Welcome to EE381V/CS395T Unconventional Computing

Spring 2023

David Soloveichik

T, Th 5:00-6:30pm
ECJ 1.308

Welcome to EE381V/CS395T aka “Things David Likes”

Spring 2023

David Soloveichik

T, Th 5:00-6:30pm

ECJ 1.308

About me

Dr. David Soloveichik
david.soloveichik@utexas.edu

[Solo-vey'-chick]

Originally from Ukraine

2002 BS in Computer Science (Harvard)
2008 PhD in Computation and Neural Systems (Caltech)
Spent time at UW and UCSF as postdoc

Faculty member at UT since 2015

I have a disability: stuttering

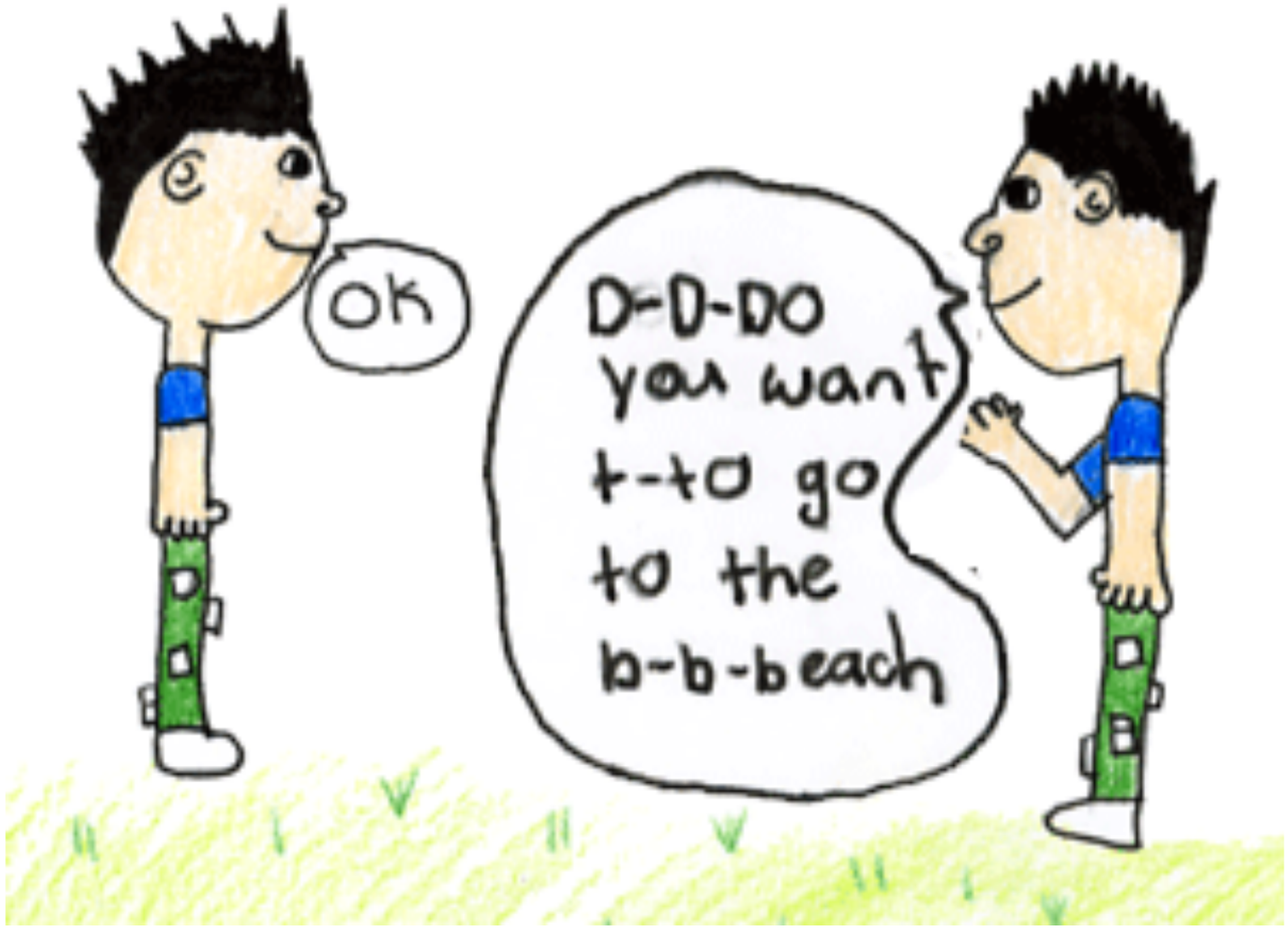


Image: cyh.com

My research

- Models of chemical kinetics and distributed computing
  We use formal models to discover the potential and limits of chemical information processing. Many of our results generalize beyond chemistry to distributed computing with computationally weak agents.
- Molecular programming: engineering smart molecules
  Using DNA "strand displacement cascades" we build molecular interactions for synthetic biology, nanotechnology, and bioengineering in our wet-lab. We use chemistry as a "programming language".
- Quantum and reversible computation
  For low energy computation and for quantum computing, computation must "look similar" both forward and backward in time. We study such computation and develop tricks to optimize it.

Teaching Assistant

Austin Luchsinger
amluchsinger@utexas.edu

Prerequisites

- Experience with proofs (e.g., discrete math, algorithms)

  This is a theory course: we will spend some time studying definitions, theorems and proofs

- Helpful: automata theory (Turing machines), logic circuits, probability, basic differential equations…

- **No physics, biology, or chemistry background needed**

<u>unconventional computation</u> inspires different **<u>models of computing</u>**

For the models of unconventional computing, we'll ask questions like:

1. How do we do interesting things in the model?

2. What are the limits to the computational ability of the model (something the model can't do at all)?

3. What is the appropriate notion of resources (Eg time: what can we do "quickly" in the model?)

Major Topics

1. Analog/real-valued computation **computation gets real**

2. Turing machines, cellular automata, counter machines, etc **almost anything can compute everything**

3. Reversible computing **computing without using energy**

4. Quantum computing ***@#? computing with negative probabilities**

Major Topics

1. Analog/real-valued computation **computation gets real**

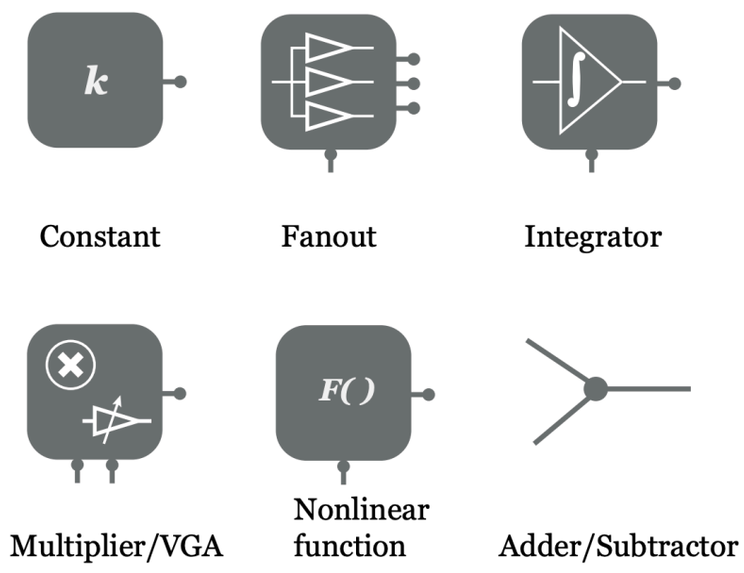
2. Turing machines, cellular automata, counter machines, etc **almost anything can compute everything**

3. Reversible computing **computing without using energy**

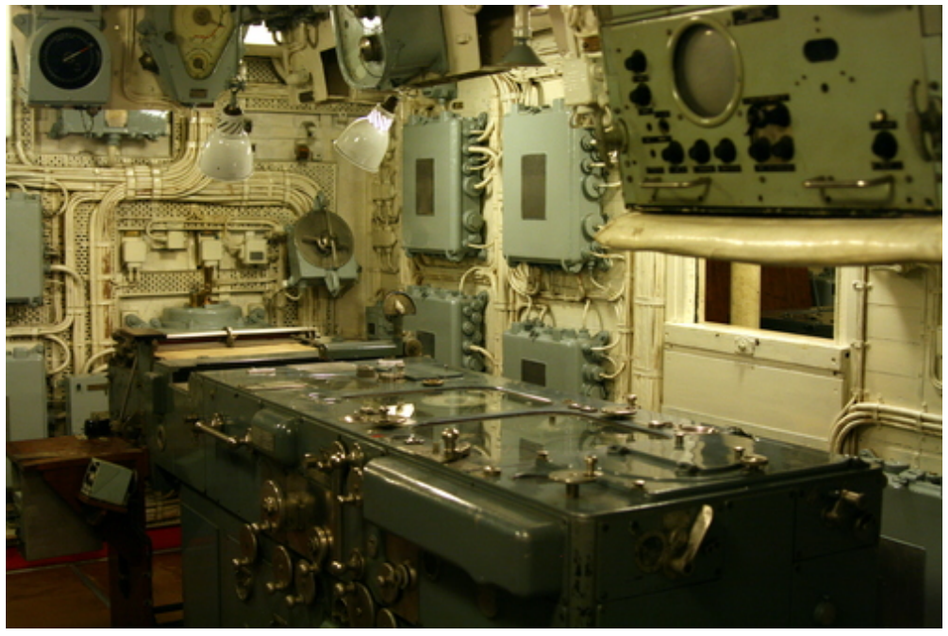
4. Quantum computing ***@#? computing with negative probabilities**

Analog computers: examples

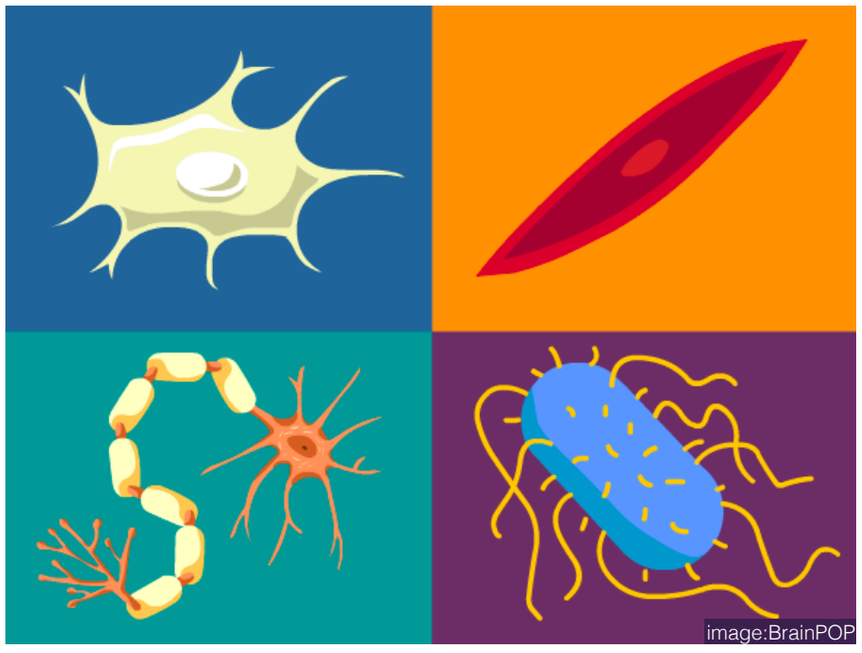
Differential analyzer
1920-40s



Legno compiler: modern electronic analog computation



Navy fire control computer
used through 1980s



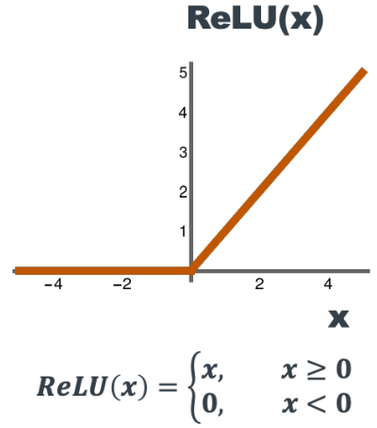
Biology
~3.7 billion years - present

example of HW problems:

Show how the GPAC analog computer can simulate the differential equation:

$$x'(t) = \tanh(x(t)).$$

Hint: if $x'(t) = \tanh(x(t))$ then $x''(t) = (1-(\tanh(x(t)))^2)x'(t)$. (No additional knowledge of differential equations is needed to solve this problem.)



Come up with a system of coupled chemical reactions that computes the ReLU function, and can thus be composed in ReLU neural networks…

(Value x is represented as a difference of the concentrations of two species X+ and X^-)

Major Topics

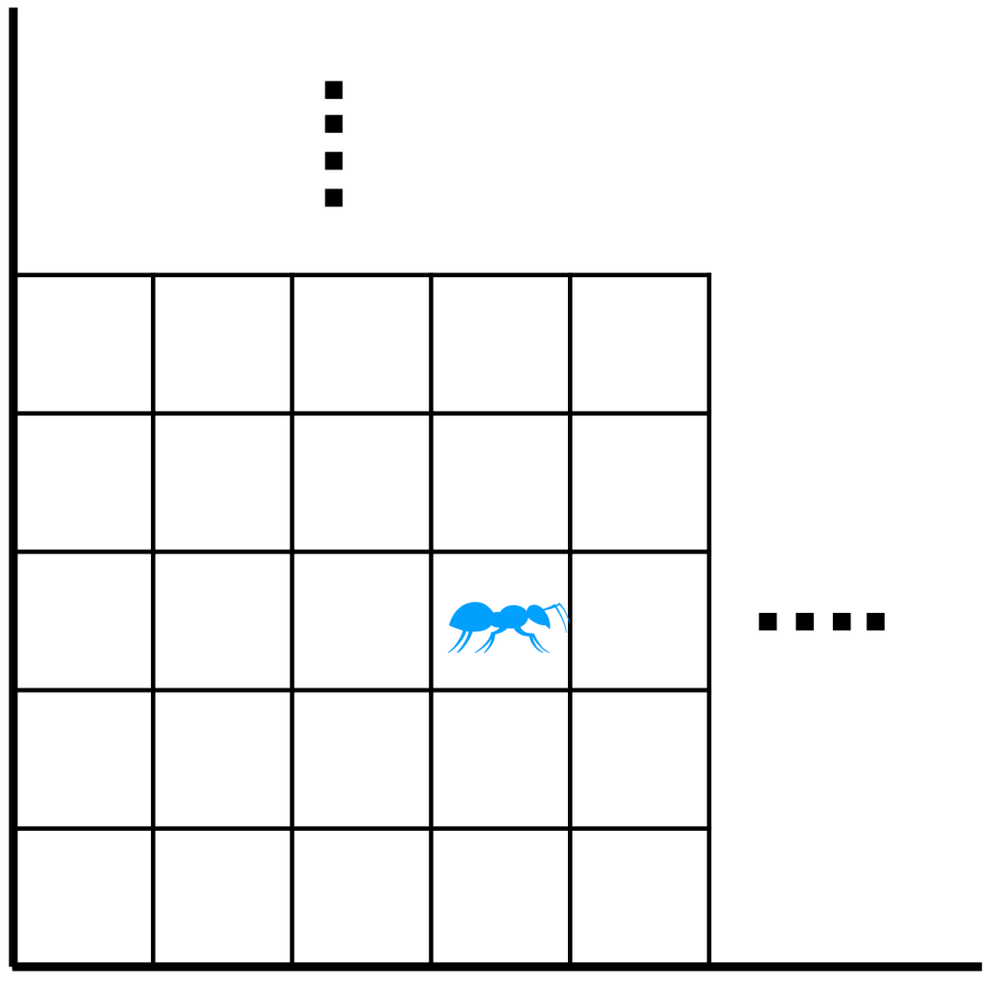
1. Analog/real-valued computation — **computation gets real**

2. Turing machines, cellular automata, counter machines, etc — **almost anything can compute everything**

3. Reversible computing — **computing without using energy**

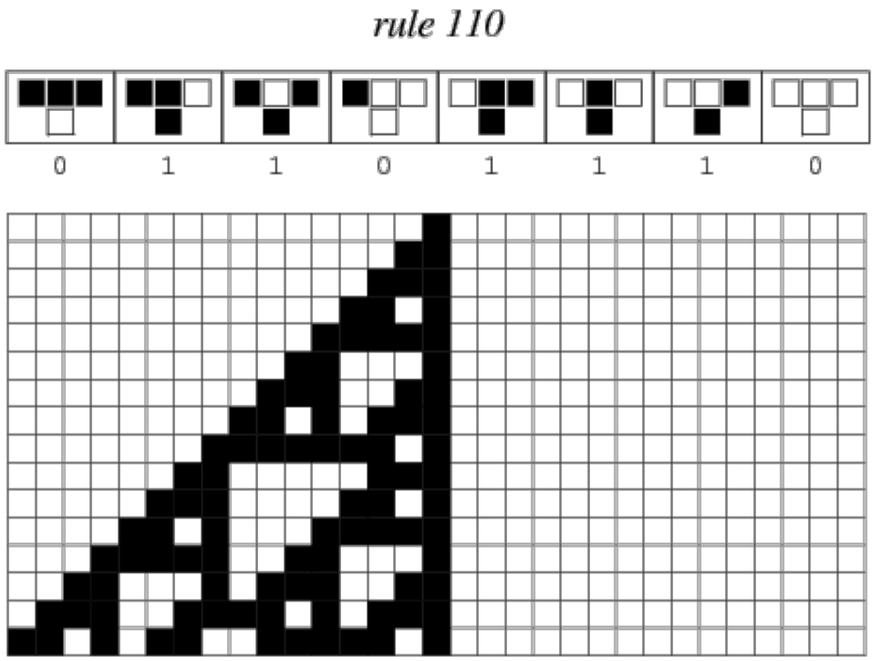
4. Quantum computing — ***@#? computing with negative probabilities**

example of HW problem:



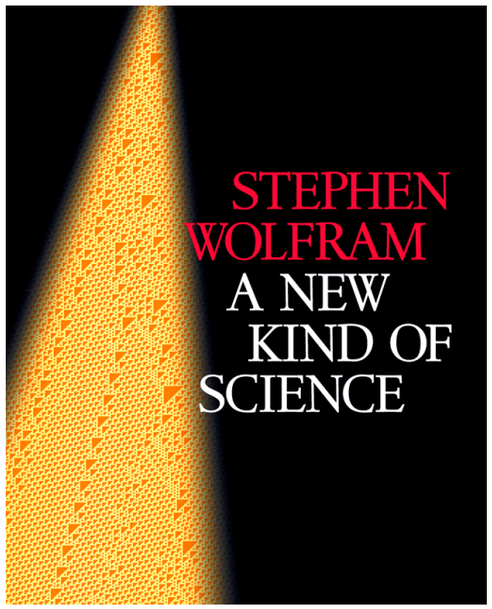
Consider a lonely ant that wonders around a quadrant of the infinite plane. It has a finite number of internal states, and at each step it moves one square north, south, east, or west, or it halts never to move again. It cannot write anything on the squares it touches, and it cannot see. All it knows about its position is whether it is touching the southern or western edge of the plane. Show that the ant can be computationally universal, with the input being encoded somehow in the initial position of the ant, and the output being encoded in the final position of the ant where it halts.

Even the simplest “laws of physics” are capable of universal computation



Elementary Cellular Automata

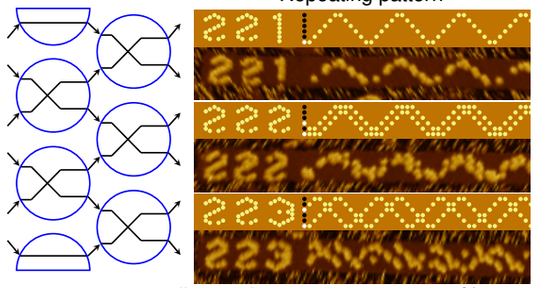
Conway’s Game of Life



Woods, Doty, Myhrvold, Hui, Zhou, Yin, Winfree (Nature 2019)

a ZIG-ZAG

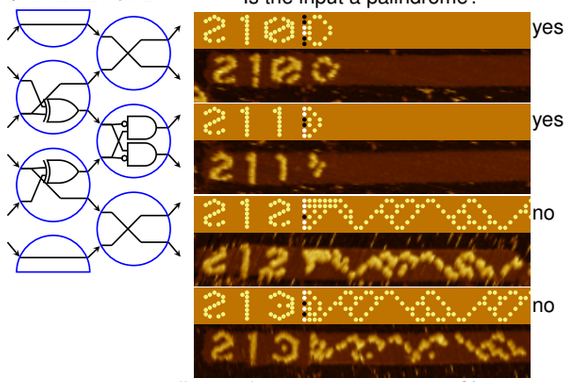
Repeating pattern



tile attachment error rate: $0.063\% \pm 0.002$

b PALINDROME

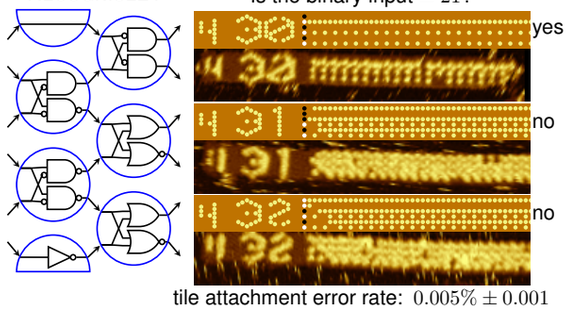
Is the input a palindrome?



tile attachment error rate: $0.017\% \pm 0.002$

c RECOGNISE21

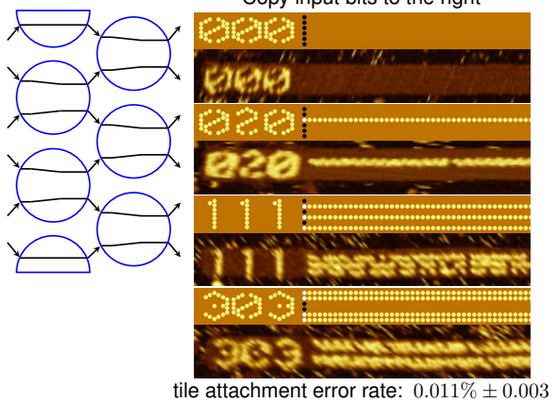
Is the binary input $= 21$?



tile attachment error rate: $0.005\% \pm 0.001$

d COPY

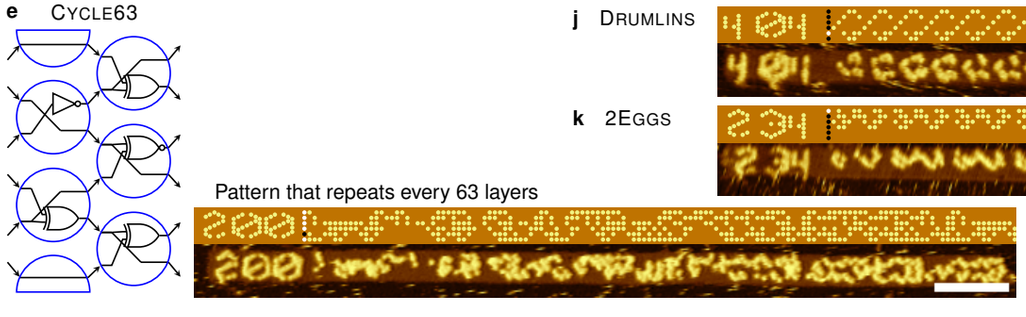
Copy input bits to the right



tile attachment error rate: $0.011\% \pm 0.003$

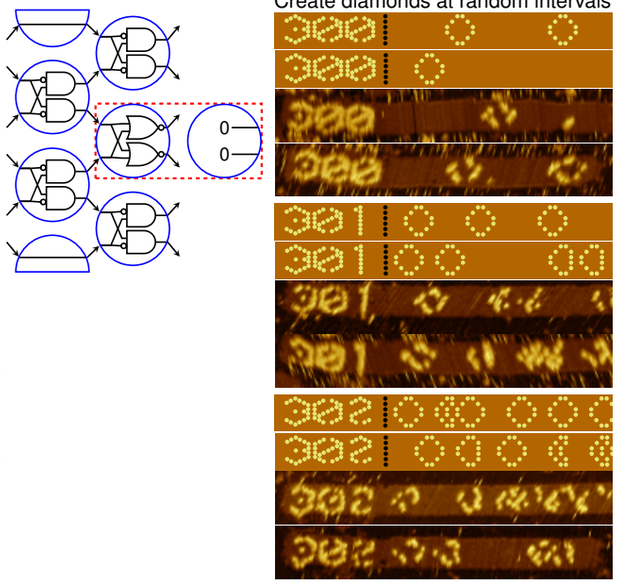
e CYCLE63

Pattern that repeats every 63 layers



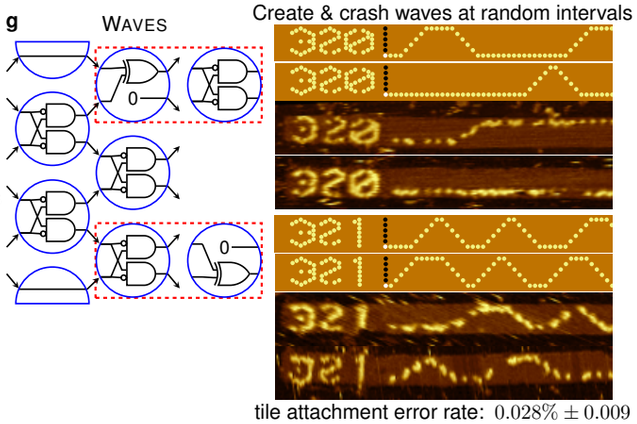
f DIAMONDSAREFOREVER

Create diamonds at random intervals



g WAVES

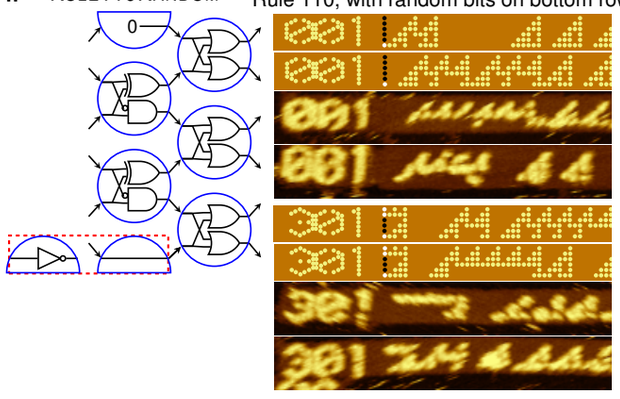
Create & crash waves at random intervals



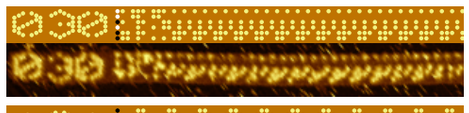
tile attachment error rate: $0.028\% \pm 0.009$

h RULE110RANDOM

Rule 110, with random bits on bottom row



i RULE30

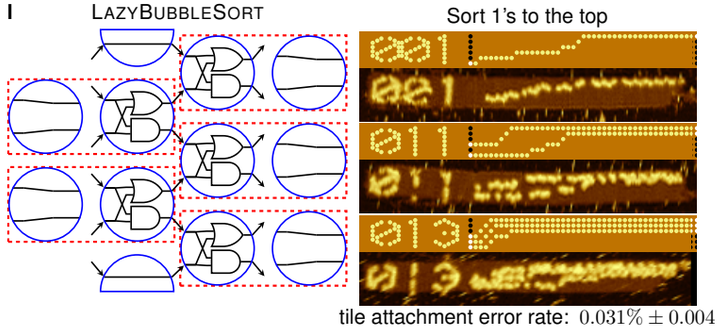


j DRUMLINS

k 2EGGS

l LAZYBUBBLESORT

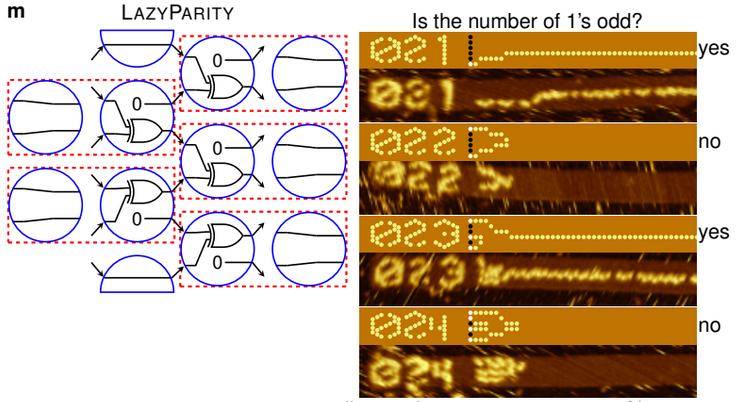
Sort 1's to the top



tile attachment error rate: $0.031\% \pm 0.004$

m LAZYPARITY

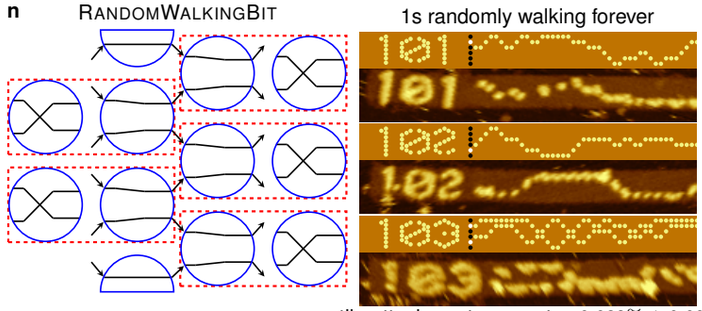
Is the number of 1's odd?



tile attachment error rate: $0.011\% \pm 0.004$

n RANDOMWALKINGBIT

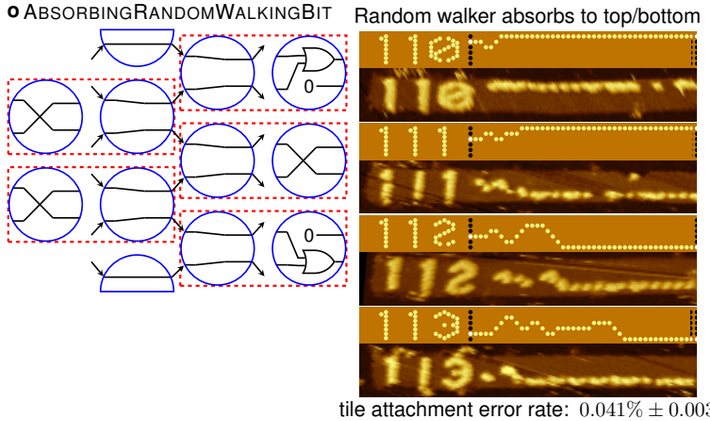
1s randomly walking forever



tile attachment error rate: $0.032\% \pm 0.005$

o ABSORBINGRANDOMWALKINGBIT

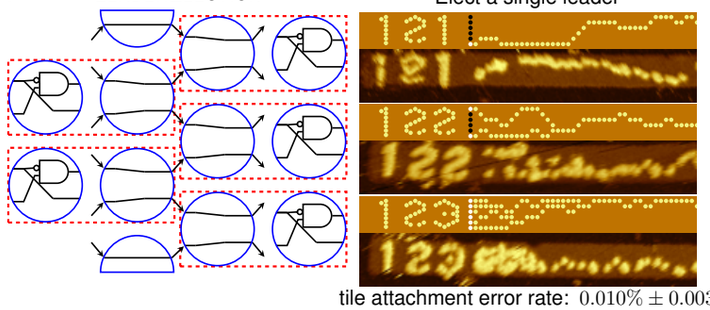
Random walker absorbs to top/bottom



tile attachment error rate: $0.041\% \pm 0.003$

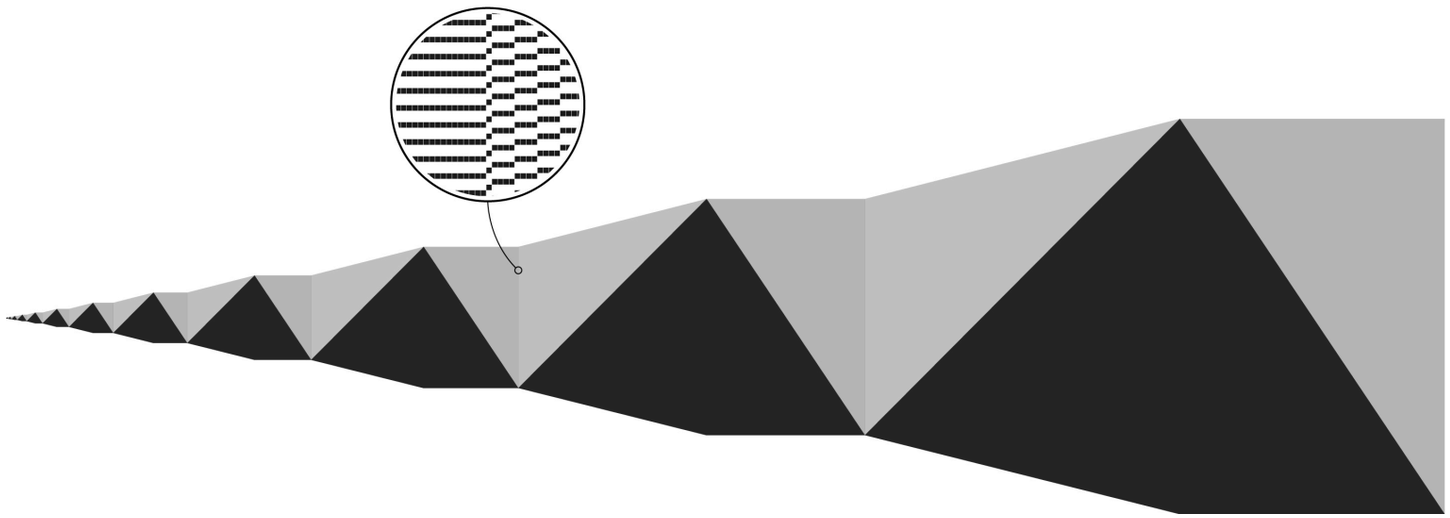
p LEADERELECTION

Elect a single leader

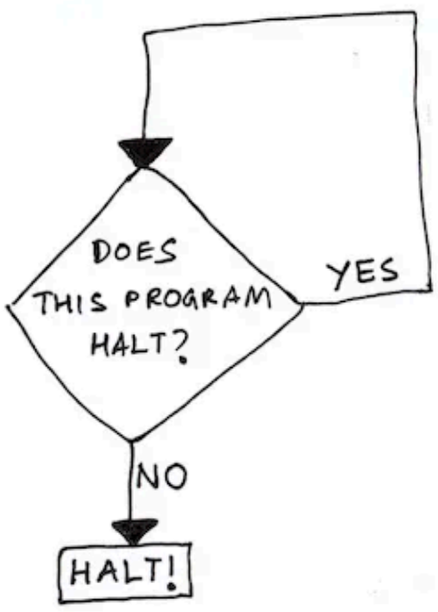


tile attachment error rate: $0.010\% \pm 0.003$

The Uncomputable (Limits of Computation)



A visualization of the longest-running five-rule Turing machine currently known. Each column of pixels represents one step in the computation, moving from left to right. Black squares show where the machine has printed a 1. The far right column shows the state of the computation when the Turing machine halts.
—from quantamagazine.com



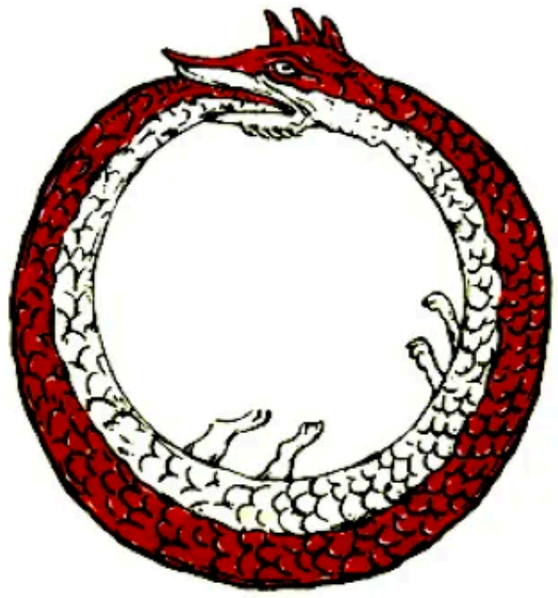


image: wired magazine

Major Topics

1. Analog/real-valued computation — **computation gets real**

2. Turing machines, cellular automata, counter machines, etc — **almost anything can compute everything**

3. Reversible computing — **computing without using energy**

4. Quantum computing — ***@#? computing with negative probabilities**

Landauer principle

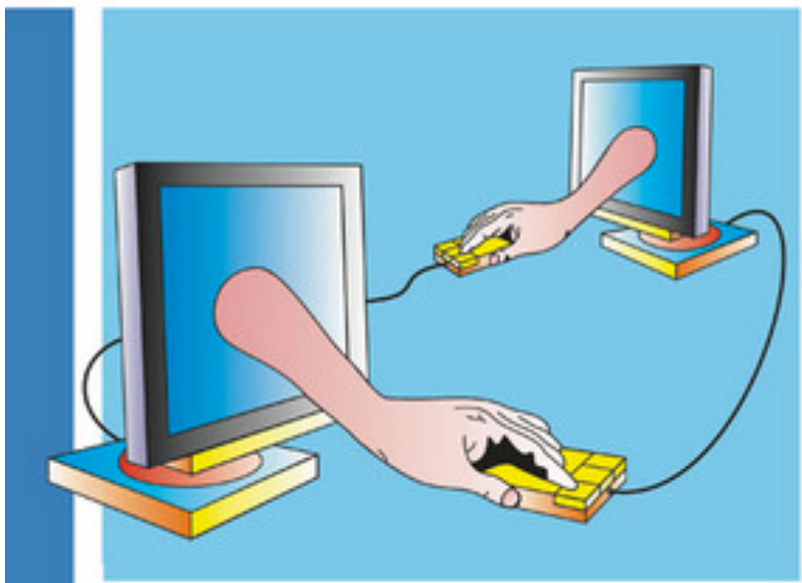
Erasing 1 bit of information must consume at least $kT \ln 2$ of free-energy

- This is 2.85 trillionths of a watt (at room temp)
- Currently computers consume millions of times more power per computational step
- But with technology improving, we are expected to reach this limit in several decades. What do we do then?

k=Boltzmann constant (1.38×10^{-23} J/K)
T=temperature (K)

Reversible computing

- Computing without erasing temporary information
- Hard part: not using much more memory than in normal computation
- Then can in principle compute using arbitrarily little energy per computational step



Major Topics

1. Analog/real-valued computation **computation gets real**

2. Turing machines, cellular automata, counter machines, etc **almost anything can compute everything**

3. Reversible computing **computing without using energy**

4. Quantum computing ***@#? computing with negative probabilities**

“I think I can safely say that nobody understands quantum mechanics”—Richard Feynman

Quantum Computing

- The only technology with the potential to fundamentally change the landscape of computational complexity—what is efficient (polynomial time).

- Most famous result: Peter Shor's polynomial-time quantum algorithm for factoring integers.

Is this class for you?

- **This is a theory course: we will spend time studying definitions, theorems and proofs**
- **There are limited practical applications of the material we cover, at least in the short term. (Limited industry interest)**
- **Do you like solving puzzles?**

Grading Policy

Homework (60%)

every 1-2 weeks

Projects (40%)

4 projects, one for each major topic

Normal grading scale: 93–100 A ; 90–92 A- ; 87–89 B+; etc

Attendance at all classes is expected, but not part of the grade. In borderline cases, extra credit will also be given for regular participation in class, coming to office hours, and participation on Piazza.

Homework

(60% of grade)

You may discuss homework problems with other students (indeed it's encouraged and expected!) and me/TA. I expect engaged office hours after class.

The solutions turned in must be written entirely by you. You must write the names of all the students you collaborated with at the top of your homework.

Homework must be typed and submitted to Gradescope

example from 2021
meant to scare you:

Homework

(60% of grade)

Problem 2. Characterizing unstable configurations
In this problem we derive a useful characterization of stable and unstable configurations. We use this characterization in class to show that stable computation is "linear". We begin with the following definitions:

Definition 1. A τ-*truncation* of a configuration $\vec{c}$ is a configuration $\vec{d}$ such that $\forall$ species S, $\vec{d}(S) = \min(\tau, \vec{c}(S))$. In other words, $\vec{d}$ is like $\vec{c}$ but all species' counts above τ are truncated.

Definition 2. Let $S \subseteq \mathbb{N}^k$. We say $\vec{m} \in S$ is a *minimal element* of S if $\forall \vec{s} \in S$, $\vec{s} \leq \vec{m}$ implies $\vec{s} = \vec{m}$. Phrased another way, $\vec{m} \in S$ is minimal if we can't get another element of S by decreasing $\vec{m}$ in some dimensions.

In this problem we work toward proving the following lemma which characterizes output stability by τ-truncation:

Lemma 1. $\forall$ CRNs, $\exists \tau \in \mathbb{N}$ such that if $\vec{o}$ and $\vec{o'}$ have the same τ-truncation, then $\vec{o}$ is output stable iff $\vec{o'}$ is output stable.

To prove this, we'll first prove a helpful lemma in part **(a)**, then prove the main lemma in part **(b)**:

(a) Let Ω be the set of configurations that are *not* output stable. Prove that for any configuration $\vec{c}$, we have $\vec{c} \in \Omega$ iff $\exists$ a minimal element $\vec{m}$ of Ω such that $\vec{m} \leq \vec{c}$.

(b) Prove Lemma 1. (You will probably need to use part **(a)**).

Hint: Use **Dickson's lemma:** Every set $S \subseteq \mathbb{N}^k$ has finitely many minimal elements. If S is non-empty then it has at least one minimal element.

Projects

(40% of grade)

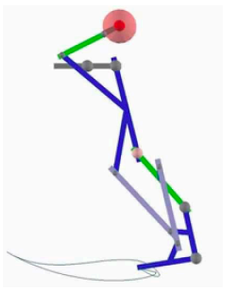
Groups of 1-4 students (depending on class size)

Typically: present a single paper or published idea which shows some example of "unconventional computation"

- ~15-30 min in-class presentation
- "homework problem" with solution (similar in style to homework problems)

Academic presentations are a big part of graduate school and are a learning objective of this class

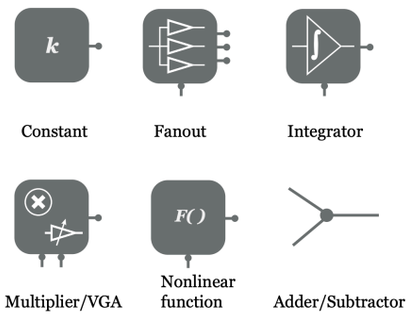
mechanical linkages can compute arbitrary algebraic curves



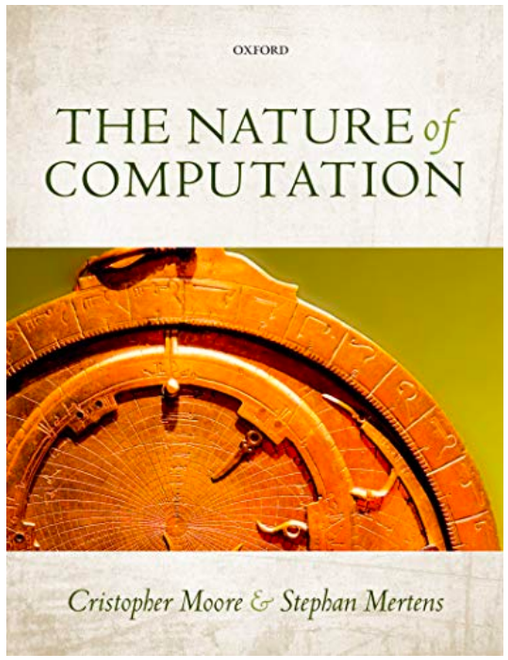
Arabidopsis plants perform arithmetic division to prevent starvation at night

DNA computing

Legno compiler: modern electronic analog computation



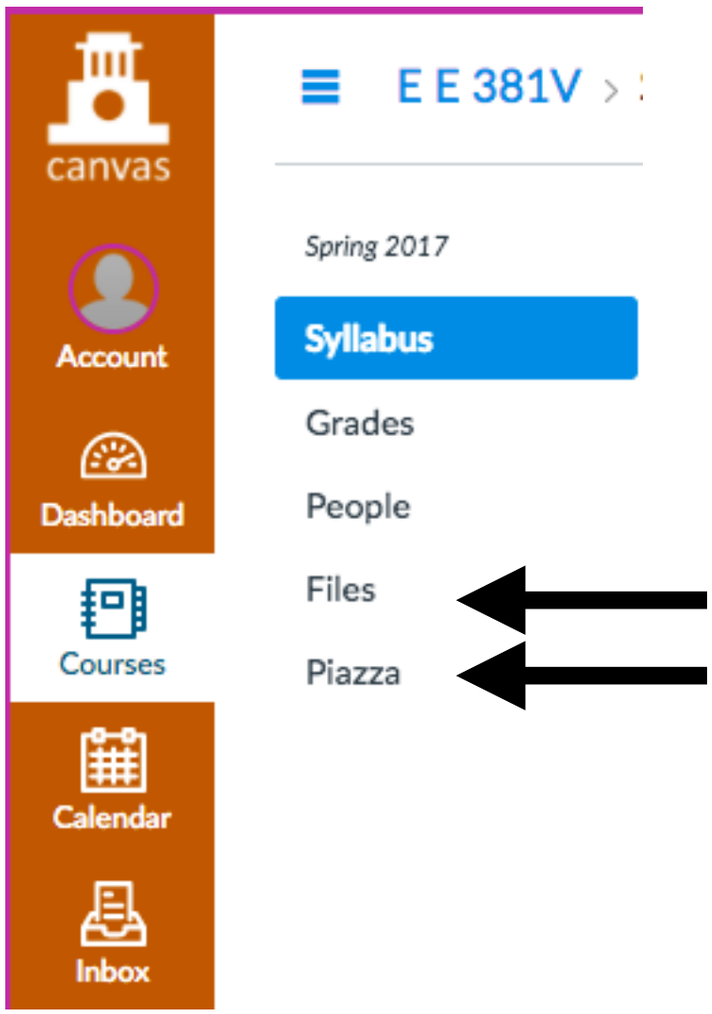
There is no textbook for this course, but the following book does a great job in covering the nature of computation inside and outside of electronic computers:



Lecture notes will be uploaded to Canvas after each class

(Optional) suggested papers will also be posted for different topics

Course Materials on Canvas



Major Topics

	approximate number of lectures
1. Analog/real-valued computation	4
2. Turing machines, cellular automata, counter machines, etc	7
3. Reversible computing	6
4. Quantum computing	7