Theory of Computational

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

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

Example: If Σ= {a,b} then a, abab, aaabb, abababababababab

Words: Words are strings belonging to some language.

Example: If Σ= {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 ∑ = { x }

We can define a language by saying that any nonempty string of alphabet letters is a word L_1 = { x, xx, xxx, xxxx, … } or L_1 = { x^n for n = 1, 2, 3, … }

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,…}

Example: 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

∑ = { 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, … }

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 = { **Λ**, x, xx, xxx, xxxx, … } = { x^n for n = 0, 1, 2, 3, … }

Here we have defined that: x^0 = **Λ** (NOT x^0 = 1 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 ∑ = { 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)

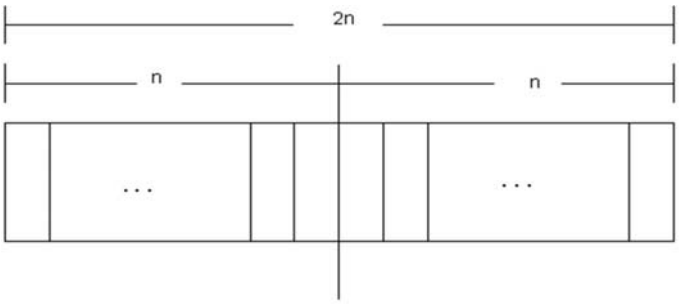
The language PALINDROME has interesting properties that we shall examine later.

Task

Q) 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.

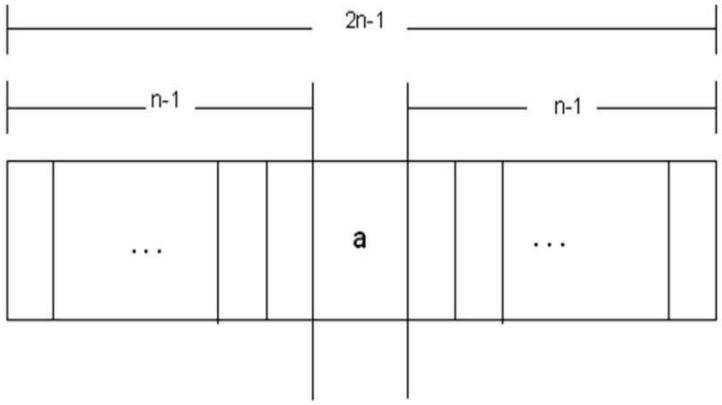
Solution

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