Theory of Automata
Preliminaries

Week 1

Contents

Languages

- In English, we distinguish 3 different entities: letters, words, and sentences.
  – Groups of letters make up words and groups of words make up sentences.
  – However, not all collections of letters form valid words, and not all collections of words form valid sentences.

- This situation also exists with computer languages.
  – Certain (but not all) strings of characters are recognizable words (e.g., IF, ELSE, FOR, WHILE …); and certain (but not all) strings of words are recognizable commands.

- To construct a general theory of **formal languages**, we need to have a definition of a **language structure**, in which the decision of whether a given string of units constitutes a valid larger unit is not a matter of guesswork, but is based on explicitly stated rules.
- In this model, language will be considered as symbols with formal rules, and not as expressions of ideas in the minds of humans.
- *The term **"formal"** emphasizes that it is the **form** of the string of symbols that we are interested in, not the meaning.*

Basic Definitions

- A finite non-empty set of symbols (letters), is called an alphabet. It is denoted by Σ (Greek letter sigma).

Example:

Σ={a,b}

Σ={0,1} //important as this is the language //which the computer understands.

Σ={i,j,k}

Strings

What is a String?

Concatenation of finite symbols from the alphabet is called a string.

- Example:

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

  a, abab, aaabb, ababababababababab

Words

- What is a 'word'?
- Words are strings belonging to some language.

Example:

If $\Sigma = \{x\}$ then a language L can be defined as

$L=\{x^n : n=1,2,3,.....\}$ or $L=\{x,xx,xxx,....\}$

Here x,xx,... are the words of L

- All words are strings, but not all strings are words.

EMPTY STRING or NULL STRING

- We shall allow a string to have no letters. We call this **empty string** or **null string**, and denote it by the symbol **Λ**.
- For all languages, the **null word**, if it is a word in the language, is the word that has no letters. We also denote the null word by **Λ**.
- Two words are considered the same if all their letters are the same and in the same order.
- *For clarity, we usually do not allow the symbol* ***Λ*** *to be part of the alphabet of any language.*

Discussion of null

- The language that has no words is denoted by the standard symbol for null set, ø.
- It is not true that **Λ** is a word in the language ø since this language has no words at all.
- If a certain language L does not contain the word **Λ** and we wish to add it to L, we use the operation "+" to form L + {**Λ**}. This language is **not** the same as L.
- However, the language L + ø is the same as L since no new words have been added.

Introduction to Defining Languages

- The rules for defining a language can be of two kinds:
  - They can tell us how to test if a string of alphabet letters is a valid word, or
  - They can tell us how to construct all the words in the language by some clear procedures.

Defining Languages

- **Example:** Consider this alphabet with only one letter

$$\sum = \{ x \}$$

- We can define a language by saying that any nonempty string of alphabet letters is a word

$$L_1 = \{ x, xx, xxx, xxxx, \ldots \} \text{ or}$$

$$L_1 = \{ x^n \text{ for } n = 1, 2, 3, \ldots \}$$

Note that because of the way we have defined it, the language L_1 does not include the null word **Λ**.

Example:

The language L of strings of odd length, defined over Σ={a}, can be written as

L={a, aaa, aaaaa,…..}

- Example:

  The language L of strings that does not start with a, defined over Σ={a,b,c}, can be written as

  L={b, c, ba, bb, bc, ca, cb, cc, …}

Example:

The language L of strings of length 2, defined over Σ={0,1,2}, can be written as

L={00, 01, 02,10, 11,12,20,21,22}

- Example:

  The language L of strings ending in 0, defined over Σ ={0,1}, can be written as

  L={0,00,10,000,010,100,110,…}

Example:

- The language **EQUAL**, of strings with number of a’s equal to number of b’s, defined over Σ={a,b}, can be written as

  {Λ ,ab,aabb,abab,baba,abba,…}

- The language **EVEN-EVEN**, of strings with even number of a’s and even number of b’s, defined over Σ={a,b}, can be written as

  {Λ, aa, bb, aaaa,aabb,abab, abba, baab, baba, bbaa, bbbb,…}

Concatenation

- Let us define an operation, **concatenation**, in which two strings are written down side by side to form a new longer string.

  xxx concatenated with xx is the word xxxxx

  x^n concatenated with x^m is the word x^{n+m}

- For convenience, we may label a word in a given language by a new symbol. For example,

  xxx is called a, and xx is called b

- Then to denote the word formed by concatenating a and b, we can write

  ab = xxxxx

- It is not true that when two words are concatenated, they produce another word. For example, if the language is

  L_2 = {x, xxx, xxxxx, …} = {x^{2n+1} for n = 0, 1, 2, …}

  then a = xxx and b = xxxxx are both words in L_2, but their concatenation ab = xxxxxxxx is not in L_2

Concatenation makes new Words?

- Note that in this simple example, we have:

  ab = ba

  But in general, this relationship does NOT hold for all languages (e.g., **houseboat** and **boathouse** are two different words in English).

- **Example:** Consider another language by beginning with the alphabet

  $$\Sigma = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$$

  Define the language

  L_3 = { any finite string of alphabet letters that does not start with the letter zero }

- This language L_3 looks like the set of positive integers:

$$L_3 = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, \dots \}$$

- If we want to define L3 so that it includes the string (word) 0, we could say

L_3 = { any finite string of alphabet letters that, if it starts with a 0, has no more letters after the first}

Definition: Length

- We define the function **length** of a string to be the number of letters in the string.
- Example:
  - If a = xxxx in the language L_1, then length(a) = 4
  - If c = 428 in the language L_3, then length(c) = 3
  - If d = 0 in the language L_3, then length(d) = 1
  - In any language that includes the null word **Λ**, then length(**Λ**) = 0
- For any word w in any language, if length(w) = 0 then w = **Λ**.

- Recall that the language L_1 does not contain the null string **Λ**. Let us define a language like L_1 but that does contain **Λ**:

$$L_4 = \{ \mathbf{\Lambda}, x, xx, xxx, xxxx, \ldots \}$$
$$= \{ x^n \text{ for } n = 0, 1, 2, 3, \ldots \}$$

- Here we have defined that

$$x^0 = \mathbf{\Lambda} \text{ (NOT } x^0 = 1 \text{ as in algebra)}$$

- In this way, x^n always means the string of n alphabet letters x's.
- *Remember that even* **Λ** *is a word in the language, it is not a letter in the alphabet.*

Definition: Reverse

- If a is a word in some language L, then **reverse**(a) is the same string of letters spelled backward, even if this backward string is not a word in L.
- Example:
  - reverse(xxx) = xxx
  - reverse(145) = 541
  - Note that 140 is a word in L_3, but reverse(140) = 041 is NOT a word in L_3

Definition: Palindrome

- Let us define a new language called Palindrome over the alphabet

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

PALINDROME = { **Λ**, and all strings x such that reverse(x) = x }

- If we want to list the elements in PALINDROME, we find

PALINDROME = { **Λ**, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, abba, ... }

Palindrome

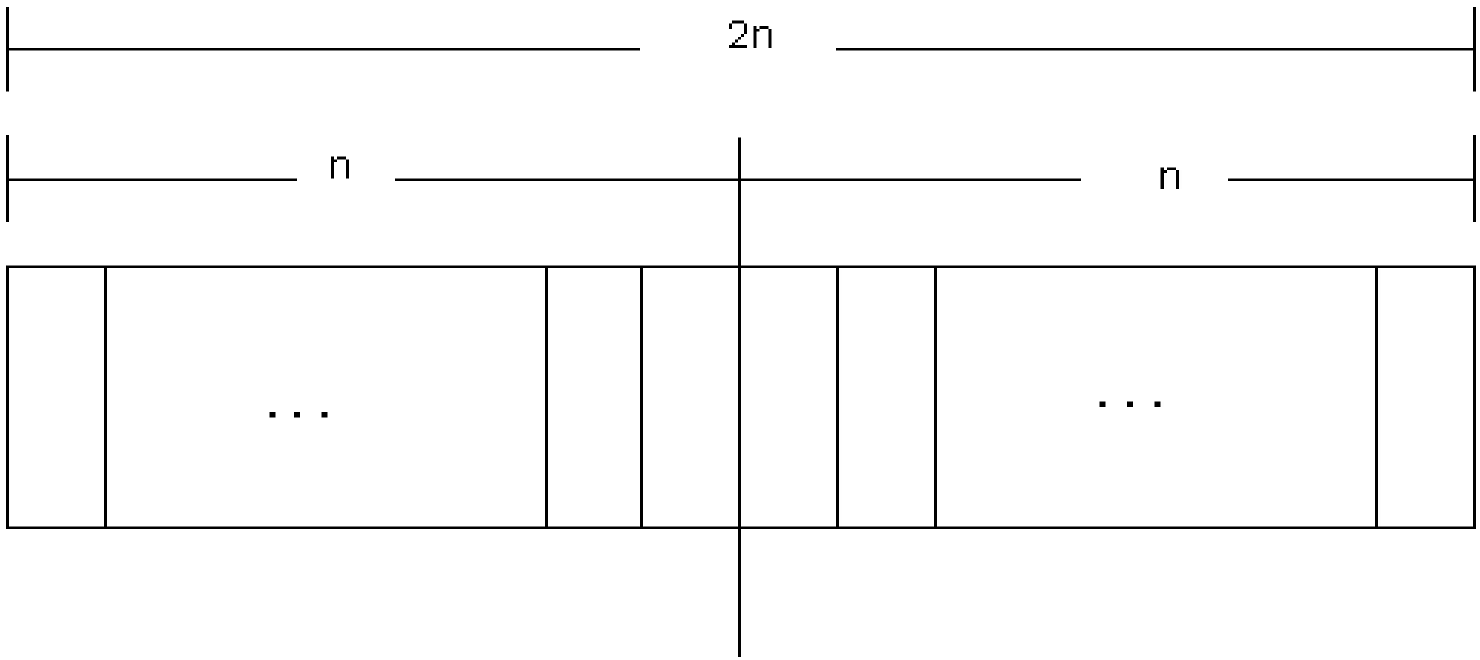
- Sometimes two words in PALINDROME when concatenated will produce a word in PALINDROME
  - **abba** concatenated with **abbaabba** gives **abbaabbaabba** (in PALINDROME)
- But more often, the concatenation is not a word in PALINDROME
  - **aa** concatenated with **aba** gives **aaaba** (NOT in PALINDROME)
- The language PALINDROME has interesting properties that we shall examine later.

Task

- Prove that there are as many palindromes of length 2n, defined over $\Sigma = \{a,b,c\}$, as there are of length 2n-1, n = 1,2,3… . Determine the number of palindromes of length 2n defined over the same alphabet as well.

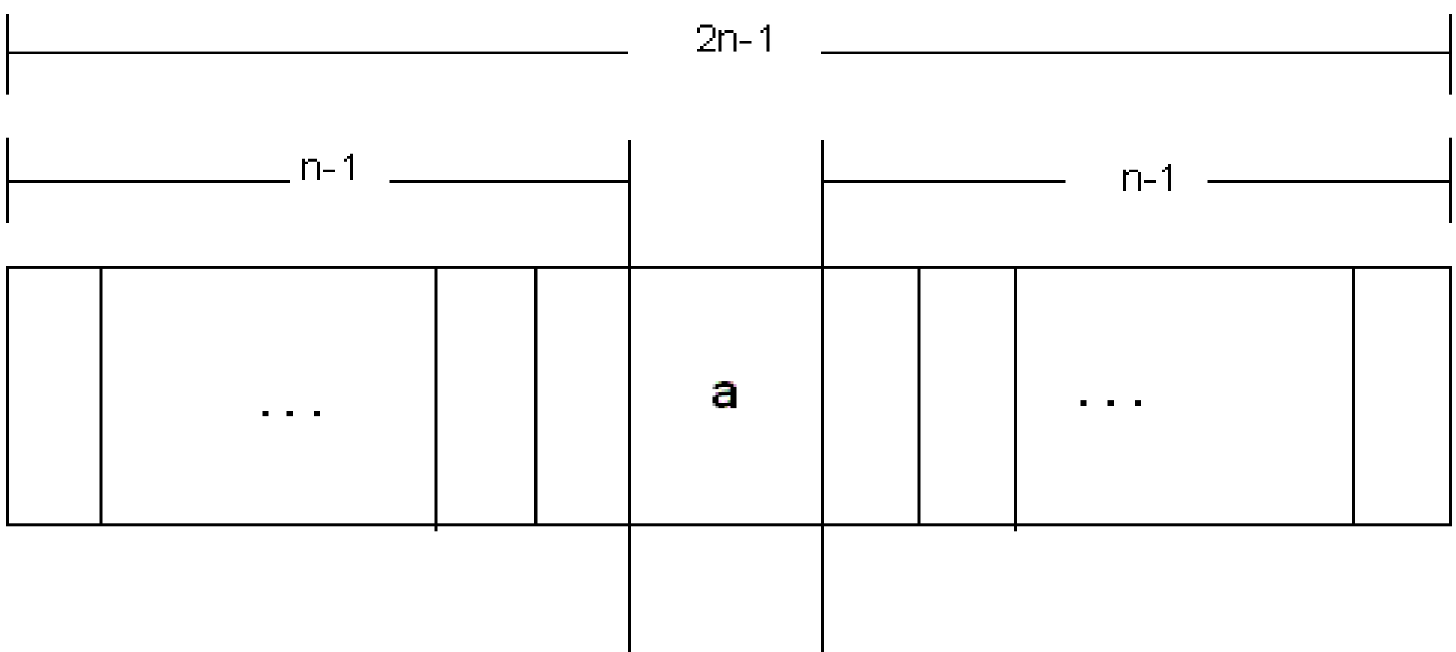
Solution

- To calculate the number of palindromes of length(2n), consider the following diagram,



- which shows that there are as many palindromes of length 2n as there are the strings of length n *i.e.* the required number of palindromes are 3^n (as there are three letters in the given alphabet, so the number of strings of length n will be 3^n).

- To calculate the number of palindromes of length (2n-1) with a as the middle letter, consider the following diagram,



which shows that there are as many palindromes of length 2n-1, with a as middle letter, as there are the strings of length n-1, *i.e.* the required number of palindromes are 3^{n-1}.

Similarly the number of palindromes of length 2n-1, with b or c as middle letter, will be 3^{n-1} as well. Hence the total number of palindromes of length 2n-1 will be

$$3^{n-1} + 3^{n-1} + 3^{n-1} = 3\,(3^{n-1}) = 3^n\,.$$

Kleene Closure

- **Definition:** Given an alphabet ∑, we define a *language* in which any string of letters from ∑ is a word, even the null string **Λ**. We call this *language* the **closure** of the alphabet ∑, and denote this language by ∑*.
- Examples:

If ∑ = { x } then ∑* = { **Λ**, x, xx, xxx, … }

If ∑ = { 0, 1 } then ∑* = { **Λ**, 0, 1, 00, 01, 10, 11, 000, 001, … }

If ∑ = { a, b, c } then

∑* = { **Λ**, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, … }

Lexicographic order

- Notice that we listed the words in a language in size order (i.e., words of shortest length first), and then listed all the words of the same length alphabetically.
- This ordering is called **lexicographic** order, which we will usually follow.
- The star in the closure notation is known as the **Kleene star**.
- We can think of the Kleene star as an **operation** that makes, out of an alphabet, an *infinite* language (i.e., *infinitely many* words, each of *finite* length).

Kleene Closure

- Let us now generalize the use of the Kleene star operator to sets of words, not just sets of alphabet letters.

- **Definition:** If S is a set of words, then S* is the set of all finite strings formed by concatenating words from S, where any word may be used as often as we like, and where the null string **Λ** is also included.

Kleene Closure

- Example: If S = { aa, b }

Kleene Closure

- Example: If S = { aa, b } then
- S* = { **Λ** plus any word composed of factors of aa and b }, or
- S* = { Λ plus any strings of a's and b's in which the a's occur in **even** clumps }, or
- S* = { Λ, b, aa, bb, aab, baa, bbb, aaaa, aabb, baab, bbaa, bbbb, aaaab, aabaa, aabbb, baaaa, baabb, bbaab, bbbaa, bbbbb, … }
- Note that the string aabaaab is not in S* because it has a clump of a's of length 3.

Kleene Closure

- Example: Let S = { a, ab }

Kleene Closure

- Example: Let S = { a, ab }. Then
- S* = { **Λ** plus any word composed of factors of a and ab },
- Or
- S* = { **Λ** plus all strings of a's and b's except those that start with b and those that contain a double b },
- Or
- S* = { **Λ**, a, aa, ab, aaa, aab, aba, aaaa, aaab, abaa, abab, aaaaa, aaaab, aaaba, aabaa, aabab, abaaa, abaab, ababa, … }
- Note that for each word in S*, every b must have an *a* immediately to its left, so the double b, that is bb, is not possible; neither any string starting with b.

How to prove a certain word is in the closure language S*

- We must show how it can be written as a concatenation of words from the base set S.

- In the previous example, to show that abaab is in S*, we can factor it as follows:

  abaab = (ab)(a)(ab)
- These three factors are all in the set S, therefore their concatenation is in S*.
- Note that the parentheses, ( ), are used for the sole purpose of demarcating the ends of factors.

- Observe that if the alphabet has no letters, then its closure is the language with the null string as its only word; that is

  if $\sum = \emptyset$ (the empty set), then $\sum^* = \{ \mathbf{\Lambda} \}$

- Also, observe that if the set S has the null string as its only word, then the closure language S* also has the null string as its only word; that is

  if $S = \{ \mathbf{\Lambda} \}$, then $S^* = \{ \mathbf{\Lambda} \}$

  because $\mathbf{\Lambda\Lambda} = \mathbf{\Lambda}$.

- *Hence, the Kleene closure always produces an infinite language unless the underlying set is one of the two cases above.*

Kleene Closure of different sets

- The Kleene closure of two different sets can end up being the same language.
- Example: Consider two sets of words

  S = { a , b, ab } and T = { a, b, bb }

Then, both S* and T* are languages of all strings of a's and b's since any string of a's and b's can be factored into syllables of (a) or (b), both of which are in S and T.

Positive Closure

- If we wish to modify the concept of closure to refer only the concatenation of **some** (**not zero**) strings from a set S, we use the notation + instead of *.
- This "plus operation" is called **positive closure**.
- Example: if $\sum = \{ x \}$ then $\sum^+ = \{ x, xx, xxx, \dots \}$
- Observe that:
- If S is a language that **does not** contain **Λ**, then S^+ is the language S^* without the null word **Λ**.

1. If S is a language that **does** contain **Λ**, then $S^+ = S^*$
2. Likewise, if $\sum$ is an alphabet, then $\sum^+$ is $\sum^*$ without the word **Λ**.

S**?

- What happens if we apply the closure operator twice?
  - We start with a set of words S and form its closure S*
  - We then start with the set S* and try to form its closure, which we denote as (S*)* or S**
- **Theorem 1:**

  For any set S of strings, we have S* = S**
- Before we prove the theorem, recall from Set Theory that
  - A = B if A is a subset of B **and** B is a subset of A
  - A is a subset of B if for all x in A, x is also in B

Proof of Theorem 1:

- Let us first prove that S** is a subset of S*:

  Every word in S** is made up of factors from S*. Every factor from S* is made up of factors from S. Hence, every word from S** is made up of factors from S. Therefore, every word in S** is also a word in S*. This implies that S** is a subset of S*.

- Let us now prove that S* is a subset of S**:

  In general, it is true that for any set A, we have a subset A*, because in A* we can choose as a word any factor from A. So if we consider A to be our set S* then S* is a subset of S**

- Together, these two inclusions prove that S* = S**.