Precedence Matrices and words over some ordered alphabet

R. Stella Maragatham[1], V. Nithya Vani*[2]
[1,2] Department of Mathematics, Queen Mary's College, Chennai 600004 India,
* Ph.D Research Scholar, Department of Mathematics, Queen Mary's College, Chennai

Abstract *- A word, mathematically expressed, is a sequence of symbols in a finite set, called an alphabet. Parikh matrix is an ingenious tool providing information on certain subsequences of a word, referred to as subwords. On the other hand, based on subwords of a word, the notion of Precedence matrix or p-matrix of a word over an alphabet has been introduced by A. Cerny (2009) in studying a property, known as fair words and it is closely related to Parikh matrix. In this paper we consider Precedence matrix for words especially over binary, ternary and tertiary alphabets and develop algorithm to display of the Precedence matrices of words.*

Key words *- Ternary word, Tertiary word, subword, Precedence matrix.*

I. INTRODUCTION

The theory of formal languages [5] is one of the fundamental areas of theoretical computer science. Combinatorics on words [6] is one of the topics of study and research (see, for example, [7, 8]) in the theory of formal languages but is a comparatively new area of research in Discrete Mathematics, with applications in many fields. The concept of Parikh vector [5], which gives counts of the symbols in a word, has been an important notion in the theory of formal languages. Extending this concept Mateescu et al. [9] introduced the notion of Parikh matrix of a word which gives numerical information about certain subwords of the word, including the information given by the Parikh vector of the word. A. Cerny [1] introduced another notion called precedence matrix or p-matrix of a word which is motivated by the notion of a fair word. The main diagonal of this matrix is the number of alphabets and the entries above the main diagonal provide information on the number of certain sub-words in w and every entry below the main diagonal has the value of reverse the subwords of above the main diagonal. In particular, since the structure of words is not reflected by using the conventional matrix calculus, as it is the case in Parikh matrices.

Here we consider Precedence matrix and developing the algorithm for display Precedence matrix of words over binary, ternary and tertiary alphabet.

II. PRELIMINARIES

A word is a finite sequence of symbols taken from a finite set called an alphabet. For example the word *abaabb* is over the binary alphabet {*a, b*}. An ordered alphabet is an alphabet with an ordering on its elements, denoted by the symbol <. For example, the ternary alphabet {*a, b, c*} with an ordering *a<b<c* is an ordered alphabet, denoted as {*a<b<c*}. For a word *w*, the mirror image or reversal of $w=a_1a_2\cdots a_{n-1}a_n$, $n\geq 1$, is the word $mi(w)=a_n\,a_{n-1}\cdots a_2a_1$ where each a_i is a symbol in an alphabet. A subword *u* of a given word *w* is a subsequence of *w*. We denote the number of such subwords *u* in a given word *w* by $|w|_u$. For example, if the word is *w =abaabb* over {*a<b*}, the number of subwords *ab* in *w* is $|w|_{ab} = 7$. The Parikh vector [5] of a word *w* gives the number of occurrences of each of the symbols in the word. For example, (6, 7, 10) is the Parikh vector of the word *babcbaacb* over the ternary alphabet {*a, b, c*}. An extension of the notion of Parikh vector is the *Parikh matrix* [9] of a word. For a word *w* over an ordered alphabet Σ, the Parikh matrix *M(w)* of *w* is a triangular matrix, with 1′*s* on the main diagonal and 0′*s* below it but the entries above the main diagonal provide information on the number of certain subwords in *w*. For a binary word *u* over the ordered binary alphabet {*a<b*}, the Parikh matrix is

$$M(u)=\begin{pmatrix} 1 & |u|_a & |u|_{ab} \\ 0 & 1 & |u|_b \\ 0 & 0 & 1 \end{pmatrix}.$$

The notion of precedence matrix or p-matrix of a word over an alphabet has been introduced in [1]. Given the square matrices A, B of the same order and with integer entries, the matrix $A\circ B$ is defined as follows: the $(i,j)^{th}$ entry of $A\circ B$ is given by

$$(A\circ B)_{ij}=\begin{cases} A_{ii}+B_{ii} & \text{if } i=j \\ A_{ij}+B_{ij}+A_{ii}B_{jj} & \text{if } i\neq j \end{cases},$$

where A_{ij}, B_{ij} are the $(i,j)^{th}$ entries of A, B respectively.

1) *Definition*: Let $\Sigma=\{a_1, a_2, \cdots, a_k\}$ be an alphabet. For a symbol $a_s \in \Sigma$ for 1 ≤ s ≤ k, let E_{a_s} be the k×k matrix defined as $\left(E_{a_s}\right)_{i,j}=1, \text{if } i=j=s$ and $\left(E_{a_s}\right)_{i,j}=0,$ otherwise. The precedence morphism (or) p-morphism on Σ is the morphism φ_k given by $\varphi_k(a_s) = E_{a_s}$. For a word $w = a_{i1}a_{i2} \ldots a_{im}$, $a_{ij} \in \Sigma$ for 1 ≤ j ≤ m, we have $\varphi_k(w) = \varphi_k(a_{i1}) \circ \varphi_k(a_{i2}) \circ \ldots \circ \varphi_k(a_{im})$. In other words $\varphi_k(w)$ is computed by the operation ∘ on matrices as defined earlier. The resulting matrix $\varphi_k(w)$ is called the precedence matrix or p-matrix of *w*.

As an illustration, let $\Sigma = \{a, b, c\}$, so that k = 3. Then

$$\varphi_3(a)=\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}, \varphi_3(b)=\begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix} \text{ and}$$

$$\varphi_3(c)=\begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix}$$

$$\varphi_3(abcb) = \begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix} \circ \begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix} \circ \begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix} \circ \begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}$$

$$= \begin{pmatrix}1&2&1\\0&2&1\\0&1&1\end{pmatrix}.$$

In fact the p-matrix φ_3 for a ternary word w over {*a*, *b*, *c*} is given by

$$\varphi_3(w) = \begin{pmatrix}|w|_a & |w|_{ab} & |w|_{ac}\\ |w|_{ba} & |w|_b & |w|_{bc}\\ |w|_{ca} & |w|_{cb} & |w|_c\end{pmatrix}$$

while the p-matrix $\varphi_2(w)$ for a binary word w over {*a*, *b*} is given by

$$\varphi_2(w) = \begin{pmatrix}|w|_a & |w|_{ab}\\ |w|_{ba} & |w|_b\end{pmatrix}.$$

Where $|w|_a$ is the number of a in w, $|w|_b$ is the number of b in w, $|w|_c$ is the number of c in w, $|w|_{ab}$ is the number of ab in w, similarly $|w|_{bc}$, $|w|_{ac}$, $|w|_{ba}$, $|w|_{cb}$ and $|w|_{ca}$ are the number of bc, ac, ba, cb and ac in w respectively.

2) *Theorem*: Let $\Sigma = \{a_1, a_2, \ldots, a_k\}$ be an ordered alphabet. For each word w over Σ with Precedence matrix $\varphi(w)$, we have,

$$(A \circ B)_{ij} = \begin{cases} A_{ii} + B_{ii} & \text{if } i = j \\ A_{ij} + B_{ij} + A_{ii}B_{jj} & \text{if } i \neq j \end{cases}$$

$$\varphi(w)_{i,j} = |w|_{a_i}, \quad i = j;$$

and $\varphi(w)_{i,j} = |w|_{a_i a_j}, \quad i \neq j;$

where $\varphi(w)_{i,j}$ is the $(i,j)^{th}$ entry of the matrix $\varphi(w)$.

III. ALGORITHM TO DISPLAY PRECEDENCE MATRIX CORRESPONDING TO A WORD

There are many methods by which we can form the Precedence matrix of binary, ternary and tertiary words, for example Precedence matrix product and use of various tools of computing Precedence matrix product etc. Here an algorithm for finding Precedence matrix of a binary, ternary and tertiary word is introduced. With the help of this algorithm Precedence matrix of a binary, ternary and tertiary word, however large it may be can be found out.

A. Algorithm for binary word of Precedence matrix

The following pseudo code gives instantly the precedence matrix of a binary sequence.

01 Initialize a word = 'w'

02 Set len = length of w

03 For $i = 0$ to lendo

04 Calculate total number of a, ab in w

05 Calculate total number of b, ba in w

06 End

// Create a matrix (a_{ij}) of order $M(=2)$

07 For $i = 0$ to M do

08 For $j = 0$ to M do

09 If $(i = j)$

10 If $(i = 0 \,\&\, j = 0)$

11 a_{ij} = total number of 'a'

12 If $(i = 1 \,\&\, j = 1)$

13 a_{ij} = total number of 'b'

14 else if $(i < j)$

15 If $(i = 0 \,\&\, j = 1)$

16 a_{ij} = total number of 'ab'

17 else if $(i > j)$

18 If $(i = 1 \,\&\, j = 0)$

19 a_{ij} = total number of 'ba'

20 End

21 End

22 End

B. Application of binary word Algorithm

1) Example

The binary word $\xi_1 = abab\underbrace{a \ldots a}_{10}\underbrace{b \ldots b}_{20}$ has the Precedence matrix

$$\varphi_{M_2}(\xi_1) = \begin{pmatrix} 12 & 242 \\ 21 & 22 \end{pmatrix}$$

2) Example

The binary word $\xi_1 = \underbrace{b \ldots b}_{15} abababab\underbrace{a \ldots a}_{10}$ has the Precedence matrix

$$\varphi_{M_2}(\xi_1) = \begin{pmatrix} 14 & 10 \\ 250 & 19 \end{pmatrix}$$

C. Algorithm for ternary word of Precedence matrix

The following pseudo code gives instantly the precedence matrix of a ternary sequence.

01 Initialize a word = 'w'

02 Set len = length of w

03 For $i = 0$ to lendo

04 Calculate total number of a, ab, ac in w

05 Calculate total number of b, ba, bc in w

06 Calculate total number of c, ca, cb in w

07 End

// Create a matrix (a_{ij}) of order $M(=3)$

08 For $i = 0$ to M do

09 For $j = 0$ to M do

10 If $(i = j)$

11 If $(i = 0 \,\&\, j = 0)$

12 a_{ij} = total number of 'a'

13 If $(i = 1 \,\&\, j = 1)$

14 a_{ij} = total number of 'b'

15 If $(i = 2 \,\&\, j = 2)$

16 a_{ij} = total number of 'c'

17 else if $(i < j)$

18 If $(i = 0 \,\&\, j = 1)$

19 a_{ij} = total number of $'ab'$

20 If $(i = 0 \,\&\, j = 2)$

21 a_{ij} = total number of $'ac'$

22 If $(i = 1 \,\&\, j = 2)$

23 a_{ij} = total number of $'bc'$

24 else if $(i > j)$

25 If $(i = 1 \,\&\, j = 0)$

26 a_{ij} = total number of $'ba'$

27 If $(i = 2 \,\&\, j = 0)$

28 a_{ij} = total number of $'ca'$

29 If $(i = 2 \,\&\, j = 1)$

30 a_{ij} = total number of $'cb'$

31 End

32 End

33 End

D. Application of ternary word Algorithm

1) Example

The ternary word $\xi_1 = abcabc\underbrace{a \ldots a}_{10}\underbrace{b \ldots b}_{15}\underbrace{c \ldots c}_{5}$ has the Precedence matrix

$$\varphi_{M_3}(\xi_2) = \begin{pmatrix} 12 & 183 & 83 \\ 21 & 17 & 88 \\ 21 & 31 & 7 \end{pmatrix}$$

2) Example

The ternary word $\xi_1 = \underbrace{b \ldots b}_{15} abcabcabcabc\underbrace{c \ldots c}_{5}\underbrace{a \ldots a}_{10}$ has the Precedence matrix

$$\varphi_{M_3}(\xi_2) = \begin{pmatrix} 14 & 10 & 83 \\ 21 & 19 & 88 \\ 21 & 31 & 14 \end{pmatrix}$$

E. Algorithm for tertiary word of Precedence matrix

The following pseudo code gives instantly the precedence matrix of a tertiary sequence.

01 Initialize a word = $'w'$

02 Set len = length of w

03 For $i = 0$ to lendo

04 Calculate total number of a, ab, ac, ad in w

05 Calculate total number of b, ba, bc, bd in w

06 Calculate total number of c, ca, cb, cd in w

07 Calculate total number of d, da, db, dc in w

08 End

// Create a matrix (a_{ij}) of order $M(= 4)$

09 For $i = 0$ to M do

10 For $j = 0$ to M do

11 If $(i = j)$

12 If $(i = 0 \,\&\, j = 0)$

13 a_{ij} = total number of $'a'$

14 If $(i = 1 \,\&\, j = 1)$

15 a_{ij} = total number of $'b'$

16 If $(i = 2 \,\&\, j = 2)$

17 a_{ij} = total number of $'c'$

18 If $(i = 3 \,\&\, j = 3)$

19 a_{ij} = total number of $'d'$

20 else if $(i < j)$

21 If $(i = 0 \,\&\, j = 1)$

22 a_{ij} = total number of $'ab'$

23 else if $(i = 0 \,\&\, j = 2)$

24 a_{ij} = total number of 'ac'

25 else if $(i = 0 \,\&\, j = 3)$

26 a_{ij} = total number of 'ad'

27 else if $(i = 1 \,\&\, j = 2)$

28 a_{ij} = total number of 'bc'

29 else if $(i = 1 \,\&\, j = 3)$

30 a_{ij} = total number of 'bd'

31 else if $(i = 2 \,\&\, j = 3)$

32 a_{ij} = total number of 'cd'

33 else if $(i > j)$

34 If $(i = 1 \,\&\, j = 0)$

35 a_{ij} = total number of 'ba'

36 If $(i = 2 \,\&\, j = 0)$

37 a_{ij} = total number of 'ca'

38 If $(i = 2 \,\&\, j = 1)$

39 a_{ij} = total number of 'cb'

40 If $(i = 3 \,\&\, j = 0)$

41 a_{ij} = total number of 'da'

42 If $(i = 3 \,\&\, j = 1)$

43 a_{ij} = total number of 'db'

44 If $(i = 3 \,\&\, j = 2)$

45 a_{ij} = total number of 'dc'

46 End

47 End

48 End

F. Application of tertiary word Algorithm

1) Example: The tertiary word

$\xi_3 = abcdabcd\underbrace{a\ldots a}_{10}\underbrace{b\ldots b}_{15}\underbrace{c\ldots c}_{5}\underbrace{d\ldots d}_{5}$ has the Precedence matrix

$$\varphi_{M_4}(\xi_3) = \begin{pmatrix} 12 & 183 & 83 & 83 \\ 21 & 17 & 88 & 88 \\ 21 & 31 & 7 & 38 \\ 21 & 31 & 11 & 7 \end{pmatrix}$$

2) Example: The tertiary word

$\xi_3 = \underbrace{b\ldots b}_{15}\underbrace{d\ldots d}_{5}abcdabcdabcdabcd\underbrace{c\ldots c}_{5}\underbrace{a\ldots a}_{10}$ has the Precedence matrix

$$\varphi_{M_4}(\xi_3) = \begin{pmatrix} 14 & 10 & 83 & 83 \\ 21 & 19 & 88 & 88 \\ 21 & 31 & 9 & 38 \\ 21 & 31 & 11 & 9 \end{pmatrix}$$

F. Algorithm for k-letter words of Precedence matrix

The following pseudo code gives instantly the precedence matrix of a k-sequence.

01 Initialize a word = 'w'

02 Set k = length of w

03 For $i = 0$ to k do

04 Calculate total number of $a_i, a_i a_{i+1}, \ldots, a_i a_{i+k}$ in w

05 Calculate total number of $a_i a_{i-1}, \ldots, a_i a_{i-k}$ in w

06 End

// Create a matrix (a_{ij}) of order $M(= k)$

07 For $i = 0$ to k do

08 For $j = 0$ to k do

09 If $(i = j)$

10 a_{ij} = total number of 'a_i'

11 else if $(i < j)$

12 a_{ij} = total number of '$a_i a_j$'

13 else if $(i > j)$

14 a_{ij} = total number of $'a_j a_i'$

15 End

16 End

IV. CONCLUSION

Here we consider a Precedence matrix similar to Parikh Matrix. In this paper we have given algorithm for finding the Precedence matrix corresponding to a word. This helps us to find Precedence matrix of binary, ternary and tertiary word big or small instantly.

V. REFERENCES

[1] A. Cerny, "On fair words" Journal of Automata, Languages and Combinatorics., 14 (2), pp 163-174, 2009.

[2] A. Bhattacharjee, B. S. Purkayastha, "Some alternative ways to find M-ambiguous binary words corresponding to a Parikh Matrix", Int. J. of Computer Applications, Vol 4 No.1 pp 53–64, 2014.

[3] A. Bhattacharjee, B. S. Purkayastha, "Parikh Matrices and Words over Ternary Ordered Alphabet" P

roceedings of fourth International Conference on Soft computing for Problem Solving, Vol 1(AISC, Vol 335) pp135–145, 2014.

[4] A. Bhattacharjee, B. S. Purkayastha, "Parikh Matrices and Words over Tertiary Ordered Alphabet", Int. J. of Computer Applications, Vol 85, No.4, pp10–15, 2014.

[5] A. Mateescu, A. Salomaa, K. Salomaa and S. Yu, A Sharpening of the Parikh Mapping, *RAIRO-Theoretical Informatics and Applications*, 35, 551—564, 2001.

[6] M. Lothaire, *Combinatorics on Words*, Cambridge Mathematical Library, Cambridge university Press, 1997.

[7] L. Kari and K. Mahalingam, Involutively bordered words, *International Journal of Foundations of Computer Science*, 18, 1089-1106, 2007.

[8] C. A. Priya Darshini, V. R. Dare, I. Venkat and K.G. Subramanian, Factors of words under an involution, *Journal of Mathematics and Informatics*, 1 (2013-14) 52-59.

[9] A. Mateescu and A. Salomaa, Matrix indicators for subword occurrences and ambiguity, *International Journal of Foundations of Computer Science*, 17, 277-292, 2004.

[10] K. Mahalingam and K. G. Subramanian, Product of Parikh Matrices and Commutativity, *International Journal of Foundations of Computer Science*, 23, 207-223, 2012.

VI. APPENDIX

In this section we are given Programme for algorithm for ternary word of Precedence matrix in Java coding

```
publicclass TEST1 {

	publicstaticvoid main(String[] args) {

	String input = "abcabbaccac";
	charinputData[] = input.toCharArray();
	intlength = inputData.length;
	intaCount=0;
	intbCount=0;
	intcCount=0;
	//To count a,b,c
	for(inti=0;i<length;i++){

	charc=Character.toLowerCase(inputData[i]);
		switch(c){
			case'a':
			aCount++;
			break;
			case'b':
			bCount++;
			break;

case'c':
			cCount++;
			break;
		}
	}

	charfindMe='b';charfindFromMe='a';intposition=0;
	intabCount=0;
	for(inti=0;i<length;i++){
		if(findFromMe == Character.toLowerCase(inputData[i])){

	for(intj=position;j<length;j++){

	if(findMe==Character.toLowerCase(inputData[j])){

	abCount++;
				}
			}
		}
		position++;
	}

	findMe='c';findFromMe='a';position=0;intacCount=0;
	for(inti=0;i<length; i++){

	if(findFromMe==Character.toLowerCase(inputData[i])){

	for(intj=position;j<length;j++){
```

```
      if(findMe==Character.toLowerCa
se(inputData[j])){

      acCount++;
                        }
                  }
            }
position++;
   }

      findMe='a';findFromMe='b';posi
tion=0;intbaCount=0;
      for(inti=0;i<length;i++){

      if(findFromMe==Character.toLow
erCase(inputData[i])){

      for(intj=position;j<length;j++
){
                        if(findMe ==
Character.toLowerCase(inputData[j]))
{

      baCount++;
                        }
                  }
            }
            position++;
      }

      findMe='c';findFromMe='b';posi
tion=0;intbcCount=0;
      for(inti=0;i<length;i++){

      if(findFromMe==Character.toLow
erCase(inputData[i])){

      for(intj=position;j<length;j++
){
                        if(findMe ==
Character.toLowerCase(inputData[j]))
{

      bcCount++;
                        }
                  }
            }
            position++;
      }

      findMe='a';findFromMe='c';posi
tion=0;intcaCount=0;
      for(inti=0;i<length;i++){

      if(findFromMe==Character.toLow
erCase(inputData[i])){

      for(intj=position;j<length;j++
){
                        if(findMe ==
Character.toLowerCase(inputData[j]))
{

      caCount++;
                        }
                  }
            }
            position++;
      }

      findMe='b';findFromMe='c';posi
tion=0;intcbCount=0;
      for(inti=0;i<length;i++){

      if(findFromMe==Character.toLow
erCase(inputData[i])){

      for(intj=position;j<length;j++
){
                        if(findMe ==
Character.toLowerCase(inputData[j]))
{

      cbCount++;
                        }
                  }
            }
            position++;
      }

      for(inti=0;i<3;i++){
            for(intj=0;j<3;j++){
                  if(i==j){
                        if(j==0)

      System.out.print(aCount+"\t");
                        elseif(j==1)

      System.out.print(bCount+"\t");
                        elseif(j==2)

      System.out.print(cCount+"\t");
                        }elseif
(i==0 &&j==1){

      System.out.print(abCount+"\t")
;
                        }elseif
(i==0 &&j==2){

      System.out.print(acCount+"\n")
;
                        }elseif
(i==1 &&j==0){

      System.out.print(baCount+"\t")
;
                        }elseif
(i==1 &&j==2){
```

```
        System.out.print(bcCount+"\n")
;
                                }elseif
(i==2 &&j==0){

        System.out.print(caCount+"\t")
;
                                }elseif
(i==2 &&j==1){

        System.out.print(cbCount+"\t")
;
                                }
                        }
                }
        }

}
```

OUTPUT

```
4       5       11
7       3       10
5       2       4
```