FACTOR GRAPHS, FAILURE FUNCTIONS AND BI-TREES

R.C. Backhouse and R.K. Lutz
Department of Computer Science
Heriot-Watt University,
Edinburgh EH1 2HJ

Abstract The factors and factor matrix of a regular language are defined and their properties stated. It is shown that the factor matrix of a language Q has a unique starth root - called the factor graph of Q. The Knuth, Morris, Pratt pattern-matching algorithm, its extensions and Weiner's substring identifier algorithm are all shown to correspond to finding the factor graph of some regular language.

Keywords: regular language, factor, pattern-matching, string-matching, substring identifiers, factor graph, factor matrix.

1. Introduction

The remarkable pattern-matching algorithm due to Knuth, Morris and Pratt [7] is well-known and needs no introduction. Much less well-known is an area of automata theory initially developed by Conway [5] - the study of the factors of a regular language. This paper correlates these two in a way which we feel is quite startling. The results we present therefore offer (in our opinion) a significant challenge to automata theorists - to explain the correlation and to exploit it by developing new algorithms for the solution of pattern matching problems.

In section 2 we define the factors of a language and state a number of properties of factors due to Conway [5]. We then prove that the factors of a regular language Q define a (non-deterministic) recogniser of Q which we call the factor graph of Q. Sections 3 and 4 then show that the failure function method of solving the pattern matching problem is equivalent to finding the factor graph of a regular language. Section 4 illustrates that, after a minor modification, Weiner's algorithm [9] is also equivalent to finding the factor graph of a regular language.

We shall assume familiarity of the reader with the terminology of graph theory and language theory. There is a well-known correspondence between labelled p-node graphs and $p\times p$ matrices, and hence we use the terms graph and matrix synonomously. e is used to denote the empty word and V is used to denote a finite alphabet. Following Conway [5] we call a matrix all of whose non-null entries are e a <u>constant matrix</u> and a matrix all of whose entries are subsets $\{a_1,a_2,\ldots,a_k\}$ of V a <u>linear matrix</u>. A <u>constant + linear matrix</u> is, as the terminology suggests, one which is the union of a constant and a linear matrix. A <u>recogniser</u> (G,S,T) consists of a constant + linear matrix G and two subsets S and T of the nodes of G which are designated as start and terminal nodes, respectively. The <u>language recognised</u> by (G,S,T) is $\bigcup_{\substack{s\in S\\ t\in T}} G^*_{st}$. A recogniser is <u>all-admissible</u> if for all nodes x of the graph there is a path in the graph from some start node s to x and a path from x to some terminal node t. Finally if X is a finite set 2^X denotes the set of all subsets of X and $|X|$ denotes the size of X.

2. Factor Theory

2.1 Fundamentals of Factory Theory and the Factor Matrix

The concept of a factor of a regular language was introduced by Conway [5]. Since Conway's work appears not to be well known, this section summarises the fundamental definitions and properties of factors. All proofs can be found in Conway's book.

Definitions Let F, G, H, ... , K, Q denote arbitrary languages F.G...H...J.K is a subfactorization of Q if and only if $F.G...H...J.K \subseteq Q$. (*).

A term H is maximal if no word may be added to H without violating the inequality (*). A factorization of Q is a subfactorization in which every term is maximal. A factor of Q is any language which is a term in some factorization of Q. A left (right) factor is one which can be the leftmost (rightmost) term in a factorization of Q.

Lemma 1 Any left factor is the left factor in some 2-term factorization. Any right factor is the right factor in some 2-term factorization. Any factor is the central term in some 3-term factorization. The condition that L.R be a factorization of Q defines a (1-1) correspondence between left and right factors of Q.

Let Q be a regular language having q left factors. Following Conway, let us index the left and right factors as L_1, L_2, ... , L_q and R_1, R_2, ... , R_q wherein corresponding factors (see lemma 1) are given the same index. We now define Q_{ij} $(1 \leq i, j \leq q)$ by the condition that $L_iQ_{ij}R_j$ is a subfactorization of Q in which Q_{ij} is maximal. (It is important to note that $L_iQ_{ij}R_j$ may not be a factorization of Q). We note that, by lemma 1, H is a factor of Q if and only if it is some Q_{ij}. Thus the factors of Q are organised into a $q \times q$ matrix which is called the factor matrix of Q and is denoted $\lceil Q \rceil$.

Various properties of the factor matrix may be observed [5], some of which are summarized below.

Theorem 2

(i) H is a factor of Q <=> H is some entry Q_{ij} in the factor matrix $\lceil Q \rceil$.

(ii) Q_{ij} is maximal in the subfactorizations $L_i.Q_{ij} \subseteq L_j$ and $Q_{ij}.R_j \subseteq R_i$. Thus Q_{ij} is a right factor of L_j and a left factor of R_i.

(iii) $\exists$ unique indices s and t such that $Q = L_t = R_s = Q_{st}$, $L_i = Q_{si}$ and $R_i = Q_{it}$.

(iv) $\lceil Q \rceil = \lceil Q \rceil^*$.

(v) If $A_1.A_2 \ldots A_m \subseteq Q_{ij}$ is a subfactorization of Q_{ij}, then $\exists$ indices $k_1, k_2, \ldots, k_{m-1}$ such that $A_1 \subseteq Q_{ik_1}$, $A_2 \subseteq Q_{k_1k_2}$, $\ldots$, $A_m \subseteq Q_{k_{m-1}j}$.

Theorem 2 is an extremely interesting and powerful theorem, from which most results on factors can be deduced immediately. Part (iii) tells us that the s th row of $\lceil Q \rceil$ contains all the left factors and the t th column all the right factors, and the intersection of this row and column is the language Q itself. This and (iv), $\lceil Q \rceil = \lceil Q \rceil^*$, suggest very strongly that there is some recogniser of Q,(G,{s},{t}), consisting of a graph G with start node s and terminal node t, such that L_i is the set of all words taking node s to node i, and R_j is the set of all words taking node j to node t. In fact there is often more than one such G, but we shall show that there is a unique minimal one.

2.2 The Factor Graph

In this section we shall outline the proof that there is a unique minimal matrix G_Q such that $\lceil Q \rceil = G_Q^*$. G_Q is a constant + linear matrix and so is called the factor graph of Q.

Theorem 3 (Conway) $\exists$ unique maximal constant and linear matrices C_{max} and L_{max} such that $\lceil Q \rceil = (C_{max} + L_{max})^*$.

Proof C_{max} and L_{max} are defined to be the unique maximal constant and linear matrices (respectively) such that $\lceil Q \rceil \supseteq C_{max}$ and $\lceil Q \rceil \supseteq L_{max}$. The reader is referred to [3] or [5] for the remainder of the proof.

Let A, B and C be p×p matrices, elements of which are regular languages. Let $[B\backslash C]_{ij} = [b_{ij}\backslash c_{ij}]$ where \ denotes set difference. Let E be the p×p matrix, where $E = [e_{ij}]$, $e_{ij} = e$ if $i = j$ and $e_{ij} = \emptyset$ otherwise. Finally B^{2+*} is defined to be $B^2.B^*$.

Lemma 4 $C_{max}\backslash E$ is acyclic.

Proof   Suppose $C_{max}\backslash E$ contains cycles. Then there must be two distinct nodes i and j such that $[C_{max}\backslash E]_{ij} = e = [C_{max}\backslash E]_{ji}$. Using $\lceil Q \rceil = \lceil Q \rceil^*$ (2(iv)) we may deduce that $Q_{si} \supseteq Q_{sj} \supseteq Q_{si}$. I.e. $Q_{si} = L_i = Q_{sj} = L_j$. Similarly $R_i = R_j$. But then $i = j$ and we have a contradiction.

The main theorem in this section is the following.

Theorem 5   Let Q be a regular language, and let C_{max} and L_{max} be as defined in Theorem 3. Then there is a unique minimal matrix G_Q such that $G_Q^* = \lceil Q \rceil$, given by $G_Q = ((C_{max} + L_{max})\backslash E)\backslash((C_{max} + L_{max})\backslash E)^{2+*}$. Moreover the triple $(G_Q,\{s\},\{t\})$ (where s and t are given by Theorem 2 (iii)) is a recogniser for Q.

G_Q is a constant + linear matrix and so its graph will be called the factor graph of Q.

Proof   The proof given here differs from that given in [4] and was suggested by Mike Paterson. We assume two fundamental properties of regular languages [8]:

(a) The matrix equations $R = AR+B$ and $S = SA+C$ have the unique solutions $R = A^*B$ and $S = CA^*$, respectively, provided that A does not possess the empty word property.

We note also from lemma 4 and the definition of the empty word property [8] that:

(b) Neither $(C_{max}+L_{max})\backslash E$ nor G_Q possess the empty word property.

The proof of the theorem is now as follows.

Let $B = (C_{max}+L_{max})\backslash E$. Then $G_Q = B\backslash B^{2+*}$. So, by definition,

$$B^+ = G_Q + B.B^+.$$

Hence, by (a) and (b), $B^+ = B^*.G_Q$.

Therefore

$$B^* = E+B^+ = E+B^*.G_Q$$
$$= G_Q^* \text{ , by (a) and (b).}$$

I.e.

$$\lceil Q \rceil = (C_{max}+L_{max})^* \qquad \text{(Theorem 3)}$$
$$= ((C_{max}+L_{max})\backslash E)^* = G_Q^* .$$

An Example

The following simple example illustrates the concepts of factor graph and factor matrix.

Let Q = (a+b)*a(a+b)*b(a+b)*

Table 1 shows the corresponding left and right factors of Q and table 2 shows the factor matrix of Q and indicates the row s and column t corresponding to the left and right factors, respectively. Finally figure 1 shows the factor graph of Q.

Row/col. no.	Left Factors	Right Factors
1	(a+b)*	Q
2	(a+b)*a(a+b)*	(a+b)*b(a+b)*
3	Q	(a+b)*

Table 1

$$\begin{array}{ll} \text{left factors} \rightarrow & \left[\begin{array}{lll} (a+b)^* & (a+b)^*a(a+b)^* & Q \\ (a+b)^* & (a+b)^* & (a+b)^*b(a+b)^* \\ (a+b)^* & (a+b)^* & (a+b)^* \end{array}\right] \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \uparrow \\ & \qquad\qquad\qquad\qquad\qquad\qquad \text{right factors} \end{array}$$

Table 2

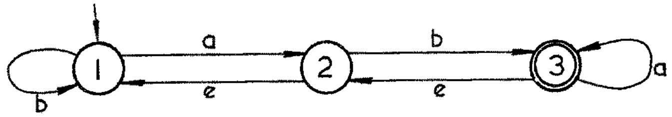


FIGURE 1

A number of other examples of factor graphs appear in [3] where also an algorithm for calculating the factor graph of a language is presented. Note however that the algorithm may have exponential time complexity (in the size of a regular expression denoting Q). Indeed the number of nodes in the factor graph of Q may be exponential in the size of a regular expression denoting Q [3]. These remarks militate strongly against the possibility of applying factor theory in any practical language recognition problems. Nevertheless the next two sections demonstrate one area where the factor graph has found a practical application.

A minor technical nuisance in the calculation of factor graphs is that $\emptyset$ may be a factor, and the factor graph can have up to two "useless" nodes i.e. nodes such that there is no path from s to the node, or no path from the node to t. We call the graph obtained by eliminating these nodes the <u>all-admissible factor graph</u> and all factor graphs we display will be all-admissible.

3. <u>Failure Functions</u>

The problem of relevance to the next three sections is the string-matching problem. That is, given a (long) symbol string $X = x_1\ x_2,\ \ldots\ x_m$, the "text", and another (short) string $Y = y_1\ y_2,\ \ldots\ y_n$, the "pattern", over the same alphabet V, find all occurrences of the pattern as a consecutive substring in the text.

Two methods for solving this problem are available, both of which have time-complexity which is linear in the combined length of the pattern and text strings. In the next section we shall relate the first method, the use of failure functions [6,7], to factor theory and in section 4 we relate the second method, Weiner's bi-trees [9], to factor theory.

<u>Definition</u> Given a string $Z = z_1\ z_2\ \ldots\ z_r$, the function f^*: $\{1\ldots r\} \rightarrow 2^{\{0\ldots r\}}$ is defined by:

$$f^*(i) = \{j \mid j \leq i \text{ and } z_1\ \ldots\ z_j = z_{i-j+1}\ \ldots z_i\}.$$

The <u>failure function</u> $f : \{1\ldots r\} \rightarrow \{0\ldots r-1\}$ is defined by [6]:

$$f(i) = \max\ \{j \mid j \in f^*(i) \text{ and } j \neq i\}.$$

The failure function of Z corresponds naturally to a transition diagram recognising V*Z. As an example, the functions f and f* defined for Z = aabba are given in table 3; fig. 2 shows the corresponding transition diagram. In the transition diagram there is an arc labelled e from node i to node j iff f(i) = j. It is our objective to show that fig. 2 is the factor graph of V*aabba, that C_{max} corresponds to f* and $(C_{max}\setminus E)\setminus(C_{max}\setminus E)^{2+*}$ to f.

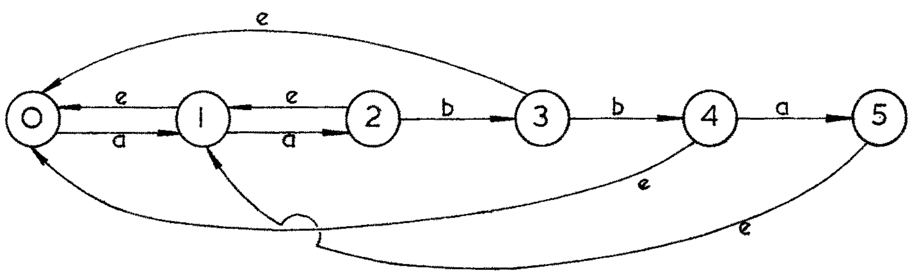


FIGURE 2

i	1	2	3	4	5
z_i	a	a	b	b	a
f*(i)	{0,1}	{0,1,2}	{0,3}	{0,4}	{0,1,5}
f(i)	0	1	0	0	1

TABLE 3

In all the following lemmas the string $Z = z_1z_2\ldots z_r$, over the alphabet V, is understood. Proofs of all results can be found in [4].

Lemma 6 $\quad f^*(i) = \{j \mid z_1\ldots z_i \in V^*z_1\ldots z_j\}$

$$= \{j \mid V^*z_1\ldots z_i \subseteq V^*z_1\ldots z_j\}.$$

Lemma 7 $\quad f^*(i) = \{i\} \cup f^*(f(i)).$

<u>Lemma 8</u> Q has r+2 left factors, namely $L_o=V^*$, $L_1=V^*z_1$, $L_2=V^*z_1z_2,\ldots$, $L_r=V^*z_1z_2\ldots z_r$ and $L_{r+1} = \phi$.

<u>Theorem 9</u> The all-admissible factor graph of $Q = V^*Z = V^*z_1z_2\ldots z_r$ can be constructed from the failure function for Z as follows:

(a) There are r+1 nodes, the (i-1)th node being connected to the ith node by an arc labelled $z_i (1\le i\le r)$

(b) There is an e-arc from the i th node to f(i).

<u>Proof</u> Let G(Z) be the graph constructed by applying steps (a) and (b).

By lemma 8 the factor graph of Q has r+1 nodes. Let C_{max} and L_{max} be the maximal constant and linear matrices such that $\lceil Q\rceil = (C_{max}+L_{max})^*$. Since $V^*z_1\ldots z_{i-1}.z_i.z_{i+1}\ldots z_r \subseteq Q$ and $V^*z_1\ldots z_{f(i)} \supseteq V^*z_1\ldots z_i$ (lemma 6) we have $C_{max}+L_{max} \supseteq G(Z)$. Let C_{min} and L_{min} be the constant and linear parts of G(Z). We must prove that (a) $C_{max} = C^*_{min}$, (b) $L_{max} \subseteq (C_{min}+L_{min})^*$, (c) $C_{min} \subseteq (C_{max}\setminus E)\setminus(C_{max}\setminus E)^{2+*}$ and (d) $L_{min} \subseteq L_{max}\setminus((C_{max}+L_{max})\setminus E)^{2+*}$. (a) and (b) establish that $\lceil Q\rceil = G(Z)^*$; (c) and (d) establish the minimality of G(Z).

Let $L_i = V^*z_1\ldots z_i$, $1\le i\le r$ and $L_o = V^*$.

(a) is immediate from lemma 7 and the identity $C^*_{min} = E + C_{min}.C^*_{min}$. (For $e \in Q_{ij} \iff L_j \supseteq L_i$ (Theorem 2) $\iff j \in f^*(i)$ (lemma 6). Also $e \in [C_{min}]_{ij} \iff j = f(i)$.)

To prove (b), suppose $a \in Q_{ij}$, where $a \in V$. Then $z_1\ldots z_i.a.z_{j+1}\ldots z_r \subseteq L_i.a.R_j \subseteq Q$ (Theorem 2) $\Rightarrow i \ge j-1$, $a = z_j$ and $z_1z_2\ldots z_{j-1} = z_{i-j+2}\ldots z_i$. Thus $j-1 \in f^*(i)$, whence $[C^*_{min}]_{i,j-1} = e$ (by (a)), and $[L_{min}]_{j-1,j} = a$. $\therefore$ $a \in [C^*_{min}.L_{min}]_{ij}$. I.e. $L_{max} \subseteq (C_{min}+L_{min})^*$.

To prove (c), suppose $e \in [C_{min}]_{ij}$ and $e \in [(C_{max}\setminus E)^{2+*}]_{ij}$. Then $j = f(i)$. But also $\exists$ k such that $[C_{max}\setminus E]_{ik} = e$ and $[C_{max}\setminus E]_{kj} = e$. I.e. $i \ne k \ne j$, $k \in f^*(i)$ and $j \in f^*(k)$. But then $i > k > j$ and $k \in f^*(i)$ contradicting the maximality of $f(i) = j$.

(d) is proved similarly. Suppose $z_j \in [(C_{max}+L_{max})\setminus E]^{2+*}_{j-1,j}$. Then $\exists$ indices k, m such that $e \in [C_{max}\setminus E]_{j-1,k}$, $z_j \in [L_{max}]_{km}$ and

$e \in [C_{max} \backslash E]_{mj}$ $\therefore$ $k < j-1$ and $j < m$. I.e. $k < m-1$. But also $z_1 \ldots z_k . z_j . z_{m+1} \ldots z_r \subseteq L_k . z_j . R_m \subseteq Q$ (Theorem 2) $\Rightarrow k \geq m-1$. This is a contradiction, so (d) is proved.

4. Generalised Failure Functions and Bi-trees

For lack of space we can only provide examples to illustrate the relationship between factor graphs and generalised failure functions and bi-trees. For further details see [4].

4.1 Generalised Failure Functions

An obvious generalisation of the string-matching problem is the following: Given a text X and p patterns $Y_1, Y_2, \ldots, Y_p$ find all occurrences of each pattern in the text. The failure function method can be generalised to solve this problem in time proportional to $m + n_1 + \ldots + n_p$, where m is the length of text and n_i is the length of the pattern Y_i [1]. The method involves constructing a tree from the set of pattern strings and defining a failure function from the nodes into the nodes of the tree. For example the tree constructed from the set of strings {abc, bc,bda} is shown in Fig. 3(a), and table 4 gives the failure function.

Node No.	1	2	3	4	5	6	7	8
Failure Node	1	1	5	6	1	1	1	2

TABLE 4

Letting $\$_1$, $\$_2$ and $\$_3$ by any new symbols not appearing in the pattern strings, fig. 3(b) shows the factor graph of $\{a,b,c,d,\$_1,\$_2,\$_3\}^* (abc\$_1 \cup bc\$_2 \cup bda\$_3)$. Note that, apart from an extra node and the \$ arcs to it, the linear part of the factor graph corresponds to the tree of fig. 3(a), and the constant part corresponds once more to the failure function, where there is an arc labelled e from node i to node j iff $f(i) = j$.

4.2 Bi-trees

Weiner's bi-tree method [9], for computing substring identifiers, constructs two trees a prefix tree and an auxiliary tree [2]. The string $Z = bbabb\vdash$ has prefix identifiers $\{bba, ba, a, bb\vdash, b\vdash, \vdash\}$. The prefix tree and auxiliary tree for Z are shown in figs. 4(a) and 4(b), respectively, whilst figs. 5(a) and 5(b) show the linear and constant parts of the factor graph of $Q = \{a,b,\vdash,\$_1,\ldots,\$_6\}^* \{bba\$_1, ba\$_2, a\$_3, bb\vdash\$_4, b\vdash\$_5, \vdash\$_6\}$, respectively.

References

1. Aho,A.V. and Corasick,M.J.
"Efficient string matching: An aid to bibliographic search"
C.A.C.M. (June 1975) 18, 6, Pp. 333-340.

2. Aho,A.V., Hopcroft,J.E. and Ullman,J.D.
"The design and analysis of computer algorithms"
Addison-Wesley: Reading, Mass. (1974).

3. Backhouse,R.C.
"Closure algorithms and the star-height problem of regular languages"
Ph.D. Thesis, Univ. of London, Sept. 1975.

4. Backhouse,R.C. and Lutz,R.K.
"Factor graphs, failure functions and bi-trees"
Dept. of Comp.Sc., Heriot-Watt U., Tech.Rep.No.4 (October 1976).

5. Conway,J.H.
"Regular algebra and finite machines"
Chapman and Hall: London (1971).

6. Fischer,M.J. and Paterson,M.S.
"String-matching and other products"
SIAM-AMS Proc. 7 (1974) Pp. 113-125.

7. Knuth,D.E., Morris,J.H. and Pratt,V.R.
"Fast pattern matching in strings"
TR CS-74-440, Stanford Univ., Stanford, California 1974.

8. Salmoaa,A.
"Theory of automata"
Pergammon Press: Oxford (1969).

9. Weiner,P.
"Linear pattern matching algorithms"
Conf. Record IEEE 14th Annual Symposium on Switching and Automata Theory (1973) Pp. 1-11.

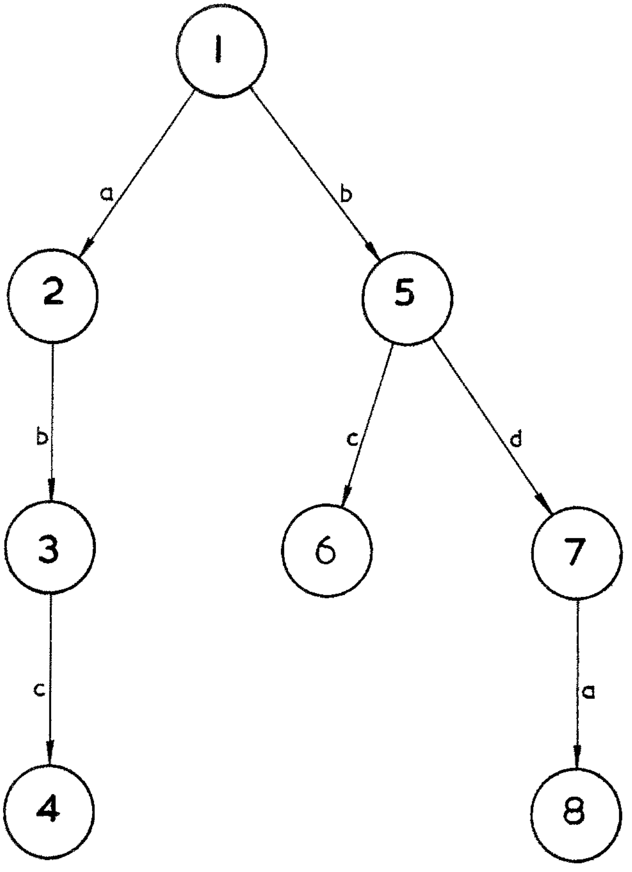


FIGURE 3(a)

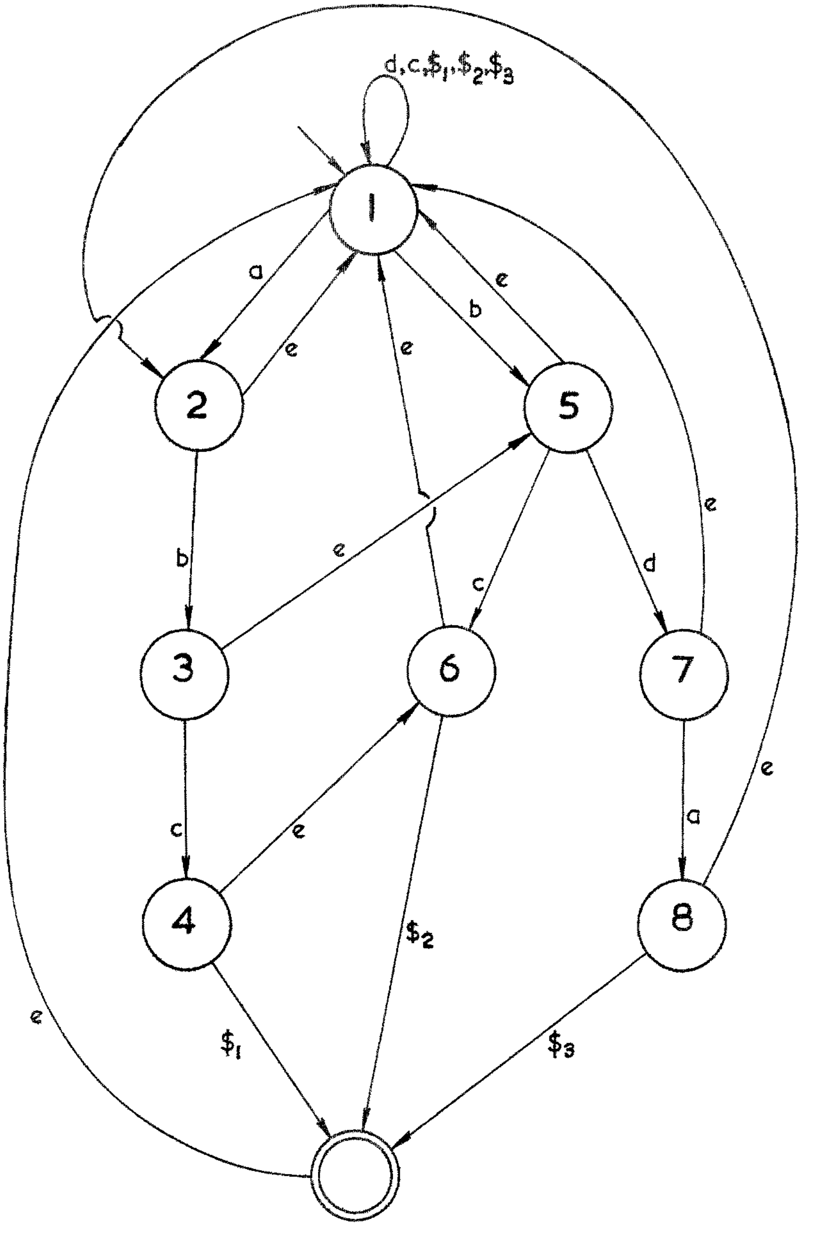
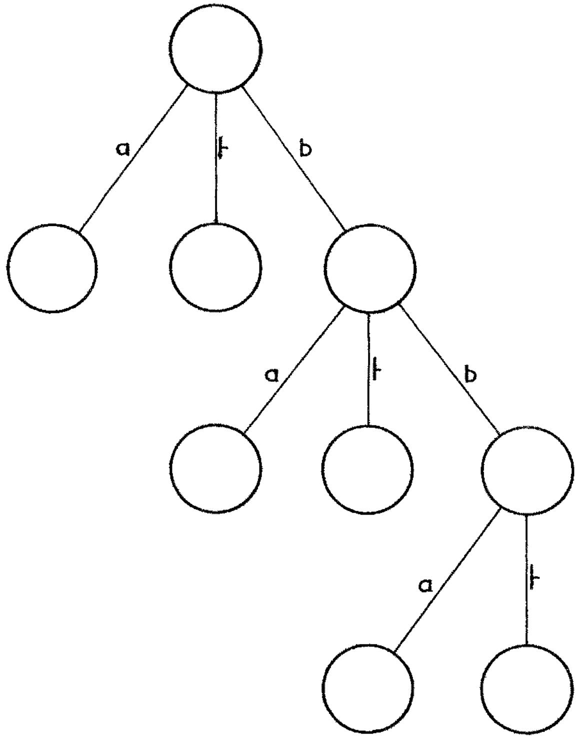
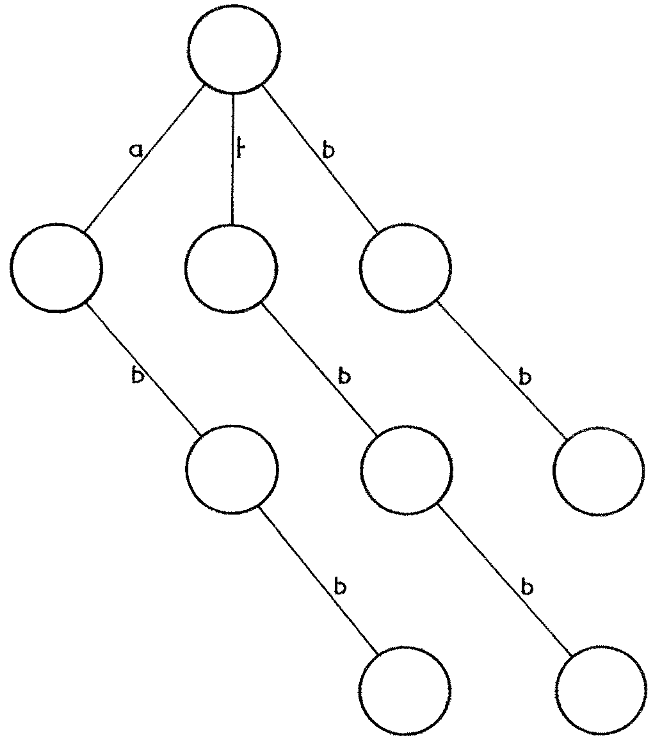


FIGURE 3(b) Factor Graph of $I^*(abc\$_1 \cup bc\$_2 \cup bda\$_3)$ where $I = \{a,b,c,\$_1,\$_2,\$_3\}$.

FIGURE 4 Z = bbabb⊦

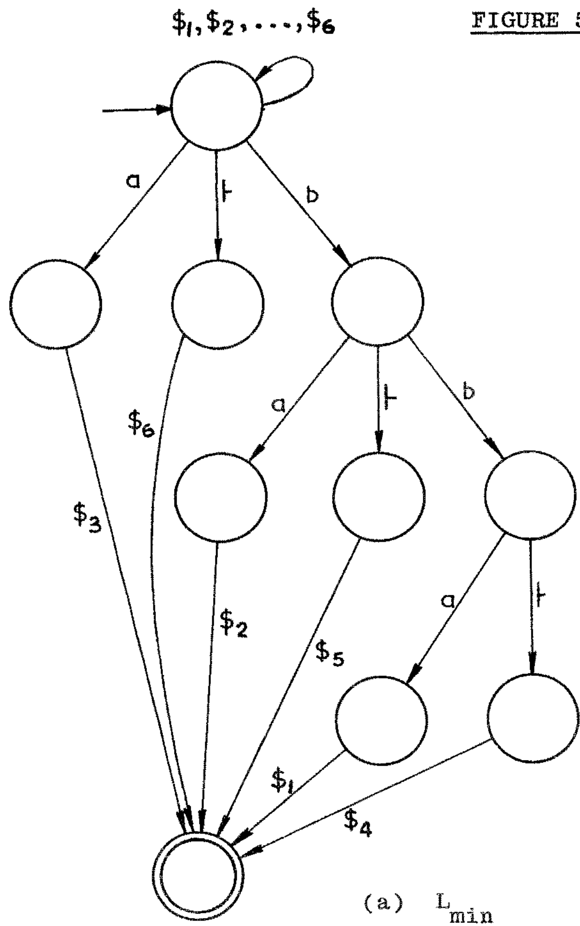


(a) Prefix tree

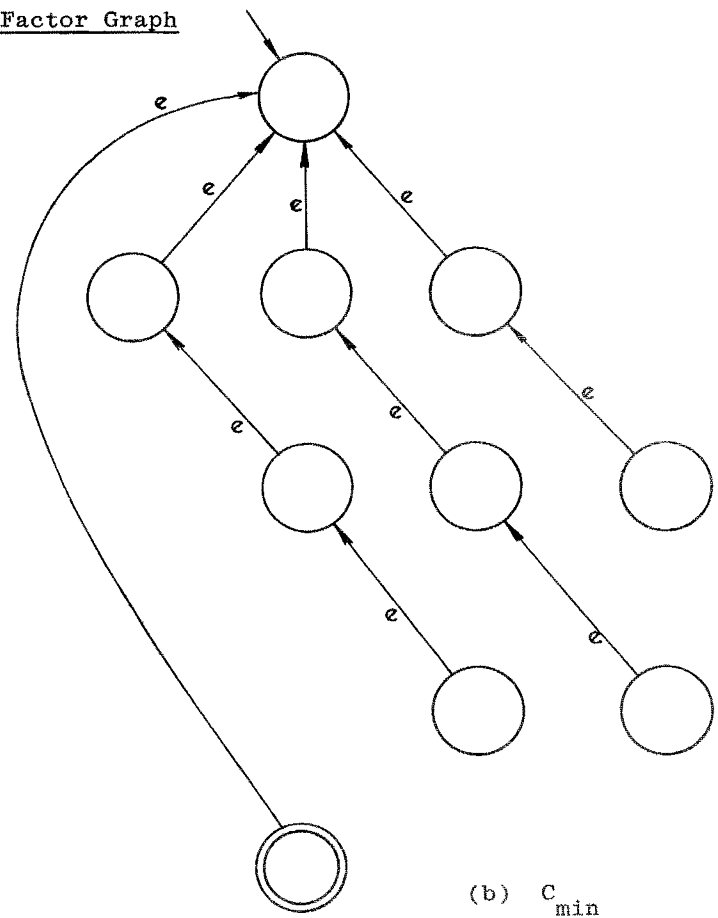


(b) Auxiliary prefix tree

FIGURE 5 Factor Graph



(a) L_{min}



(b) C_{min}