Introduction to Automata Theory

- Study of abstract computing devices, or “machines”
- **Automaton** *(Representation)*= an abstract computing device
- **Note:** A “device” need not even be a physical hardware !

automaton

/ɔːˈtɒmət(ə)n/

noun

a moving mechanical device made in imitation of a human being.
"a collection of 19th century French automata: acrobats, clowns, and musicians"

- a machine which performs a range of functions according to a predetermined set of coded instructions.
  "sophisticated automatons continue to run factory assembly lines"
- used in comparisons to refer to a person who seems to act in a mechanical or unemotional way.
  "like an automaton, she walked to the door"

Definitions from Oxford Languages

Feedback

Introduction to Automata Theory

- It comprises the fundamental mathematical properties of hardware, software, and applications.
- Determine what can and cannot be computed.
- It has purely philosophical aspects.
- A fundamental question in computer science:

  Find out what different models of machines can do and cannot do

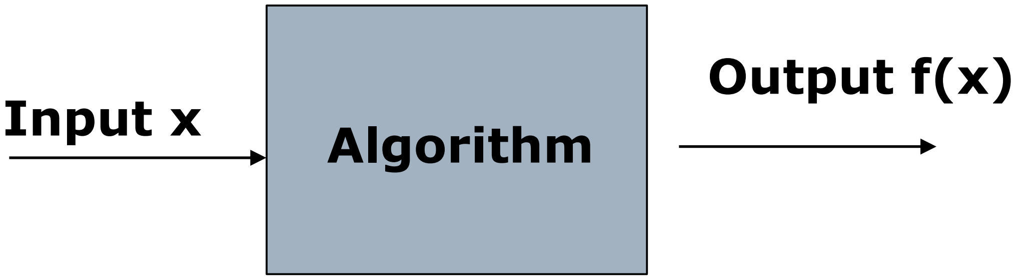
- The theory of computation
- Computability vs. Complexity

Introduction to Automata Theory

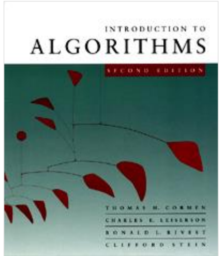
- Theory of computation -> Theory of Programs → Theory of Algorithms.
- Algorithms: A recipe to carry out input to output transformation
  - Finite
  - Deterministic.
  - Unambiguous.
- Every algorithm computes a function.

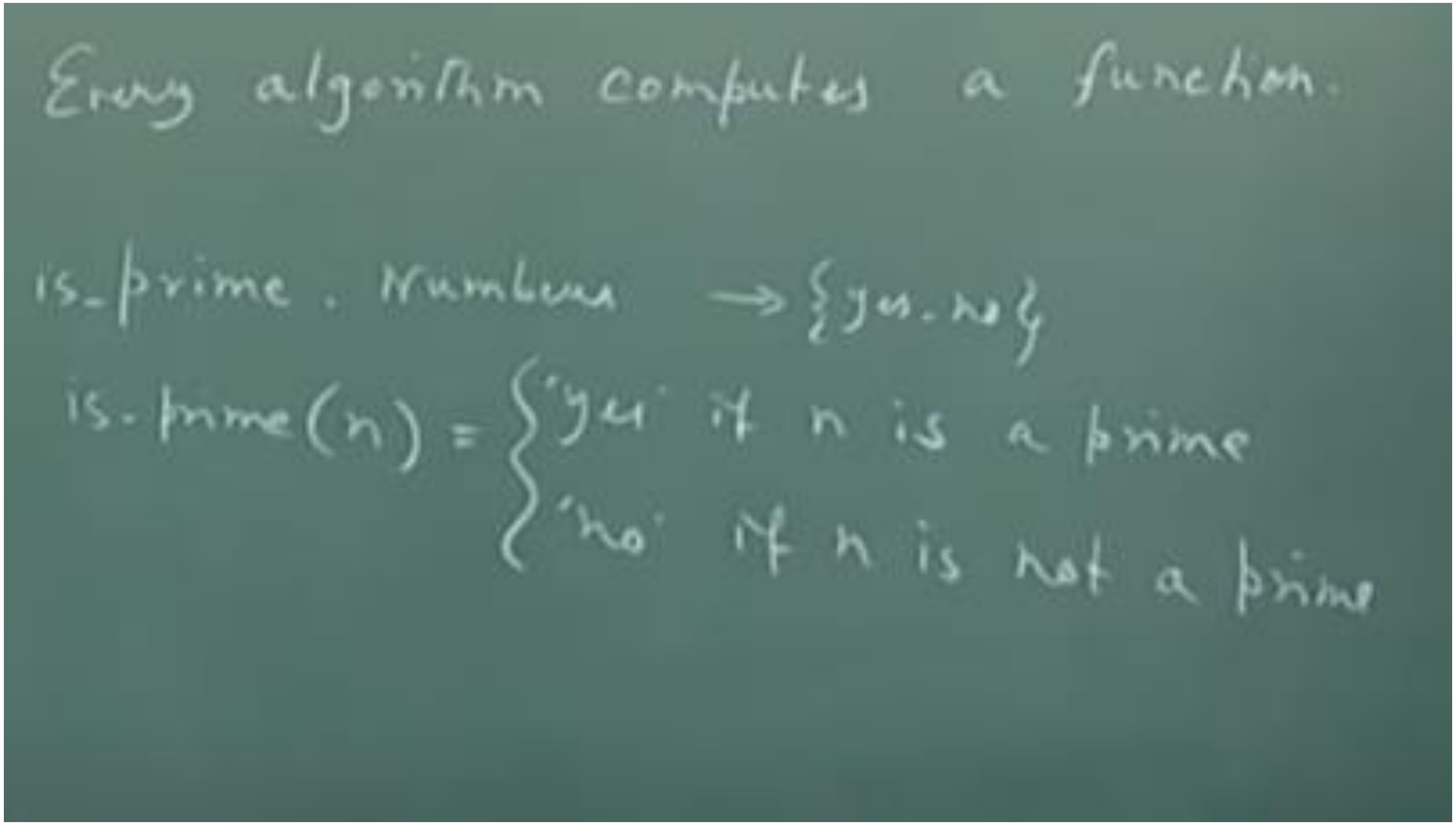
Introduction to Automata Theory

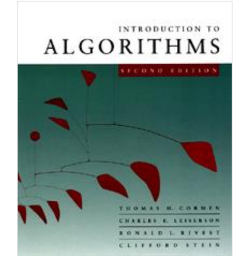
- Every algorithm computes a function.
- The algorithm tells how to obtain the output (desired) from the input (specific).



- Basic goal of TOC: To figure out for what functions we can have Algorithms.


INTRODUCTION TO
ALGORITHMS
SECOND EDITION

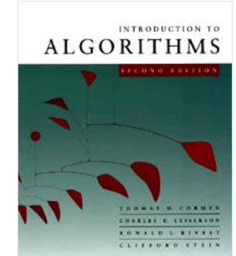

Every algorithm computes a function.
is_prime : Numbers → {yes, no}
is_prime(n) = 'yes' if n is a prime
'no' if n is not a prime



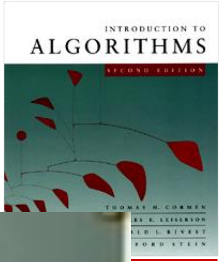
- We can say that algorithm is completely a function.

- Although it may be possible to define a function **BUT** the definition of the function does not immediately point out in all cases to an algorithm to compute that much.

- If that is the case then at least you can now see that there is a possibility that I may be able to define a function I may be able to describe what the output should be without having an idea how to obtain the correct answer?



- Although *you* have not possibly encountered such situations in your programming experience but it might surprise you that actually it is a fact that for most functions there are no algorithms to compute.
- If you think of the class of all functions, then only a tiny subset of these functions admit algorithms to compute them.
- Hence, the primary goal of TOC is going to be to figure out which functions can admit or will admit algorithms to compute them, and which not.



Set membership problem

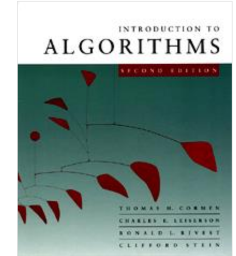
S a set

Given any a, to decide if $a \in S$

Suppose there is an algorithm to compute f.
For (a, b), given as input
we compute
$f(a)$ using the algorithm for computing f

$b = f(a) \iff (a,b) \in \text{graph}(f)$

Suppose we show that there is no algorithm to solve the set membership problem $\text{graph}(f)$.
Then, we can conclude that there is no algorithm to compute f.



Our sets are going to be sets of finite strings

Symbols $0, 1, a, b,$

Alphabet : An alphabet is a finite set of symbols.

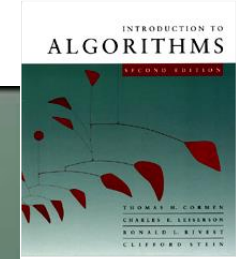
Ex:

$\{0,1\}$, $\{a,b,c,d,\dots,z\}$, ...

Σ is an alphabet.

Σ^* denotes the set of all finite strings over Σ.

$\Sigma = \{0,1\}$, Σ^*



$\Sigma = \{0, 1\}$

Σ^* is then the set of all finite binary strings.

A formal language L over the alphabet Σ is a subset of Σ^*.

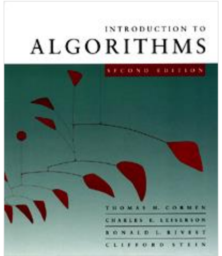
Ex: $\Sigma = \{0, 1\}$

$L = \{01, 11001, 011, 10101110\} \subseteq \Sigma^*$

$L_1 = \{x \in \{0,1\}^* \mid x \text{ has even number of 0's and even no. of 1's}\}$

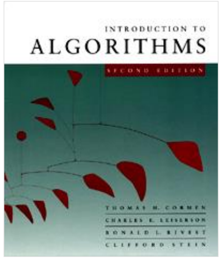
$L_1 \subset \{0,1\}^*$

What is Automata Theory?



- Nutshell: We shall be concerned with the set membership of formal languages.
- Also called, theory of formal languages.
- But we will come to that goal in a series of steps if you like, i.e. we are going to do we are going to invert the problem in some sense.

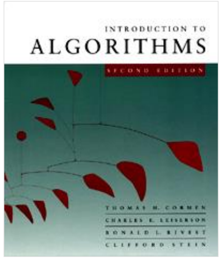
What is Automata Theory?



- We will think in terms of models of computation.
- It means some abstract way we are seeing we will describe a class of algorithms and the that abstract we in fact going to be by specifying what are called automata
- So our models of computations, for the time being are called **automata** of various kinds. So we will define a class of automata and then will ask the question:

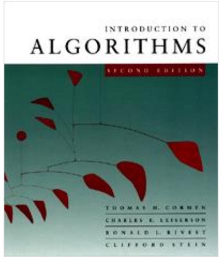
What kinds of set membership problem this class of automata can solve?

Theory of Computation: A Historical Perspective



1930s	• Alan Turing studies Turing machines • Decidability • Halting problem
1940-1950s	• "Finite automata" machines studied • Noam Chomsky proposes the "Chomsky Hierarchy" for formal languages
1969	Cook introduces "intractable" problems or "NP-Hard" problems
1970-	Modern computer science: compilers, computational & complexity theory evolve

Languages & Grammars



An alphabet is a set of symbols:

Or "**words**"

{0,1}

Sentences are strings of symbols:

0,1,00,01,10,1,...

A language is a set of sentences:

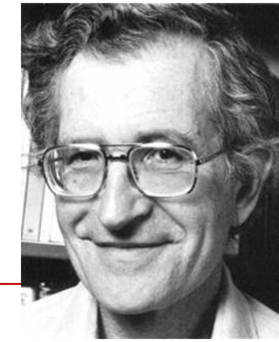
L = {000,0100,0010,..}

A grammar is a finite list of rules defining a language.

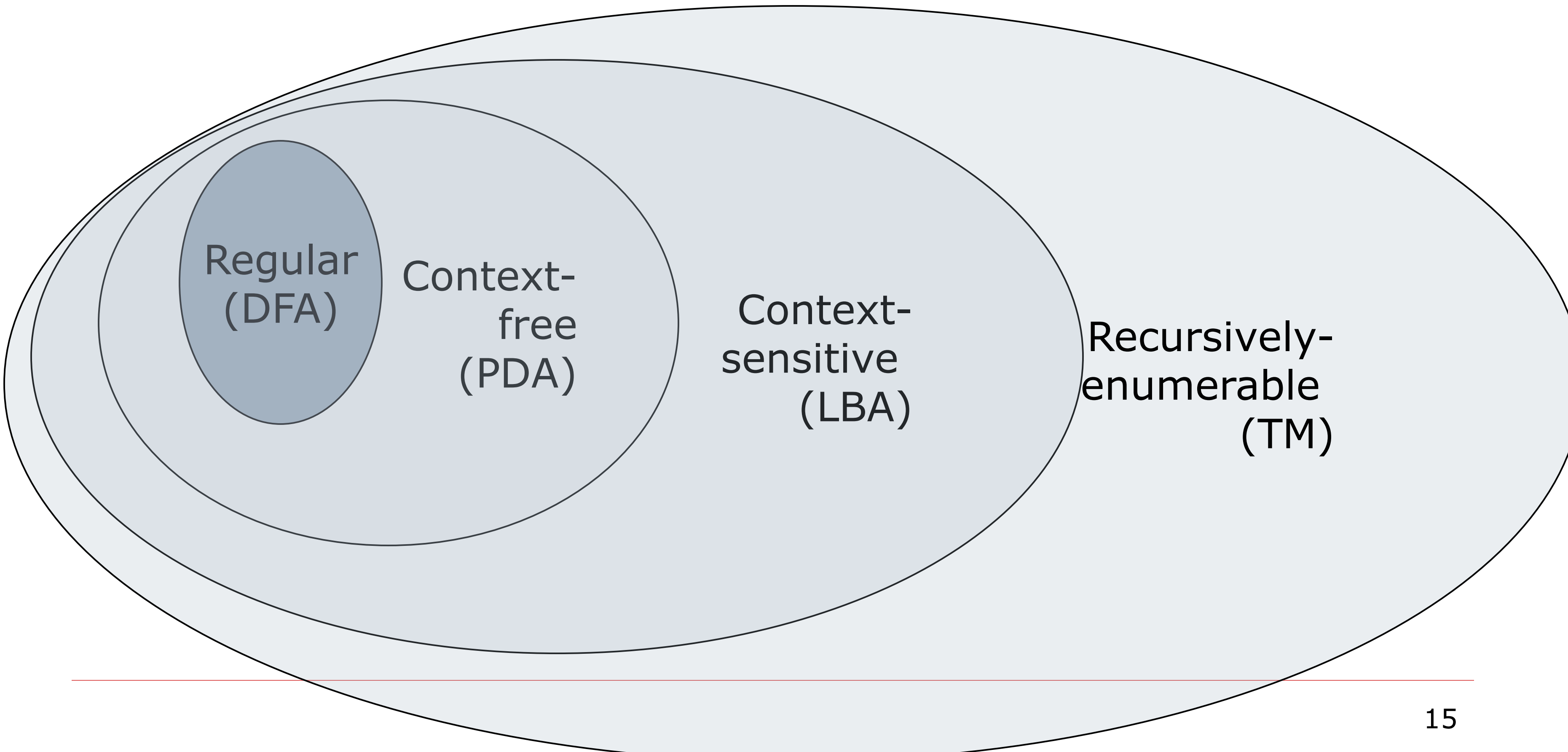
$S \rightarrow 0A$	$B \rightarrow 1B$
$A \rightarrow 1A$	$B \rightarrow 0F$
$A \rightarrow 0B$	$F \rightarrow \varepsilon$

- <u>Languages</u>: "*A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols*"
- <u>Grammars</u>: "*A grammar can be regarded as a device that enumerates the sentences of a language*" - nothing more, nothing less

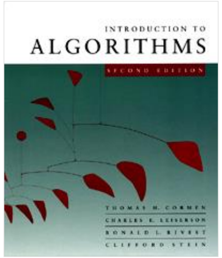
The Chomsky Hierachy



- A containment hierarchy of classes of formal languages



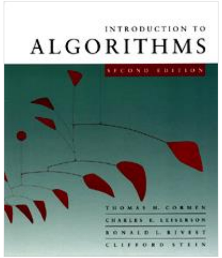
Alphabet



An alphabet is a finite, non-empty set of symbols

- We use the symbol Σ (sigma) to denote an alphabet
- Examples:
  - Binary: Σ = {0,1}
  - All lower case letters: Σ = {a,b,c,..z}
  - Alphanumeric: Σ = {a-z, A-Z, 0-9}
  - DNA molecule letters: Σ = {a,c,g,t}
  - …

Strings



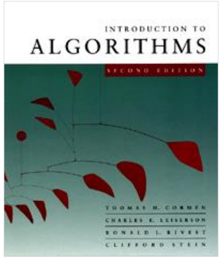
A string or word is a finite sequence of symbols chosen from ∑

- ***Empty string is ε (or "epsilon")***

- Length of a string *w*, denoted by "|*w*|", is equal to the *number of (non- ε) characters in the string*
  - *E.g., x = 010100* *|x| = 6*
  - *x = 01 ε 0 ε 1 ε 00 ε* *|x| = ?*

- *xy* = concatentation of two strings *x* and *y*

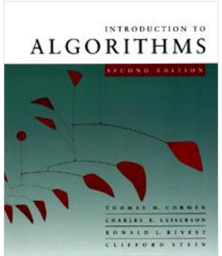
Powers of an alphabet



Let Σ be an alphabet.

- Σ^k = the set of all strings of length k
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

Languages



*L is a said to be a language over alphabet ∑, only if L $\subseteq$ ∑**

➔ this is because ∑* is the set of all strings (of all possible length including 0) over the given alphabet ∑

Examples:

1. Let L be *the* language of all strings consisting of *n* 0's followed by *n* 1's:

   L = {ε, 01, 0011, 000111,…}

2. Let L be *the* language of all strings of with equal number of 0's and 1's:

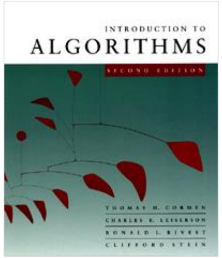
   L = {ε, 01, 10, 0011, 1100, 0101, 1010, 1001,…}

Canonical ordering of strings in the language

Definition: Ø denotes the Empty language

- Let L = {ε}; Is L=Ø? NO

The Membership Problem



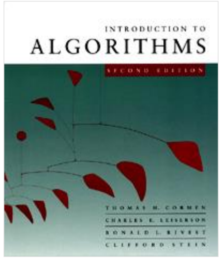
Given a string $w \in \Sigma^*$ and a language L over Σ, decide whether or not $w \in L$.

<u>Example:</u>

Let w = 100011

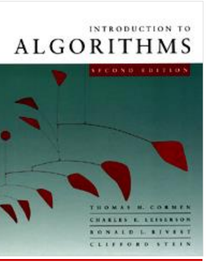
Q) Is w $\in$ the language of strings with equal number of 0s and 1s?

Models



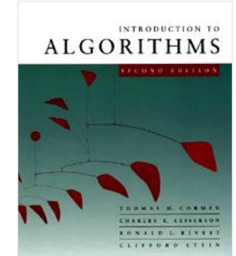
- Finite state automata
- Push down automata
- Linear bounded automata
- Turing Machines

Finite Automata-Applications

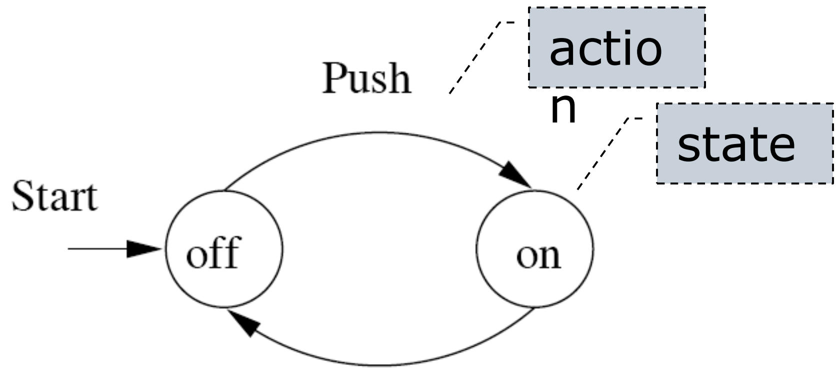


- Some Applications
  - Software for designing and checking the behavior of digital circuits
  - Lexical analyzer of a typical compiler
  - Software for scanning large bodies of text (e.g., web pages) for pattern finding
  - Software for verifying systems of all types that have a finite number of states (e.g., stock market transaction, communication/network protocol)

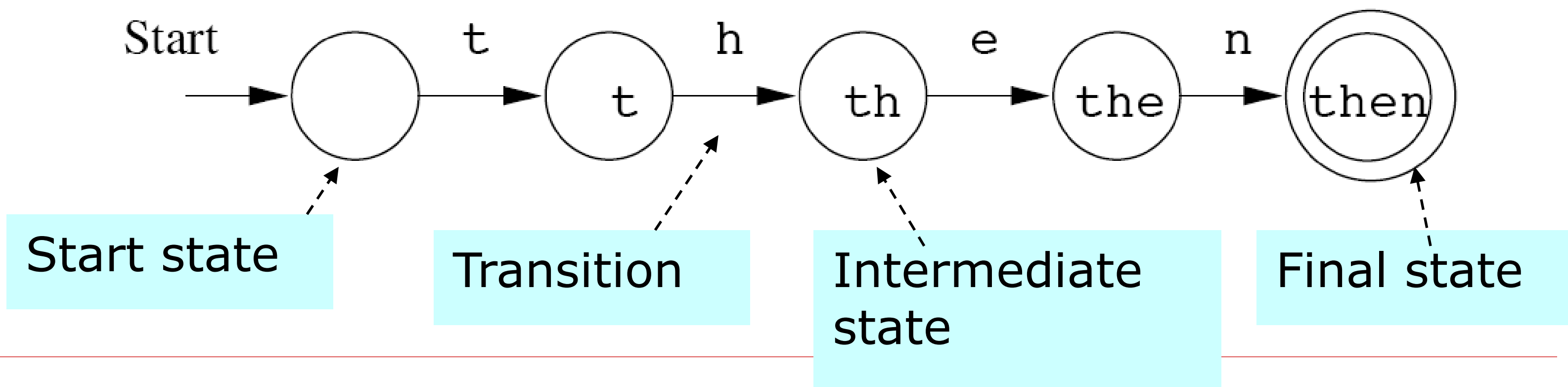
Finite Automata : Examples



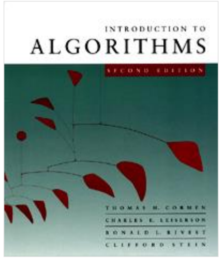
- On/Off switch



- Modeling recognition of the word "*then*"



Structural expressions



- Grammars
- Regular expressions
  - E.g., unix style to capture city names such as “Palo Alto CA”:
    - [A-Z][a-z]*([ ][A-Z][a-z]*)*[ ][A-Z][A-Z]

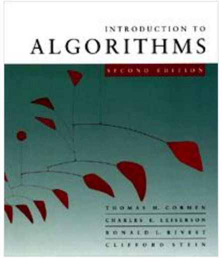
Start with a letter

A string of other letters (possibly empty)

Other space delimited words (part of city name)

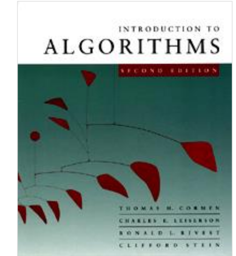
Should end w/ 2-letter state code

Summary



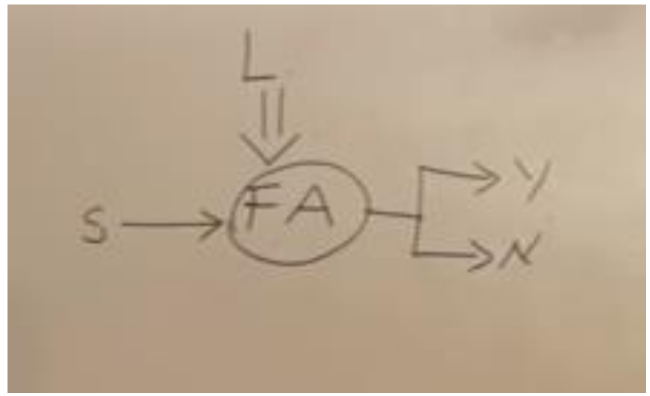
- Automata theory & a historical perspective
- Chomsky hierarchy
- Finite automata
- Alphabets, strings/words/sentences, languages
- Membership problem
- Proofs:
  - Deductive, induction, contrapositive, contradiction, counterexample
  - If and only if

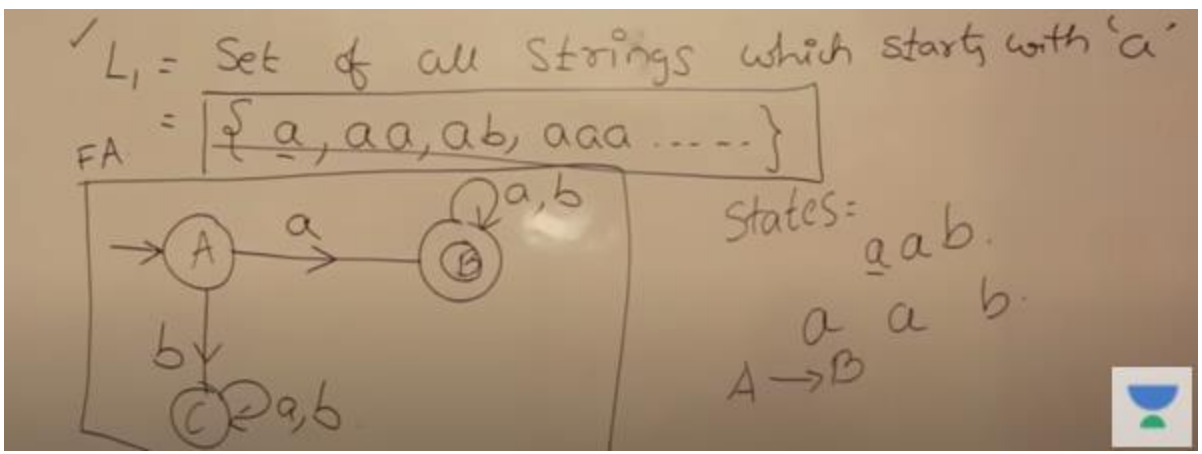
- Read chapter 1 for more examples and exercises

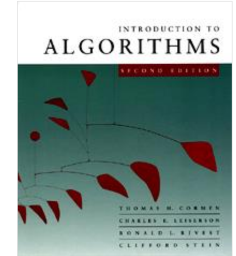


Finite state automata/machine

Example:

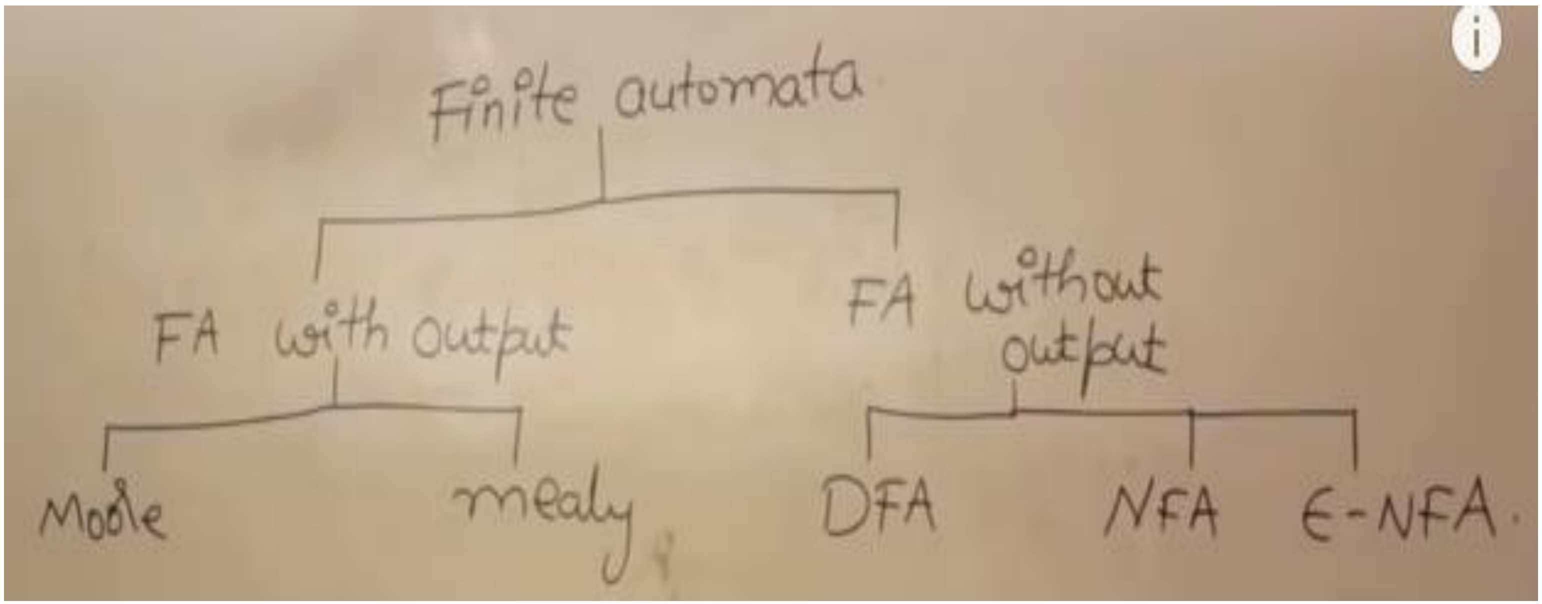




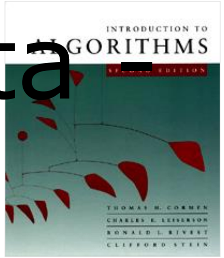


Finite state automata

Types:

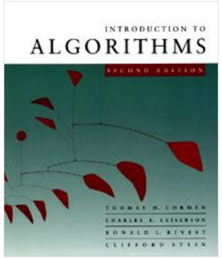


Deterministic Finite Automata – Definition



- A Deterministic Finite Automaton (DFA) consists of:
  - Q ==> a finite set of states
  - Σ ==> a finite set of input symbols (alphabet)
  - q_0 ==> a start state
  - F ==> set of accepting states
  - δ ==> a transition function, which is a mapping between Q x Σ ==> Q
- A DFA is defined by the 5-tuple:
  - $\{Q, \Sigma, q_0, F, \delta\}$

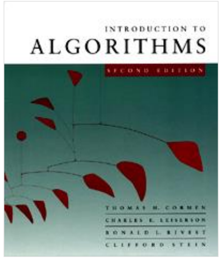
Example:



Construct a DFA, that accepts Set of all Strings over $\{a, b\}$ of length 2

$$\Sigma = \{a, b\}.$$

$$L = \{\underline{aa}, ab, ba, bb\}$$

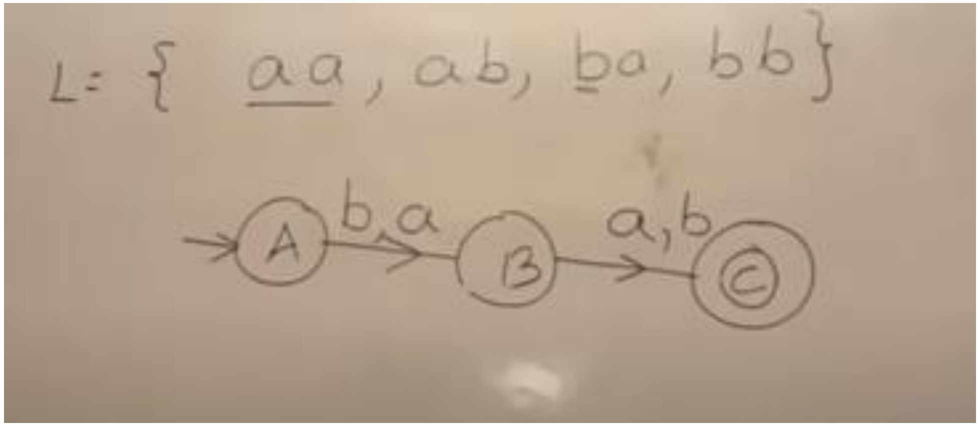


Example:

Construct a DFA, that accepts Set of all Strings over {a, b} of length 2

$\Sigma = \{a, b\}$

$L = \{aa, ab, ba, bb\}$



So the DFA can be specified as ??