BIO LOGIC: Biological Computation

Kay Kirkpatrick, UIUC

2018
Kay Kirkpatrick, UIUC

Urbana, Miami and Mascouten lands, October 2018

Movie still from the Yonath lab; next slide from Cambridge U.
Turing 1948: Intelligent Machinery

3. Varieties of machinery

It will not be possible to discuss possible means of producing such machinery without introducing a number of technical terms to describe the kinds of existent machinery.

‘Discrete’ and ‘Continuous’ machinery. We may say a machine is discrete if it is capable of existing in a finite number of states (configurations) at any one time, and if its behaviour is determined by the state in which it is. The behaviour of a machine can thus be described by the sequence of states it passes through, and this description can be used to determine the sequence of actions performed by the machine.

‘Active’ and ‘Passive’ machinery. Active machinery are those that can change their state in response to external stimuli, whereas passive machinery are those that cannot.

Examples

- A Bulldozer is an example of an active machine, as it can move in response to external stimuli.
- A Telephone is an example of a passive machine, as it can only respond to input from another machine.
- A Brunsviga is a type of calculator that can perform calculations.
- A Differential Analyzer is a machine that can perform complex mathematical calculations.

We shall mainly be concerned with discrete controlling machinery, because mentioned, brains very nearly fall into this class, and there seems reason to believe that they could have been made to fall generally into a state where they change in their essential properties. However, the property of being semi-

* Editor’s note: The Brunsviga was a popular desk calculating machine.

Courtesy Copeland, Complete Turing
‘Controlling’ and ‘Active’ machinery. Machinery may be described as ‘controlling’ if it only deals with information. In practice this condition is much the same as saying that the magnitude of the machine’s effects may be as small as we please, so long as we do not introduce confusion through Brownian movement etc. ‘Active’ machinery is intended to produce some definite physical effect.

**Examples**

- A Bulldozer is **Continuous Active**
- A Telephone is **Continuous Controlling**
- A Brunsviga is **Discrete Controlling**
- A Brain is probably **Continuous Controlling**, but is very similar to much discrete machinery
- The ENIAC, ACE etc. is **Discrete Controlling**
- A Differential Analyser is **Continuous Controlling**
The brain processes information AND is active:

- ions, neurotransmitters, neuromodulators,
- hormones, remodeling synapses, movement, fields.
His 1950 Imitation Game relies on this mistake courtesy wikipedia

Turing Test involves just info input & info output.
His 1950 Imitation Game relies on this mistake
courtesy wikipedia

Turing Test involves just info input & info output.

Same with the Chinese Room Argument objection.
We need new kinds of abstract machines

Active Information Machine: processes information AND produces significant physical effects.

E.g., brains, other organs, robots, prosthetics...
We need new kinds of abstract machines

Active Information Machine: processes information AND produces significant physical effects.

E.g., brains, other organs, robots, prosthetics...

Computers? (require user)

Artificial neural networks?
Statistical mechanics might help us figure out the brain
Statistical mechanics might help us figure out the brain

microscopic first principles $\leadsto$ zoom out $\leadsto$ Macroscopic states

Courtesy Greg L and Digital Vision/Getty Images
1. Proper microscopic unit of computation?
2. Unit properties vis-à-vis info & computation.
3. Consequences of this new approach.
What are fundamental computational units in the brain?

Usually thought to be the neuron, modeled by a scalar. (McCulloch & Pitts, von Neumann, etc.)

But neurons are mesosopic and sophisticated.
What are fundamental computational units in the brain?

Usually thought to be the neuron, modeled by a scalar. (McCulloch & Pitts, von Neumann, etc.)

But neurons are mesoscopic and sophisticated.

Claim: ribosomes & proteins are better fundamental computational units.
Ribosomes synthesize proteins from mRNA genes

Courtesy Jay Swan
How to model a ribosome mathematically? (Take 1)

Via Molecular Dynamics (too detailed). But movie: https://www.youtube.com/watch?v=Jml8CFBWcDs

As a particle in TASEP (too reductive). But theory.
How to model a ribosome mathematically? (Take 1)

Via Molecular Dynamics (too detailed). But movie: https://www.youtube.com/watch?v=Jml8CFBWcDs

As a particle in TASEP (too reductive). But theory.

For the moment here: as an information channel.
The ribosome as an information channel

proto-code DNA \rightarrow \text{transcription, regulation} \rightarrow \text{mRNA } X^* \\
\downarrow \\
\text{channel: Ribosome} \\
\downarrow \\
folded protein \tilde{Y} \leftarrow \text{post-translational modifications} \rightarrow \text{polypeptide } Y^*
The ribosome operates under its Shannon capacity

**Theorem (K.-Osuagwu 2018):** Shannon capacity of the ribosome is at least 266 bits per second, or 44 codons per sec. Cf. observed rates average 17 codons per sec, max about 20.

Proof is a straightforward calculation with uniform distribution.

This helps explain why ribosomes are so accurate: $\sim 99.99\%$. 
How to model a ribosome, Take 2

As a Universal Turing Machine, in the cell (Caetano-Anollés & Caetano-Anollés 2015).

As a finite automaton (still reductive).
As a Universal Turing Machine, in the cell (Caetano-Anollés & Caetano-Anollés 2015).

As a finite automaton (still reductive).

Still purely computational, not active.
Notation to get beyond pure computation

Input mRNA tape $X^*$ is string of codons in $\Sigma := \{A, C, G, U\}^3$

$\Sigma^* := \{\text{words with letters in } \Sigma, \text{ incl. empty word } \epsilon\}$. 
Notation to get beyond pure computation

Input mRNA tape $X^*$ is string of codons in $\Sigma := \{A, C, G, U\}^3$

$\Sigma^* := \{\text{words with letters in } \Sigma, \text{ incl. empty word } \epsilon\}$. 

Output polypeptide $Y^*$ is a string of **physical blocks** in

$\Delta^* := \{\text{strings of blocks in } \Delta := \{b_1, b_2, \ldots, b_{20}\}\}$. 
Notation to get beyond pure computation

Input mRNA tape $X^*$ is string of codons in $\Sigma := \{A, C, G, U\}^3$

$\Sigma^* := \{\text{words with letters in } \Sigma, \text{ incl. empty word } \epsilon\}.$

Output polypeptide $Y^*$ is a string of physical blocks in

$\Delta^* := \{\text{strings of blocks in } \Delta := \{b_1, b_2, \ldots, b_{20}\}\}.$

$Y^*$ contains info and is also a physico-chemical structure.
Turing’s definition of a 2-tape automatic machine

Consists of six parts: \( M := \langle Q, \Sigma, \Delta, \delta, q_0, F \rangle \)

- State space \( Q \)
- \( \Sigma \) input alphabet, and input program \( X^* \in \Sigma^* \)
- \( \Delta \) output symbols, and output tape \( Y^* \)
Turing’s definition of a 2-tape automatic machine

Consists of six parts: \( M := \langle Q, \Sigma, \Delta, \delta, q_0, F \rangle \)

- State space \( Q \)
- \( \Sigma \) input alphabet, and input program \( X^* \in \Sigma^* \)
- \( \Delta \) output symbols, and output tape \( Y^* \)
- Initial state \( q_0 \)
- Final/accepting states \( F \subset Q \)
- Transition function (rules about updating state/tape) \( \delta : (Q \setminus F) \times \Sigma \to Q \times \Sigma \times \Delta \times \{L, R\} \)
Our definition of an automatic bio-chemical machine

An \textbf{abc-machine} has 8: \( M_{T,A} = \langle Q, \Sigma, \Delta, \delta, q_0, F, T, A \rangle \)

- \( Q, \Sigma, q_0, F \) as before; input program \( X^* \in \Sigma^* \)
- State space \( Q \), with initial state \( q_0 \), final states \( F \subset Q \)
- Transitions \( \delta : (Q \setminus F) \times \Sigma \to Q \times \Sigma \times \Delta \times \{L, R\} \)
Our definition of an automatic bio-chemical machine

An **abc-machine** has 8: \( M_{T,A} = \langle Q, \Sigma, \Delta, \delta, q_0, F, T, A \rangle \)

- \( Q, \Sigma, q_0, F \) as before; input program \( X^* \in \Sigma^* \)
- State space \( Q \), with initial state \( q_0 \), final states \( F \subset Q \)
- Transitions \( \delta : (Q \setminus F) \times \Sigma \to Q \times \Sigma \times \Delta \times \{L, R\} \)

- \( \Delta \) output **blocks**, and \( Y^* \) is **block-string** output
- \( T \) a random stopping time, when program \( X^* \) **halts**
- \( A \) an auxiliary machine, optimizing a bio-chemical energy functional, e.g., protein-folding
Example: ribosome as an abc-machine $M_{T,A}$

$T = T(X^*, \omega)$ is random degradation time of $X^*$ by hydrolysis

Auxiliary $A$ is a protein-folder $Y^* \rightsquigarrow \tilde{Y} = \text{argmin } H(Y^*)$
The ribosome is more than just a Turing machine

**Observation (K.-Osuagwu 2018):** The ribosome $M_{T,A}$ is equivalent to a deterministic two-tape Turing machine with two oracles: protein-folding machine $A$ and stopping time machine $B := \{t < T\}$, and folded output $\tilde{Y}$. 
The ribosome is more than just a Turing machine

**Observation (K.-Osuagwu 2018):** The ribosome $M_{T,A}$ is equivalent to a deterministic two-tape Turing machine with two oracles: protein-folding machine $A$ and stopping time machine $B := \{ t < T \}$, and folded output $\tilde{Y}$.

Remarks: $\tilde{Y}$ is active, like a bulldozer. Can process info.

The ribosome $M_{T,A}$ is both active and information-processing.

(Actually, pieces of $X^*$ can act like bulldozers too: e.g., ATP.)
An abc-machine gets around the Halting Problem:

Any particular input eventually stops one way or another by quantum randomness and hydrolysis.
Consequences for the brain

Individual computational units are more than TMs.
Consequences for the brain

Individual computational units are more than TMs.

Conjecture: brain is more than a TM. Embodiment is important.
When will we be able to model the whole brain?

The brain has $\sim 10^{11}$ cells (neurons and glia).

Each cell has $\sim 10^6$ ribosomes and $\sim 10^7$ proteins.
When will we be able to model the whole brain?

The brain has $\sim 10^{11}$ cells (neurons and glia).

Each cell has $\sim 10^6$ ribosomes and $\sim 10^7$ proteins.

Total: $\sim 10^{18}$ units, each $\sim 100$ binary operations per second.

Need zettascale ($10^{21}$ flops) or yottascale ($10^{24}$) computing, expected around the years 2030 and 2040, respectively.
When will we be able to model the whole brain?

The brain has $\sim 10^{11}$ cells (neurons and glia).

Each cell has $\sim 10^6$ ribosomes and $\sim 10^7$ proteins.

Total: $\sim 10^{18}$ units, each $\sim 100$ binary operations per second.

Need zettascale ($10^{21}$ flops) or yottascale ($10^{24}$) computing, expected around the years 2030 and 2040, respectively.

Cf: von Neumann iPhone-scale; weather modeling zettascale.
Why are artificial “neural networks” successful?

Not because they’re like a brain, but because they’re like something else in biology...
Maybe artificial neural networks are successful...

Because they’re like biochemical networks in E. coli, which can basically do Stochastic Gradient Descent.
But artificial neural networks fail spectacularly

An adversarial example:

“panda”
57.7% confidence

+ $\epsilon$

= “gibbon”
99.3% confidence

Courtesy Open AI
Key takeaway: “Neural network” is a misnomer

Their neurones are way simpler than actual neurons.

Bioinformatics is more sophisticated than we thought. "Decoding" neural activity may be harder than we thought.

Biological information \(\gg\) abstract information (Keller, 2009)

More takeaways
More takeaways

Biological information $\gg$ abstract information (Keller, 2009)

Biological computation is more sophisticated than we thought.
More takeaways

Biological information $\gg$ abstract information (Keller, 2009)

Biological computation is more sophisticated than we thought.

“Decoding” neural activity may be harder than we thought.
What’s next: outside the Turing universality class

Abc-machine a semi-direct product of TMs & bulldozers?
What’s next: outside the Turing universality class

Abc-machine a semi-direct product of TMs & bulldozers?

Conjecture: biological computation is outside the Turing class. Siegelmann & Sontag ’95.

Church-Turing Theses: Mathematical CTT probably true; Conjecture: both Bold & Modest Physical CTT are false.
Thanks

NSF CAREER Award DMS-1254791, Simons Sabbatical Fellowship

faculty.math.illinois.edu/~kkirkpat/Kirkpatrick2018TTfaultyAssumption.pdf
What about DNA computing?
If protein folding is NP-complete (like its model self-avoiding random walk in 2D or 3D, Berger and Leighton 1998), then the ribosome’s auxiliary machine $A$ and $M_{T,A}$ are in class $\textbf{NP}$. Then $A$ can be approximated by a non-deterministic Turing Machine (NDTM), but can’t be simulated in polynomial time. Also, $T$ can be approximately simulated by NDTMs, but $T$ is actually random and non-computable. (Halting Problem.)
Complexity & Computability

If protein folding is NP-complete (like its model self-avoiding random walk in 2D or 3D, Berger and Leighton 1998), then the ribosome’s auxiliary machine $A$ and $M_{T,A}$ are in class $\textbf{NP}$.

Then $A$ can be approximated by a non-deterministic Turing Machine (NDTM), but can’t be simulated in polynomial time.

Also, $T$ can be approximately simulated by NDTMs, but $T$ is actually random and non-computable. (Halting Problem.)