Theory of Automata and Formal Languages

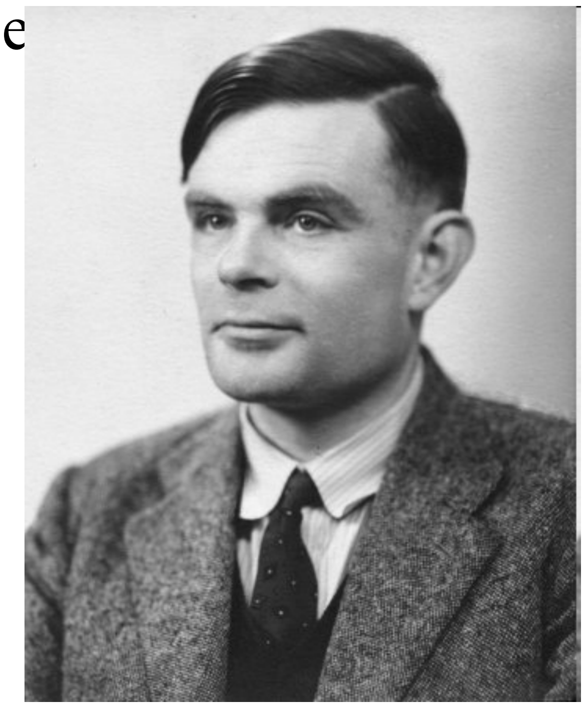
What is Automata Theory?

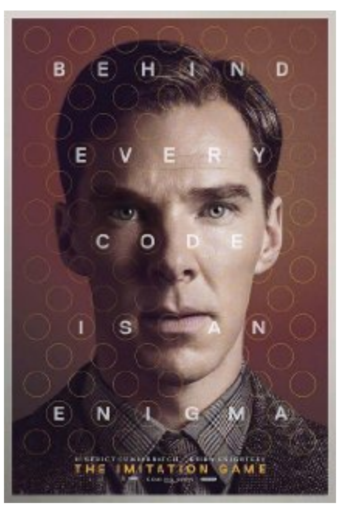
- *Study of abstract (existing in thoughts or as an idea) computing devices, or "machines"*
- Automaton = an abstract computing device
  - Note: A "device" need not even be a physical hardware!
- A fundamental question in computer science:
  - Find out what different models of machines can do and cannot do
  - The *theory of computation*
- Computability vs. Complexity

(A pioneer of automata theory)

Alan Turing (1912-1954)

- Father of Modern Computer Science
- English mathematician
- Studied abstract machines called ***Turing machines*** even before computers existed
- Heard of the Turing test?





Theory of Computation: A Historical Perspective

1930s	• Alan Turing studied 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

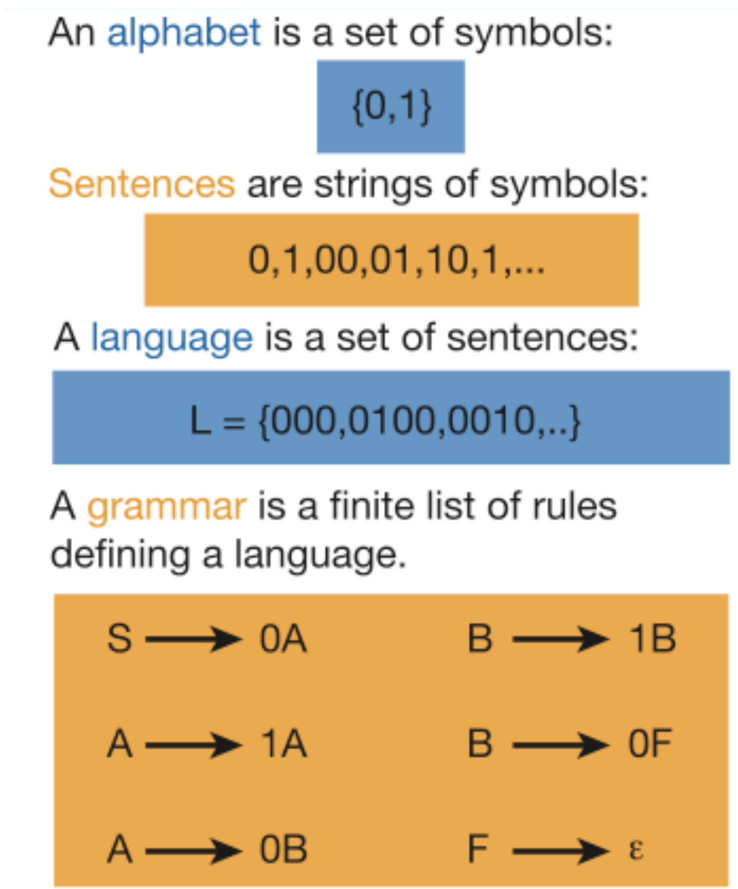
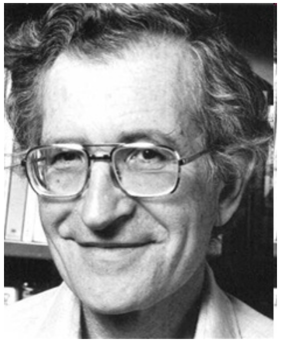


Image source: Nowak et al. Nature, vol 417, 2002

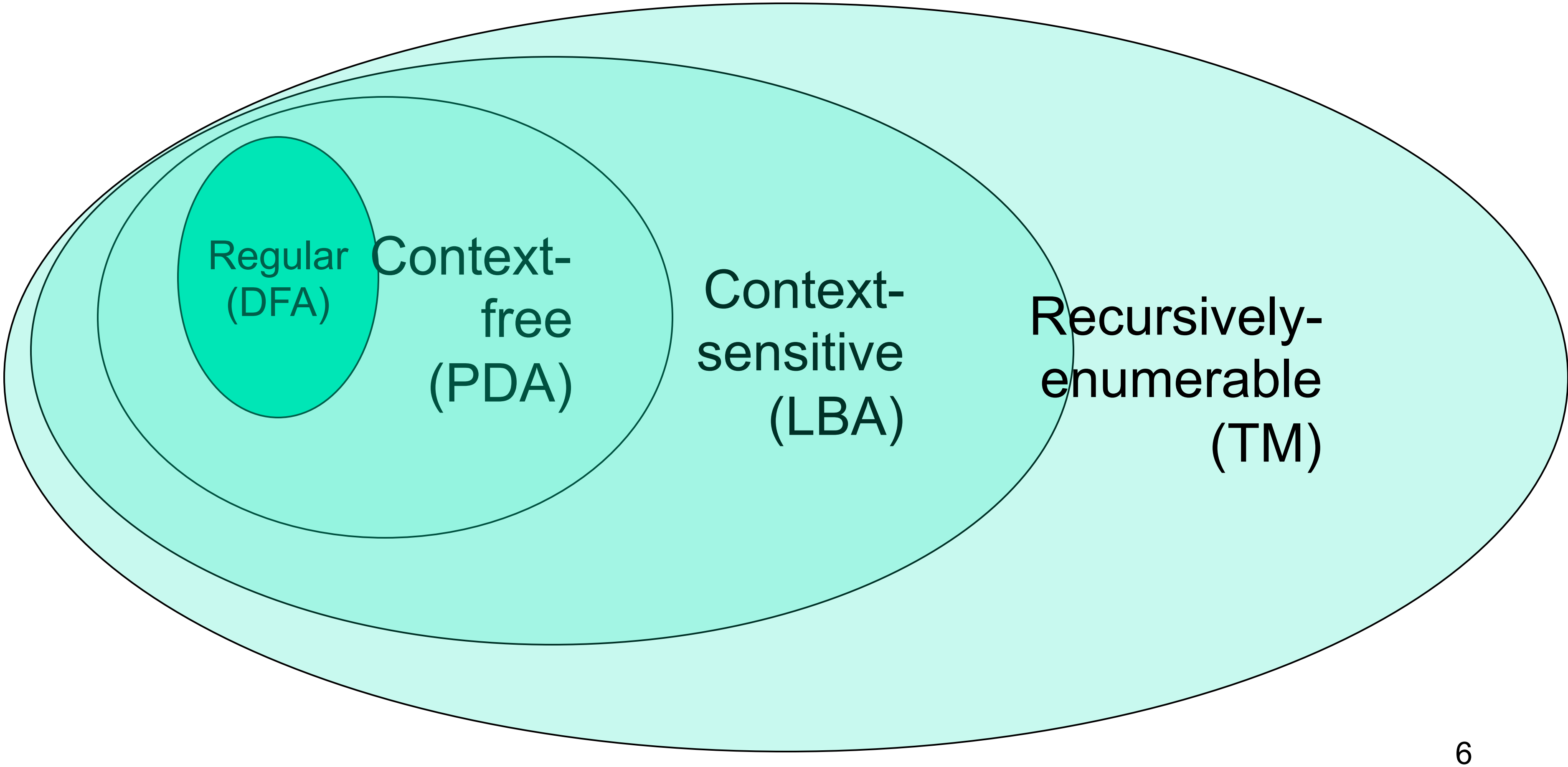
- <u>Languages</u>: "*A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols*"
- <u>Grammars</u>: "*A grammar can be regarded as a device that enumerates the sentences of a language*" - nothing more, nothing less

- *N. Chomsky, Information and Control, Vol 2, 1959*

The Chomsky Hierachy



- A containment hierarchy of classes of formal languages



The Central Concepts of Automata Theory

Alphabet

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

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

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*
  - *y = 1010101* *|x| = ?*

- *xy* = *c*oncatentation of two strings *x* and *y*

Powers of an alphabet

Let $\sum$ be an alphabet.

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

Languages

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

➔ this is because $\sum^*$ is the set of all strings (of all possible length including 0) over the given alphabet $\sum$

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

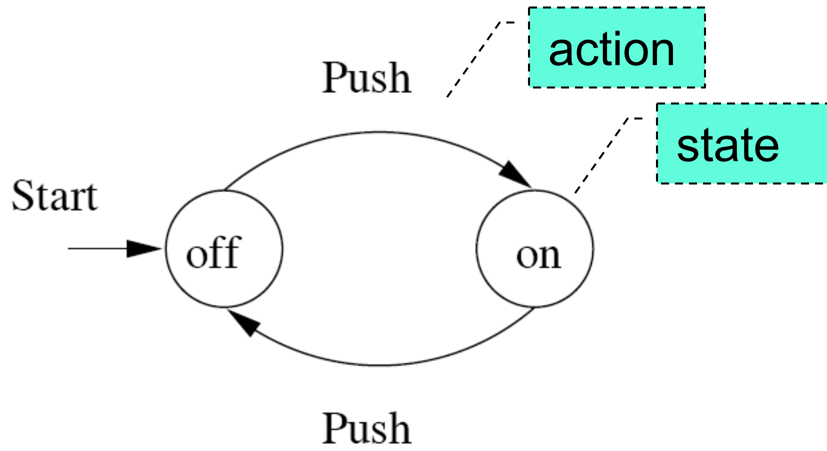
Q) Is w ∈ the language of strings with equal number of 0s and 1s?

Finite Automata

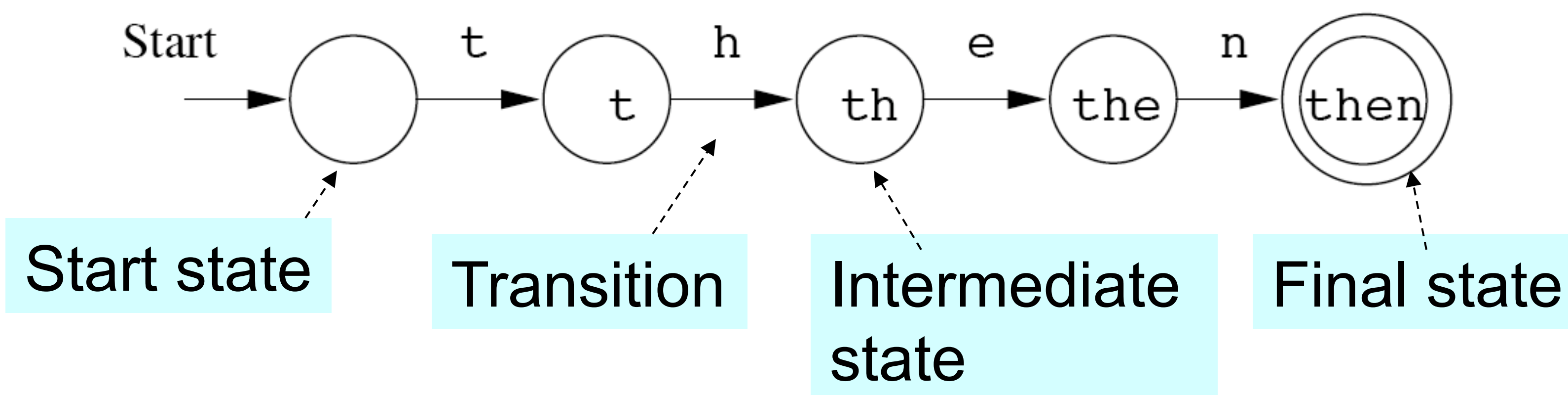
- Some Applications
  - Software for designing and checking the behavior of digital circuits
  - Lexical analyzer of a typical compiler
  - Software for scanning large bodies of text (e.g., web pages) for pattern finding
  - Software for verifying systems of all types that have a finite number of states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

Formal Proofs

Deductive Proofs

- A **deductive proof** consists of a sequence of statement whose truth leads us from some *initial statement* (hypothesis or given statements) to a *conclusion statement*.
- Each step of a deductive proof MUST follow from a given fact or previous statements (or their combinations) by an accepted **logical principle**.
- The theorem that is proved when we go from a hypothesis H to a conclusion C is the statement **''if H then C''.** We say that C is deduced from H.

Deductive Proofs

Example: Proof of a Theorem

- Assume that the following theorem (initial statement) is given:
  - Given Thm. (initial statement): **If $x \geq 4$, then $2^x \geq x^2$**
  - We are not going to prove this theorem, we assume that it is true.
    - If we want we can prove this theorem using proof by induction.

- **Theorem to be proved**:

If x is the sum of the squares of four positive integers, then $2^x \geq x^2$

Hypothesis: x is the sum of the squares of four positive integers

Conclusion: $2^x \geq x^2$

Deductive Proofs

Example: Proof of a Theorem

Proof of

If x is the sum of the squares of four positive integers, then $2^x \geq x^2$

Statement	Justification
1. If $x \geq 4$, then $2^x \geq x^2$	Given theorem
2. $x = a^2 + b^2 + c^2 + d^2$	Given
3. $a \geq 1$ $b \geq 1$ $c \geq 1$ $d \geq 1$	Given
4. $a^2 \geq 1$ $b^2 \geq 1$ $c^2 \geq 1$ $d^2 \geq 1$	From (3) and principle of arithmetic
5. $x \geq 4$	From (2), (4) and principle of arithmetic
6. $2^x \geq x^2$	From (1) and (5)

Read it by yourself

Second Induction Example

- If $x \geq 4$ then $2^x \geq x^2$
- Basis: If x=4, then 2^x is 16 and x^2 is 16. Thus, the theorem holds.
- Induction: Suppose for some $x \geq 4$ that $2^x \geq x^2$. With this statement as the hypothesis, we need to prove the same statement, with x+1 in place of x: $2^{(x+1)} \geq (x+1)^2$

Second Induction Example

- $2^{(x+1)} \geq (x+1)^2$? (i)
- Rewrite in terms of S(x)
  - $2^{(x+1)} = 2*2^x$
  - We are assuming $2^x \geq x^2$
  - So therefore $2^{(x+1)} = 2*2^x \geq 2x^2$ (ii)
- Substitute (ii) into (i)
  - $2x^2 \geq (x+1)^2$
  - $2x^2 \geq (x^2+2x+1)$
  - $x^2 \geq 2x+1$
  - $x \geq 2 + 1/x$
  - Since x >=4, we get some value >=4 on the left side. The right side will equal at most 2.25 and in fact gets smaller and approaches 2 as x increases. Consequently, we have proven the theorem to be true by induction.

On Theorems, Lemmas and Corollaries

We typically refer to:

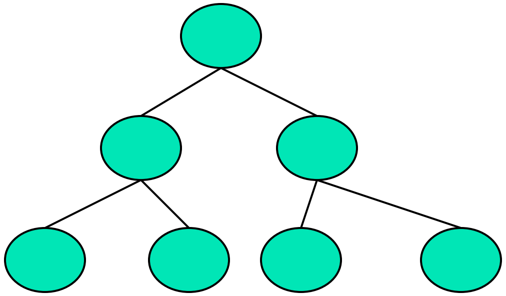
- A major result as a “**theorem**”
- An intermediate result that we show to prove a larger result as a “**lemma**”
- A result that follows from an already proven result as a “**corollary**”

An example:

Theorem: *The height of an n-node binary tree is at least floor(lg n)*

Lemma: *Level i of a perfect binary tree has 2^i nodes.*

Corollary: *A perfect binary tree of height h has $2^{h+1}-1$ nodes.*



Quantifiers

"For all" or "*For every*"

- Universal proofs
- Notation= $\forall$

"There exists"

- Used in existential proofs
- Notation= $\exists$

Implication is denoted by =>

- E.g., "IF A THEN B" can also be written as "A=>B"

Proving techniques

- By contradiction
  - Start with the statement contradictory to the given statement
  - E.g., To prove (A => B), we start with:
    - (A and ~B)
    - … and then show that could never happen

- By induction
  - (3 steps) Basis, inductive hypothesis, inductive step
- By contrapositive statement
  - If *A* then *B* $\equiv$ If ~*B* then ~*A*

Proving techniques…

- By counter-example
  - Show an example that disproves the claim

- Note: There is no such thing called a "proof by example"!
  - So when asked to prove a claim, an example that satisfied that claim is *not* a proof

Different ways of saying the same thing

- "*If* H *then* C":
  1. H *implies* C
  2. *H => C*
  3. C *if* H
  4. H *only if* C
  5. *Whenever* H *holds*, C *follows*

Proof of an iff Theorem

Let x be a real number. Then $\lfloor x \rfloor = \lceil x \rceil$ if and only if x is an integer.

If-Part:

- Given that x is an integer.
- By definitions of ceiling and floor operations. $\lfloor x \rfloor = x$ and $\lceil x \rceil = x$
- Thus, $\lfloor x \rfloor = \lceil x \rceil$.

Only-If-Part:

- Given that $\lfloor x \rfloor = \lceil x \rceil$
- By definitions of ceiling and floor operations. $\lfloor x \rfloor \leq x$ and $\lceil x \rceil \geq x$
- Since given that $\lfloor x \rfloor = \lceil x \rceil$, $\lceil x \rceil \leq x$ and $\lceil x \rceil \geq x$
- By the properties of arithmetic inequalities, $\lceil x \rceil = x$
- Since $\lceil x \rceil$ is always an integer, x MUST be integer too. □

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