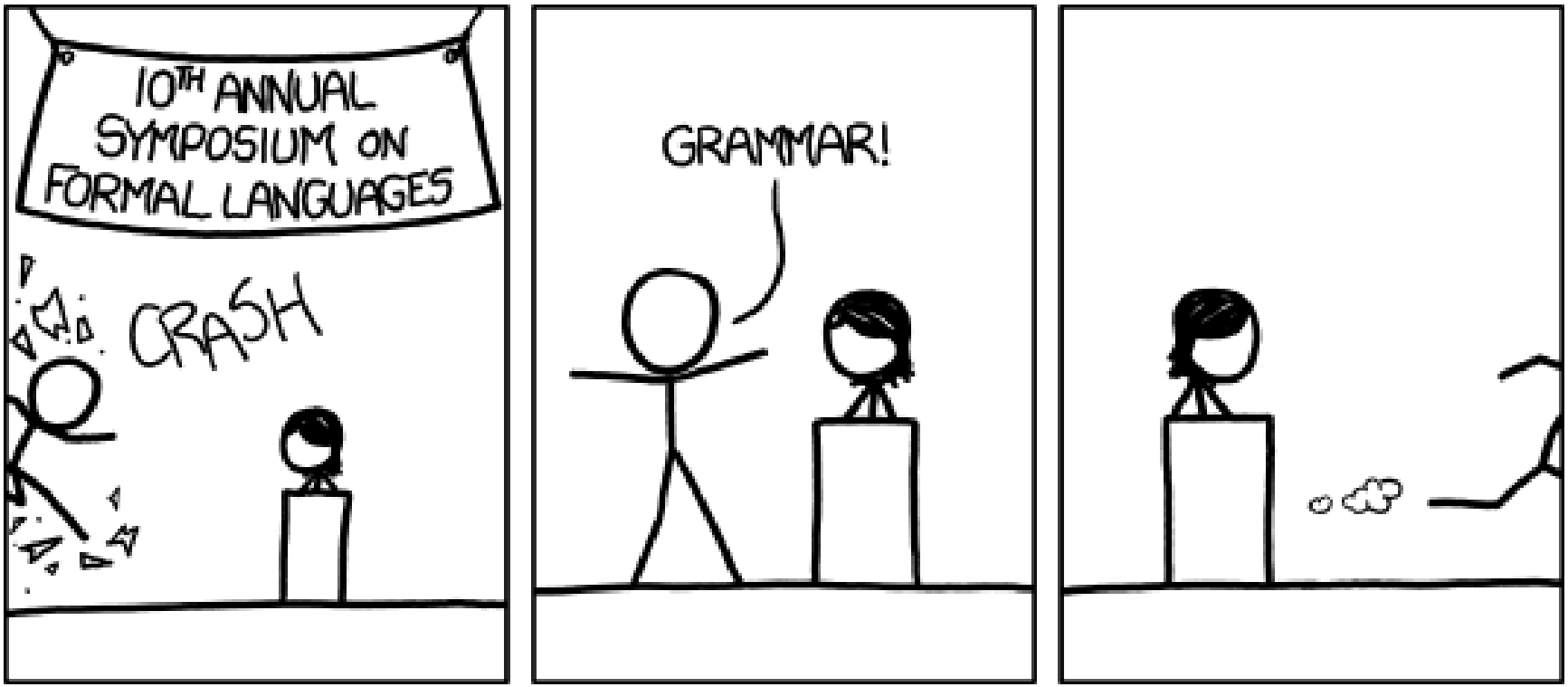
Discrete Mathematics

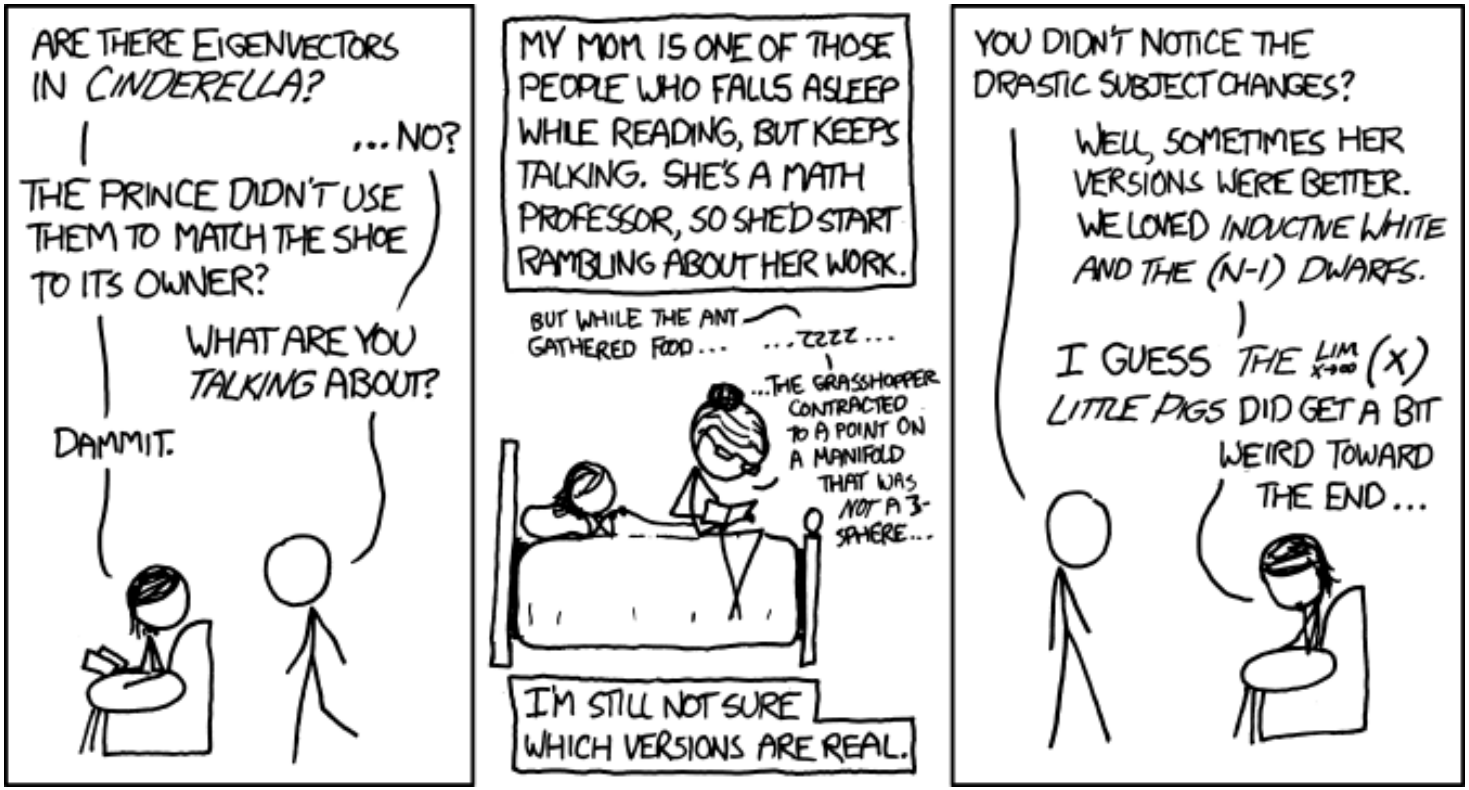
MATH1064, Lecture 35

Jonathan Spreer



Extra exercises for Lecture 35

Section 13.1: Problems 1–5



Modelling computation

- A key property of a computer is that it is programmable
- First programmable "machine": Analytical engine due to Charles Babbage (1837)
- First computer program: Compute Bernoulli numbers due to Ada Lovelace (1843)

- How to communicate with a machine?
- How to build a machine / model computation?

These questions must be answered before we build a an actual computer

Modelling computation

Remember addition of integers in binary?

$(x_n x_{n-1} \dots x_1 x_0)_2 + (y_n y_{n-1} \dots y_1 y_0)_2 = (z_n z_{n-1} \dots z_1 z_0)_2$

(assuming $x_n = y_n = 0$)

How do we model this by a machine?

s_0 = "remember previous carry is 0"
s_1 = "remember previous carry is 1"

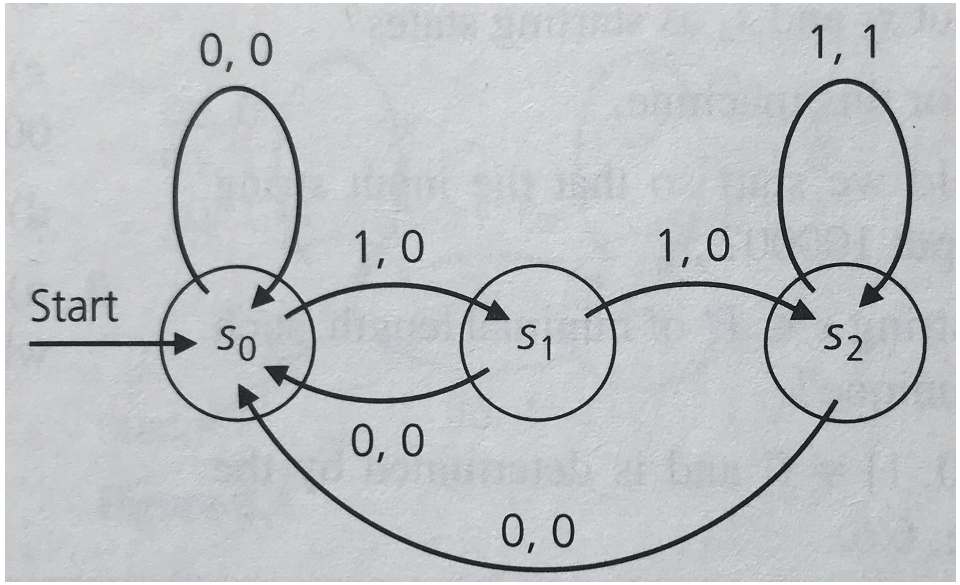
Finite state machine

A finite state machine $M = (S, I, O, f, g, s_0)$ consists of

- a finite set S of states,
- a finite input alphabet I,
- a finite output alphabet O,
- a transition function $f\colon\ S \times I \to S$,
- an output function $g\colon\ S \times I \to O$, and
- an initial state s_0.

Question

Given the input 1110101111 what is the output of the following finite state machine?



Let's generalise: Formal languages

- Communication with finite state machine through input / output
- Strings of symbols (eg., 0 and 1) with a certain structure
- This is called a Formal language

Formal languages are essential for:

- designing programming languages
- building compilers
- complexity theory (analysing difficulty of computational tasks)

We can talk about all this without having built a computer

How do we communicate?

How can we decide that a sentence is grammatically correct?

1. Colourless green ideas sleep furiously.
2. Furiously sleep ideas green colourless.

Neither sentence (1) nor sentence (2) has ever occurred in a conversation, but we know that (1) is correct and (2) is not!

How can a speaker generate new sentences?

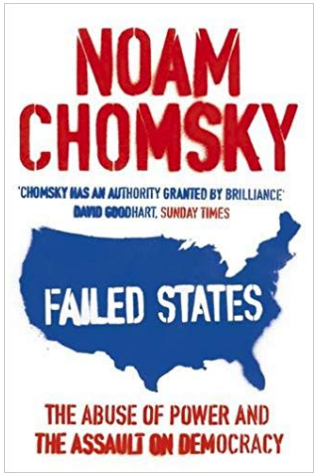
1. Functional syntax or behaviourist theory (Piaget)
2. Generative syntax theory (Chomsky)

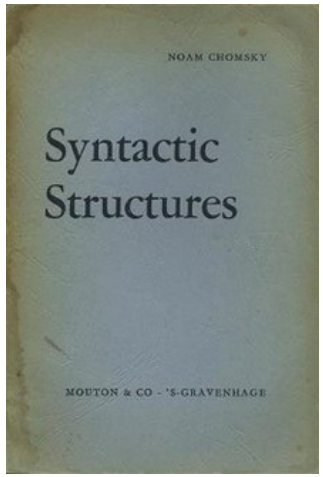
Chomsky's theory: Language is only partially learned!

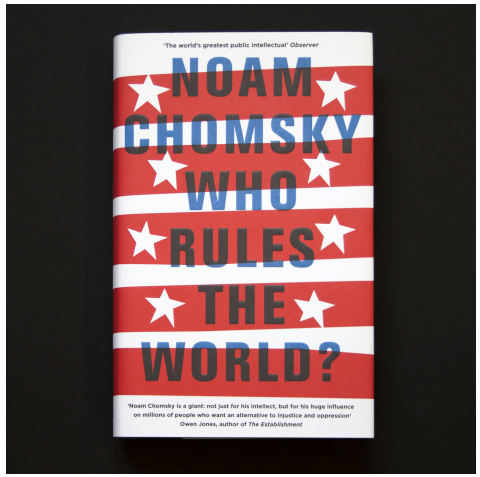
1. There is a universal or generative grammar – principles valid for all languages that we are born with.
2. We can judge sentences correct because we possess an abstract system of unconscious knowledge about language.

Noam Chomsky

Linguist, philosopher, cognitive scientist, historian, and social critic.







The Guardian, 2005: the most influential intellectual of our times

Formal languages

Let A be a finite alphabet.

A formal language L is a set of strings with symbols in A.

The empty string is denoted λ.

Examples:

1. $A = \{0, 1\}$, $L = \{0^n 1^m \mid m, n \in \mathbb{N}\}$
2. $A = \{0, 1\}$, $L = \{0^n 1^n \mid n \in \mathbb{N}\}$
3. $A = \{a, b, \ldots, z\}$, $L =$ all English words

Grammars

How to describe an infinite language with a finite amount of resources?

Use grammars as formalised and categorised by Chomsky (1957).

Idea: Grammar of language L will be device that generates all the grammatical sentences of L and none of the ungrammatical sentences.

Aside: Chomsky aimed (and failed) to define natural languages.
His work became an important and useful tool in theoretical computer science and mathematics.

Example: A simplistic grammar to generate English sentences

- Σ := all English words
- V := structural components in an English sentence, such as ⟨**sentence**⟩ ⟨**subject**⟩, ⟨**predicate**⟩, ⟨**noun**⟩, ⟨**verb**⟩, ⟨**article**⟩, and so on.
- S := ⟨**sentence**⟩
- Some typical productions are:

⟨**sentence**⟩ → ⟨**subject**⟩ ⟨**predicate**⟩

⟨**subject**⟩ → ⟨**noun**⟩

⟨**predicate**⟩ → ⟨**verb**⟩ ⟨**article**⟩ ⟨**noun**⟩

⟨**noun**⟩ → **Lori**

⟨**noun**⟩ → **algorithm**

⟨**verb**⟩ → **wrote**

⟨**article**⟩ → **an**

Grammars

Phase-structure grammar

A phase-structure grammar $G = (V, T, S, P)$ consists of

1. a vocabulary (or alphabet) V,
2. a subset $T \subseteq V$ of terminal symbols,
3. a start symbol $S \in V$,
4. a finite set of productions P.

The non-terminal symbols are $N = V - T$. Every production in P must have at least one non-terminal symbol on its left side.

The language generated by G is the set $L(G)$ of all words in terminals that can be derived from S using the productions P.

The set of all sentences (or words) over V is $V^\star$.
So $L(G) \subseteq T^\star \subseteq V^\star$.

Example:

$G = (V, T, S, P)$ where
$V = \{0, 1, S, A\}$ (vocabulary or alphabet),
$T = \{0, 1\}$ (terminal symbols),
$P = \{S \to 0A,\ S \to 0S,\ A \to 1A,\ A \to 1\}$ (productions).

$V^\star$ is the set of all words in V, e.g. $01AAA1$, λ, $SSA101$

$T^\star$ is the set of all words in T, e.g. 011, λ, 101

$L(G) \subseteq T^\star \subseteq V^\star$

Let's derive a word in the language $L(G)$:

Example 2

$G = (V, T, S, P)$ where
$V = \{0, 1, S\}$
$T = \{0, 1\}$,
$P = \{S \to 0S1,\ S \to 01\}$.

Let's derive a word in the language $L(G)$:

Example 3

$G = (V, T, S, P)$ where
$V = \{0, 1, 2, S, B, C, H\}$
$T = \{0, 1, 2\}$,

$$P = \{S \to 0BC,\ S \to 0SBC,$$

$$CB \to BC,$$

$$0B \to 01,\ 1B \to 11,\ 1C \to 12,\ 2C \to 22\}.$$

Let's derive a word in the language $L(G)$:

Summary

Phase-structure grammar

A phase-structure grammar $G = (V, T, S, P)$ consists of

1. a vocabulary (or alphabet) V,
2. a subset $T \subseteq V$ of terminal symbols,
3. a start symbol $S \in V$,
4. a finite set of productions P.

The non-terminal symbols are $N = V - T$. Every production in P must have at least one non-terminal symbol on its left side.

The language generated by G is the set $L(G)$ of all words in terminals that can be derived from S using the productions P.

The set of all sentences (or words) over V is $V^\star$.
So $L(G) \subseteq T^\star \subseteq V^\star$.

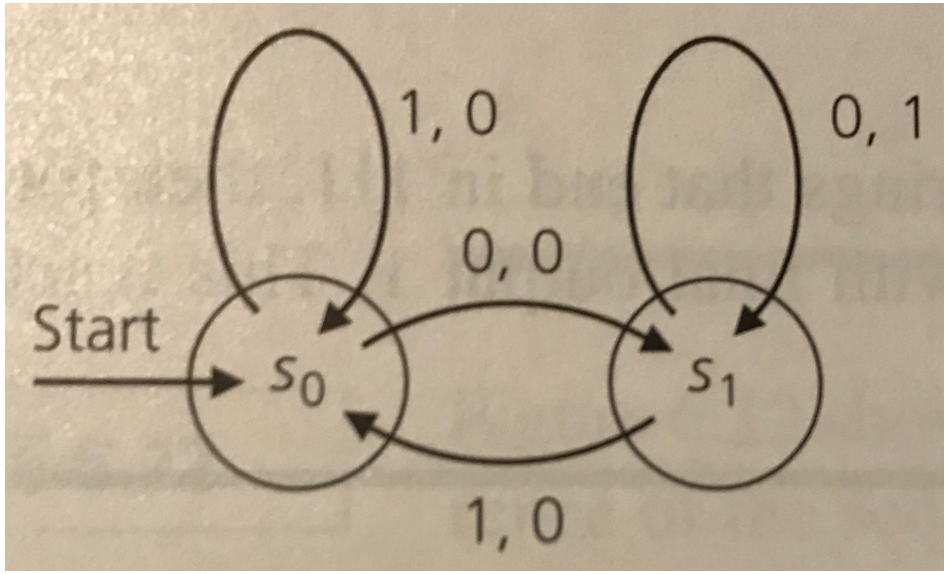
Exercise

Construct a one-unit delay machine: A finite state machine that given an input string $x \in \{0,1\}^\star$ outputs $0x$.

Eg. the input 10011 produces the output 010011.

Exercise

Consider the following finite state machine:



Determine the output for the following inputs:

- 111111
- 000000
- 011011
- 100100

Can you describe in words how strings are transformed?