COMSCI 33 (AUTOMATA THEORY and FORMAL LANGUAGES)

Lesson 1: Introduction to formal languages

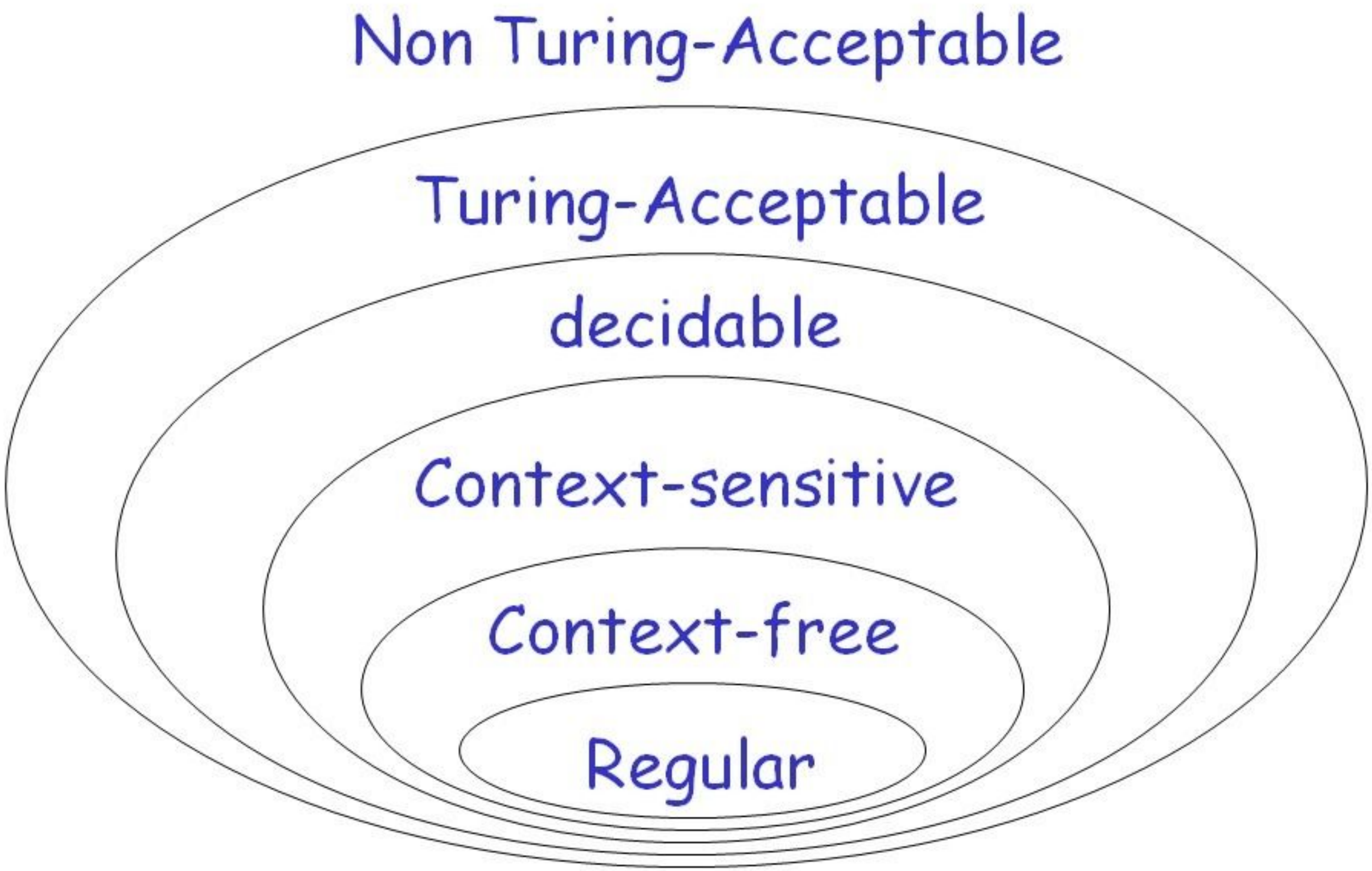
Automata Theory

- In theoretical computer science, automata theory is the study of **abstract computing devices or "machines"** and the problems which they are able to solve.
- These abstract machines are called automata.

Formal Languages

- These are languages with precise **syntax and semantics**
  - Formal languages are defined by two sets of rules:
    - Syntax: precise rules that tell you the symbols you are allowed to use and how to put them together into legal expressions.
  - Semantics: precise rules that tell you the meanings of the symbols and legal expressions
- Programming languages are examples of formal languages

The Chomsky Hierarchy



Course Outline (1/3)

- Formal Languages
  - Symbols
    - Alphabets
    - Strings
    - Formal Languages
    - Regular Expressions
    - Regular Languages

Course Outline (2/3)

- Finite State System
  - Deterministic Finite Automata (DFA)
  - Non-deterministic Finite Automata (NFA)
  - Equivalence of DFA and NFA
  - Properties of Languages accepted by FA
  - Finite Automata and Regular Expressions

Course Outline (3/3)

- Context-free Languages
  - Context-free Grammar (CFG)
    - Regular Languages and Context-free Languages
    - Pushdown Automata
    - Pushdown Automata and CFG

- Turing Machines (TM)
  - Turing Machines
    - Computing with Turing Machines
    - Combining Turing Machines

Grading System (1.0 = 60%)

- Midterm Grade
  - Midterm Exam 35%
  - Quizzes 35%
  - Assignments 15%
  - Seatwork's 15%

- Final Grade
  - Midterm Grade 25%
  - Final Exam 30%
  - Exer/Homeworks 20%
  - Quiz 25%

Books

- Textbook
  - Sipser, Michael(2006). Introduction to the Theory of Computation – Second Edition. Thomson Learning, Inc. Thomson Learning.
- References
  - any automata books/ebooks

Why study automata theory?

- The study of automata is important because
  - Automata theory plays an important role when we make software for designing and checking the behavior of a digital circuit
  - The lexical analysis of a compiler breaks a program into logical units, such as variables, keywords, and punctuation using this mechanism
  - Automata theory works behind software for scanning large bodies of text, such as web pages to find occurrence of words, phrases, etc
  - Automata theory is the most useful concept of software for natural language processing

Basic Concepts

Symbol

- a single character or mark
- an abstract entity with no meaning by itself and often called un-interpreted
- may be letters from various alphabets (like in the english alphabet – a,b,c,d, ..) or digits or special characters

Alphabet

Definition: Any nonempty finite set is called an **alphabet**. Every element of an alphabet Σ is called a **symbol** of Σ.

- is a finite set of symbols
- often represented by the greek letter sigma (Σ) but still you can give any name you want
- for example
  - A = {0, 1}   Boolean alphabet
  - B = {a, b}
  - Σ = {a, b, c, …, z} Latin alphabet

String or Word

Definition: Let Σ be an alphabet. A **word** over Σ is any finite sequence of symbols of Σ. The **empty word** λ (or e) is the only word without any symbol.

- concatenation of 0 or more symbols from an alphabet
- Examples:
  - A = {0, 1}

    e, 0, 1, 01, 10, 11, 10101, … *are words over the alphabet {0,1}*
  - B = {a, b, c}

    e, a, b, c, ab, ac, abc, cab, aacabb, …
  - Σ = {a, b, c, …, z}

    e, a, b, c, d, e, abc, abcde, chi, chu, chuchi… *are words over the latin alphabet*
- Note:
  - e means the empty string or string with no symbols

Length

Definition: The **length of a word** w over Σ, denoted by |w|, is the number of symbols in w.

- is the number of symbols in the string
- |w| = the length of string w
- for example:

  |0| = 1

  |abcd| = 4

  |e| = 0

Powers of an alphabet

- Σ^k is the set of strings of length k, each of whose symbols is in Σ.
- If $\Sigma=\{0,1\}$
  - $\Sigma^0 = \{e\}$
  - $\Sigma^1 = \{0,1\}$
  - $\Sigma^2 = \{00,01,10,11\}$
  - $\Sigma^3 = \{000,001,010,011,100,101,110,111\}$

Definition: The set of all words over Σ is denoted by Σ^*. Then, $\Sigma^+ = \Sigma^* - \{\lambda\}$, to be the set of words without the empty word

- Example:
  - $\{0,1\}^* = \{e,0,1,00,01,10,11,000, . . .\}$
  - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
- The set of nonempty strings from alphabet Σ is denoted Σ^+
  - $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$
  - $\Sigma^* = \Sigma^+ \cup \{e\}$

Concatenation

- The concatenation of two string **x** and **y** is the string **x** followed by the string **y**, that is **xy**

- **Example:**

  u= 0110 and v=10110.

  Then

  uv=011010110

  while

  vu=101100110

Substring or Subword

Definition: Let u, w be element of Σ^*.

–u is a **subword** of w iff there are x,y in Σ^* such that w=xuy.

where:

–y is the **suffix** of w. if $y \neq \lambda$, it is **proper suffix.**

–x is a **prefix** of w. if $x \neq \lambda$, it is **proper prefix.**

- Exercise:

  Count the number of subwords of the word

  abbcbbab

Reversal

- the reversal of a string w, denoted by w^R is the string spelled backwards
- Example
  - w = abbab
  - w^R = babba

Formal definition of Reversal

- Formally,
  - if **|w|=**0, then $\mathbf{w=w^R=e}$
    - if the length of the string **w** is 0 then string **w** is equal to its reverse as well as to the empty string.
  - if **|w|=n+1**, then **w=ua** for some **a ε** Σ and $\mathbf{w^R=au^R}$

- Example
  - Show the reverse of **w = aabab**

Σ^*

- Σ^* is the set of all strings over the alphabet Σ, including **e**.

- Example
  - 1. Σ **=** {a}

    Σ^* **=**{e, a, aa, aaa, …}
  - 2. Σ **=** {0,1}

    Σ^* **=**{e, 0, 1, 00,01,10,11, 000, …}

A (formal) Language is …

- A set of strings from an alphabet. The set may be empty, finite or infinite.
- Any subset of Σ^*

Examples

- The languages of all strings consisting of n 0's followed by n 1's for some n>=0

  {e,01,0011,000111, …}

- The set of strings of 0's and 1's with an equal number of each

  {e, 01, 10, 0011, 0101, 1001, …}

- The set of binary numbers whose value is prime

  {10, 11, 101, 111, 1011, … }

- Some special languages:

{} The empty set/language, containing no strings

{ε} A language containing one string, the empty string.

Set-Former as way to define languages

- It is common to describe a language using a “set-former”
  - { w / something about w}
- This expression is read “the set of words w such that whatever is said about w to the right of the vertical bar”
- Examples:
  - { w / w consists of an equal number of 0’s and 1’s}
  - {w / w is a binary integer that is prime}

Example Languages

- **L1** = {x, xx, xxx, xxxx, …}

  L1 = {w ∈ {x}* / $w=x^n$ for n=1,2,3, …}

- **L2** = {x, xxx, xxxxx, xxxxx, … }

  L2 = {w∈ {x}* / $w=x^{odd}$}

  L2 = {w∈ {x}* / x^{2n+1} for n=0, 1, 2, 3, …}

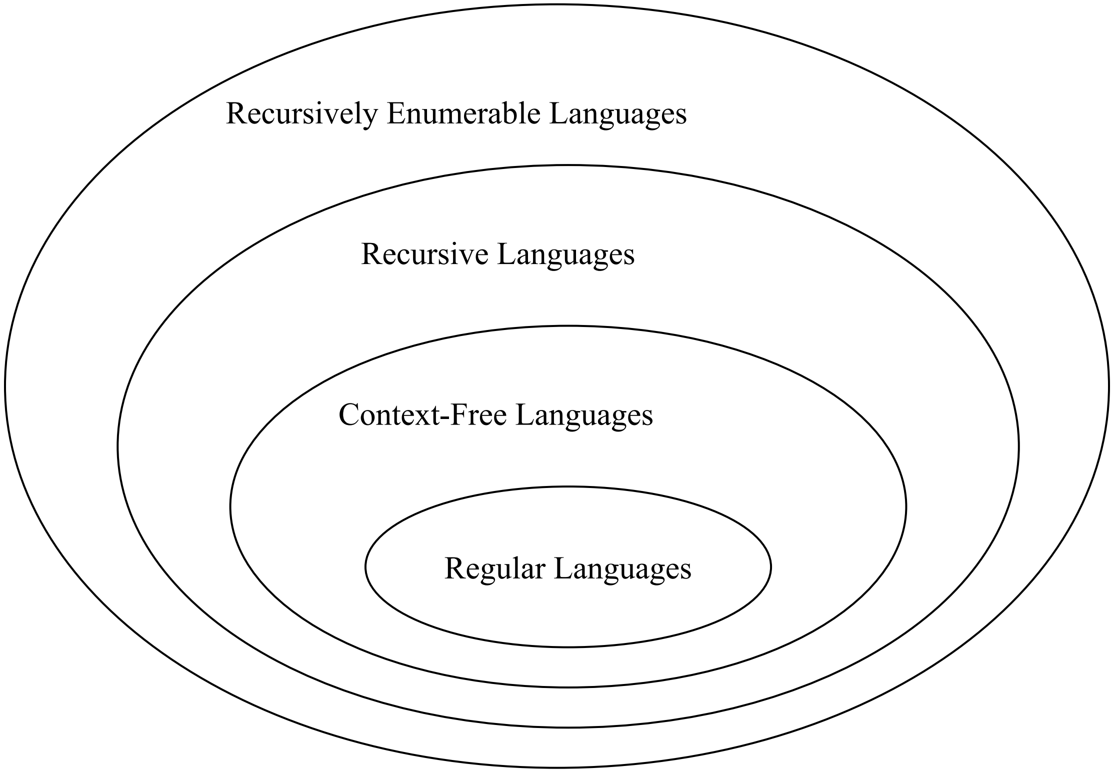
- **L3**= {aa, bb, ab, ba}

- **L4**= {b, ba, baa, baaa, …}

- **L5**= {w ∈ {a,b}* / $w=w^R$}

Hierarchy of languages

Non-Recursively Enumerable Languages



Operations on Languages

- **Concatenation**

  L1. L2 = L1L2

  = { w ∈ Σ^* / w=xy for some x∈**L1** and y∈**L2** }

- **Kleene Closure**

  L* = {w ∈ $\boldsymbol{\Sigma}^*$ / w=$w^0 w^1 w^2 \ldots w^k$,

  where w^0, w^1, w^2, … w^k **∈ L** }

Kleene Star *

- The **Kleene closure** (*) is defined as the concatenation of none, one, two, or any countable number strings it applies to. The notation is sometimes known as the **Kleene star**.
- Example:

  L1 = { a }

  L1* = { **e**, a, aa, aaa, … }

Set operations on languages

- Since languages are set, they can be combined with the operations: union, intersection and difference.
  - **Union** $L = L1 \cup L2$
  - **Intersection** $L = L1 \cap L2$
  - **Difference** $L = L1 - L2$
  - **Complement** $\acute{L} = \Sigma^* - L$

Examples

- L1 = {a}   L2 = {bb}   L3 = {a,bb,aa}
- Solve for:
  - Union
  - Intersection
  - Difference
  - complement