Turing machine. All PCs nowadays approximate a UTM.

- **Substrate independence**: The substrate on which a Turing machine is implemented may vary considerably.

  * silicon-based
  * vacuum-tube
  * pulley-and-lever
  * hydraulic
  * …

Moniac



Computationalism: A Family of Views

The classical computational theory of mind

- Computationalists view the mind (/brain) as a computational system, i.e. cognitive processes are thought to compute.

- Beyond this claim, the classical computationalists claim that the mind computes in a manner akin to a Turing machine.

  * both employ operations to transform inputs into outputs
  * both rely on memory to do so

- Having said this, there are clear differences between them:

  * **inputs and outputs**: symbols vs. electrical impulses
  * **memory capacity**: infinite vs. finite
  * **type of operation**: serial vs. parallel

Classical: Machine functionalism

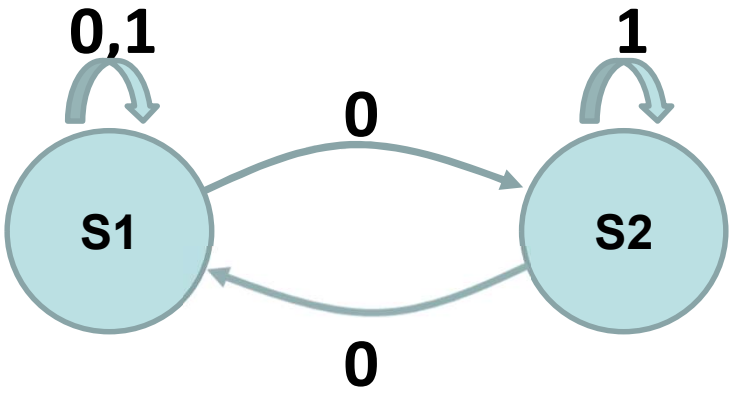
- Recall that functionalism is the view that mental states are states that play the relevant role in a system.

- **Machine functionalism**: The mind (/brain) is a functionally organised system akin to Turning machines but stochastic.

Example: State Transition Table

	Input	
Current State	**0**	**1**
S1	S1, S2	S1
S2	S1	S2

State Transition Diagram



NB: Machine functionalism was put forth by Hilary Putnam.

Classical: The representational theory of mind

- On this view, the mind (/brain) is a system that manipulates mental representations like Turing machines do symbols.

- What are mental representations? They are mental states that stand for things and which have semantic properties.

  Example: The belief that COVID is dangerous.

- The view was developed by J. Fodor (1975; 2008) who also posited the (closely tied) Language of Thought hypothesis.

- The same operations are meant to happen in the mind as in computers, e.g. logical inferences involving conjunction.

Non-classical: Connectionism

- The mind (/brain) computes in a manner akin to ANN computations (McClelland, Rumelhart and Hinton 1986).
- Though trivial-sounding, several advances in ANN theory have been made independently of neuroscientific work.
- Connectionists de-emphasise rule following and emphasise parallel computation.

NB: Neural nets are built and run on digital computers and digital computers can be built and run on analog neural nets.

Pancomputationalism

A triviality argument against CTM

- Searle (1990) famously denounced computationalism as being too broad and therefore collapsing to triviality:

  "Thus for example the wall behind my back is right now implementing the Wordstar [word-processing] program, because there is some pattern of molecule movements which is isomorphic with the formal structure of Wordstar. But if the wall is implementing Wordstar then if it is a big enough wall it is implementing any program, including any program implemented in the brain (p. 27).

- The argument invites us to imagine the absurdity of such a view, which goes by the name of 'pancomputationalism'.

Physical vs. abstract computation

- Searle's objection brings up interesting questions about the nature of computation.

- We already know about *abstract computation*, namely it is the computation performed by a Turing machine.

- What exactly is *physical computation*?

- Theories that seek to answer this and related questions are known as theories of 'implementation':

  * Under what conditions can we say that:
    - a physical process computes?
    - two physical computations are the same?

The challenge of pancomputationalism

- Implementation theories need to address the challenge of pancomputationalism. The view takes different forms.

- **Unlimited** (clearly false): Every physical system can produce every computation.

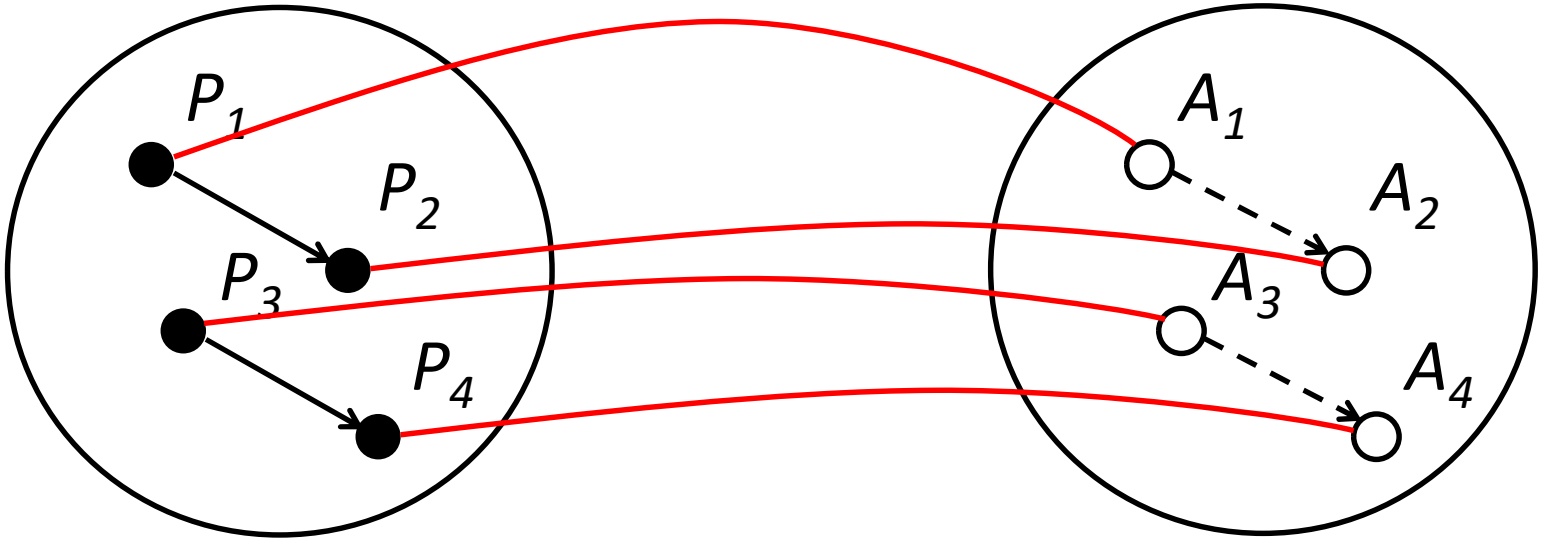
- **Unlimited** (slightly weaker): Every sufficiently complex physical system can produce every/many computations.

  NB: The slightly weaker version is the one that Putnam (1988) and Searle (1990) have in mind.

- **Limited**: Every physical system performs at least one computation.

Mapping: A general requirement

- All implementation theories hold that a mapping is *required* between a physical system & an abstract computing system.



Physical System

Abstract System

Theories of implementation

- **Simple/weak mapping account**:
  A physical system *computes*^{P} iff it's describable as a sequence of states isomorphic to an abstract machine's sequence.

  NB: This view goes back to Putnam (1960) and is clearly too liberal as illustrated by Searle's objection.

- **Modal accounts**: A physical system *computes*^{P} iff it meets the mapping requirement + it's modally constrained.

  Modal constraints: causal, counterfactual and dispositional.

  On the causal version (Chalmers 2011), for example, output states must be the effects of input states which caused them.

Theories of implementation (2)

- **Semantic accounts**: A physical system *computes*^{P} iff it meets the mapping requirement + its states are content-bearing.

  NB: This is captured in Fodor's memorable pronouncement: "no computation without representation" (1975: 34)

- **Mechanistic accounts**: A physical system *computes*^{P} iff it meets the mapping requirement + it qualifies as a certain FM.

  Functional Mechanism (FM): Made up of parts whose organisation endows the system w/computational capacities.

  NB: One prominent advocate of this view is Piccinini (2015).

The End