A story of computation—and, in particular, the story of computation "in the wild"—begins with digraphs.

You have a language—this is, briefly just a list of words. Sentences composed of a finite number of words.

The words: infinitely long list! (Or so.)

The goal: can you figure out the underlying grammar of the language? What will tell us why

"I ate the sheep"

is a valid sentence, but

"I the ate sheep"

isn’t.

"Infinite use of finite means."

The problem: how to describe a grammar? What does it look like? What are the minimal resources you need?

"Infinite productivity"—"finite means" (don’t just memorize the good sentence, but learn any a Universal grammar)
The answer (in 1965) —
the Chomsky hierarchy

decreasing number \implies increasing expressive power.

Think about levels
"Derivation"

Grammars are described in terms of derivation,

rewriting, or "production" rules.

\[ V_N = \{ A, B, C, D, \ldots \} \]
\[ V_T = \{ a, b, c, d, \ldots \} \]

an example of a production rule:

\[ aAb \rightarrow aBb \]
\[ N \rightarrow AN \]
\[ A \rightarrow a \]

aaaaAbbb
\[ \rightarrow \] aaaaBbbb.
Language rules for the levels

**level 3**:  
\[ A \rightarrow aB \]  
(non-terminal symbol rewritten as combination)

**level 2**:  
\[ A \rightarrow \text{(any word made from combination of non-terminal symbols)} \]

**level 1**:  
\[ Q_1AQ_2 \rightarrow Q_1 \text{ (any word made from } Q_1, Q_2 \in (V_N U V_T)^* \text{)} \]

**level 0**:  
no restrictions
Examples

[NP] $\rightarrow$ [the] [N]  
level 3

[NP] $\rightarrow$ [AP][NP]  
level 2

[SUB][VERB][OBJ] $\rightarrow$ [OBJ][VERB]  
by [SUB]  
level 0

[SUB][NP][VP:ing] $\rightarrow$ [NP:ing][VP:ing]  
agreement  
level 1

$\Rightarrow$

level 2  "context-free"  
(really: context ignoring!)

common in definitions of computer languages.

level 1  - computing notion such as verb agreement;  
"context"

level 0  - long-range dependence.  
"transformational grammar"  
(& the Chomskyian one.)
**PUMPING LEMMAS**

1. Prove that you have "at least" a regular language.

   \[ XYZ \rightarrow X Y^n Z \]

2. Prove that it must actually be level 2.

   \[
   ((( ( ))) )
   \]

   Can "pump" up the parentheses.

   \[ XYZ \rightarrow X^n Y Z^n \]

   Repeated n times.
Finite State Automata

\[ \text{[environmental]} \]

\[ \text{[input]} \]

\[ \text{[input letter]} \]

\[ \text{[a, b, c, ...]} \]

\[ \text{[internal state]} \]

\[ \text{[A, B, C, ...]} \]

includes:

\[ A \to aB \]
\[ A \to aC \]
\[ B \to aD \]
\[ \&c. \]

all of these are level 3 production rules!

"Non-deterministic"—when you are in state A, what do you do when you receive "a"?
now it is deterministic!

This is nice, for many reasons—most of all, you know what "state" the system is in.
You will notice that some automata tend to drag you into a subspace of states from which you can not escape.

E.g. — in our friend, keep hitting the system with "b" — what happens?

It is not so like hitting "control-C" on your computer when you've done something wrong. It is, in other words — a reset.

In fact, all machines amount to a collection of permutations and resets.

What is a machine that always has an undo?

A. a Mac.
B. a snap!
GROUPS are a kind of finite state automaton!

Wait... what does this mean?

For every element, 
\[ g \in G \]
there is an inverse 
\[ g^{-1} \]
that takes you back.

No resets! (No "undo" command always available).
JORDAN-HOLDER DECOMPOSITION

A group is either a simple group, or has a hierarchical decomposition into it has "normal subgroups" nested!

What is a normal subgroup?

\[ G \quad N \quad \text{if} \quad gNg^{-1} \text{ is also in } N. \]

Is \( \mathbb{Z}_3 \) a normal subgroup of \( S_3 \)?

\[ a \in N \quad b \in N \quad \text{if} \quad bab^{-1} \in N? \]
each state in the above original machine has a "coordinate" representation in the above decomposition.

--- talk about SEMIGROUPS.

HERE: talk about the Krohn-Rhodes theorem.
What is this top level?

- to know what's going on (how I'm swapping, moving around and it) I don't have to know anything about the low levels.

[Diagram of waveforms with annotations for full, smoothed, and residual components]
How to handle stochasticity?

You may think this is a little like the non-deterministic case — you’d be right!

We’ll consider the singleton alphabet only.

A Markov process!
Basic idea:

Consider all semigroup action on the state.

permute

has

zero probability

partial reset

&c.

Each of these gets a probability given by the original Markov process.
Now

each transition has a probability.

so: roll the dice, pick a member of the semigroup, & let's go!
Figure 1: Example of sonogram of Bengalese finch song and its syllable label sequence. (A) Sonogram of Bengalese finch (BF09) with syllable labels annotated by three human experts. Labeling was done based on visual inspection of sonogram and syllables with similar spectrogram given same syllable. (B) Bigram automaton representation (transition diagram) of syllable sequences obtained from same song set as (A). Ellipses represent one syllable and arrows with values represent transitional probabilities. Rare transitions with probabilities < 0.01 are omitted. (C) POMM representation of same sequences as (B). Syllables that have significant higher-order dependency on preceding syllables (colored states in (B)) are divided into distinct states depending on preceding syllables (context).