III THE FACTOR MATRIX AND FACTOR GRAPH

1. Motivation - The Star-Height Problem

In this chapter and the next we present some new results in the theory of factors of regular languages. The term "factor", as we shall use it, was introduced by Conway [13], and to him are due all the fundamental results of "factor theory". Although we would hope that our results will be of value in their own right as a contribution to factor theory, our interest in this theory was motivated by an interest in the "star-height problem" of regular languages.

Any regular language may be denoted by an unbounded number of different regular expressions. Thus (a+b)* and (a*b)*a* are two expressions denoting the same language (consisting of the set of all strings of a's and b's). Different expressions denoting the same language may of course differ rather trivially, but often they are remarkably "unalike". For example (b+a(aa*b)*b)* and (b+ab)*+(a+b)*b(b+ba)*b both denote the same language, (see example 1 of this chapter), but are quite different in form from each other. It is therefore natural to seek some canonical expression denoting a particular language - wherein the "canonicality" of an expression signifies that it is the "simplest" of all expressions which denote that language. Little, if any, progress has been made in finding a canonical form for regular languages, and so, as an intermediate step, efforts have been directed towards finding a way of assigning to each regular language some measure of the "complexity" of the language.

As a measure of the complexity of a language, Eggan's [18] definition of "star-height" would appear to be very reasonable and is generally accepted [9, 10, 11, 12, 15, 28, 29]. To define this, one first defines the star-height of a regular <u>expression</u> to be the maximum depth of embedded starred terms in the expression. In our earlier example (b+a(aa*b)*b)* has star-height 3 and (b+ab)*+(a+b)*b(b+ba)*b has star-height 1. The <u>star-height</u> of a <u>regular language</u> is the minimum star-height of all regular expressions which denote that language. The star-height problem is then just the problem of finding the star-height of any given regular language.

This problem was first posed in 1963 by Eggan, and has been tackled by various authors [9, 10, 11, 12, 15, 28, 29]. But, in common with many mathematical problems which are quite simply stated, its solution has not been forthcoming and it would appear to be a very difficult problem. In the next few paragraphs we have summarised those results which we consider an essential part of the repertoire of anyone who wishes to tackle this problem. We then continue to discuss, quite briefly, other results on this problem which have appeared, and indicate why these results led us to feel that Conway's factor theory was pertinent to the problem.

1.1 <u>Previous Work</u>

A concept which is fundamental to any study of the star-height problem, is Eggan's [18] notion of the <u>rank</u> of a transition graph. The rank is a measure of the loop complexity of a graph, but it is also very closely related to the notion of star-height.

In order to define rank some additional terminology is needed. A <u>subgraph</u> of a graph G is a graph G_Y determined by a set $Y \subseteq X$ of the nodes of G, having just those arcs (x_i, x_j) of G between nodes x_i and x_j, both of which are in Y. A subgraph is <u>strongly</u> <u>connected</u> if there is a path from x_1 to x_2 for every ordered pair (x_1, x_2) of its nodes. A <u>section</u> of a graph G is a strongly connected subgraph that is not a proper subgraph of any strongly connected subgraph of G.

The <u>rank</u> r(G) of a transition graph G is then defined as follows:

(i) If G is not strongly connected then

a) if G has no strongly connected subgraph $r(G)=0$, otherwise

b) r(G) is the maximum rank of all the sections of G.

(ii) If G is strongly connected $r(G) = n+1$ if and only if

a) it does not have rank i for any $i \leq n$, and

b) it has a node x whose deletion from G results in a subgraph of rank n.

The above recursive definition of rank is not particularly enlightening; readers not familiar with the notion should refer to McNaughton's paper [29], (from which the above definitions were taken), for a more detailed discussion.

Now consider any recogniser $(G,\mathcal{S},\mathcal{T})$ of the language Q. If we use, for example, the escalator method to calculate G* we obtain some regular expression α for Q. Reordering the nodes of G and reapplying the escalator method to calculate G* will result in a different regular expression β for Q which, moreover, will often have a different star-height to that of α. Thus there will be some optimal ordering of the nodes of G which, when using the escalator method to calculate G*, will yield some minimal star-height expression γ for Q from the graph G. The definition of the rank of the graph G is so contrived that the star-height of γ equals the rank of G. This is expressed by Eggan's theorem [18] which is essentially the following:

Eggan's Theorem

Consider the use of the escalator method to calculate G* from a given graph G. Then

(i) for a suitable ordering of the nodes of G the resulting regular expressions for those entries $[G^*]_{ij}$ for which G is an all-admissible recogniser have star-height equal to the rank of G. For other entries (ones for which G is not an all-admissible recogniser) the resulting regular expressions have star-height less than or equal to the rank of G.

(ii) For all other orderings of the nodes the resulting regular expressions for entries $[G^*]_{ij}$, for which G is an all-admissible recogniser, have star-height greater than or equal to the rank of G.

A converse to this result was also observed by Eggan, namely that to every regular expression there naturally corresponds a graph G having rank equal to the star-height of the expression. Thus one obtains the following corollary:

Corollary [18, 29] The star-height of a regular language equals the smallest rank of all transition graphs which recognise the language.

This corollary to Eggan's theorem immediately suggests an approach to the problem of determining the star-height of a regular language which is to find a method of obtaining a graph of least rank which is a recogniser of the language; it is this approach that almost all papers on the star-height problem have adopted. (Note that in the literature [29] the above corollary is usually referred to as "Eggan's theorem" - for reasons which will emerge we would like to remove the emphasis from this corollary.)

The most significant contribution to the star-height problem has been made in two papers by McNaughton [28,29]. In the first of the two, McNaughton studies languages whose semi-group is a pure group. For this class of languages McNaughton solved the star-height problem completely, although his solution involved enumerating a possibly rather large number of different graphs. However, for the subclass of this class consisting of languages for which the finite-state machine has a unique terminal state, he showed that the star-height of the language equals the rank of the finite-state machine. In spite of this result any connection between the structure of the semigroup of the language and its star-height would seem to be very illusory.

In order to establish his method McNaughton introduced the idea of a pathwise homomorphism between two graphs, and then proved a simple but fundamental theorem on the ranks of the graphs. As we shall need this result in the next chapter we state the theorem below.

Definition A pathwise homomorphism is defined as a mapping γ from the nodes and arcs of the transition graph G onto the nodes and arcs of G', such that $\gamma(x)$, for any node x of G, is a node of G' and the following two conditions hold between the arcs of G and G':

(PH1): For each arc B of G labelled b and leading from node x_1 to node x_2, either $\gamma(B)$ is a node of G' and $\gamma(x_1) = \gamma(x_2) = \gamma(B)$ or there is an arc $\gamma(B)$ in G' labelled b and leading from $\gamma(x_1)$ to $\gamma(x_2)$.

(PH2): If w is a word taking node x_1' to node x_2' in G' there are nodes x_1 and x_2 in G with $\gamma(x_1) = x_1'$ and $\gamma(x_2) = x_2'$ such that w takes node x_1 to node x_2 in G.

McNaughton's theorem is the following:

McNaughton's Pathwise Homomorphism Theorem If there is a pathwise homomorphism γ from G onto G' then the rank of G' is less than or equal to the rank of G.

Following McNaughton's work a number of papers were written by Cohen [9,10,11] and by Cohen and Brzozowski [12]. Some of these papers were concerned with extending McNaughton's work on pure group languages (to "reset-free" and "permutation-free" languages and to languages with the "finite intersection property"), the basic idea being to apply a combination of McNaughton's pathwise homomorphism theorem and the corollary to Eggan's theorem. Others provided more empirical results on the star-height problem.

However, with the exception of Eggan's theorem and McNaughton's pathwise homomorphism theorem, progress towards solving the star-height problem has been rather slow and fragmentary.

Our own first step in tackling the problem was as follows: Eggan's work showed that one closure algorithm (the escalator method) yielded regular expressions having star-height characterised by the rank of the graph. Is it possible that other closure algorithms yield regular expressions characterised by some other property of the graph and possibly even offer an improvement over the escalator method? In particular, do any of the elimination methods of Chapter II (e.g. Jordan elimination) offer such an improvement?

It is not long before one realises that this is not so, and that the rank is indeed the appropriate characteristic of a graph when applying any "elimination method". This statement is made precise in Appendix B where we give a general formulation of an "elimination method" and use this to prove the following theorem.

Theorem B4 If an elimination method is used to find G^* for a graph G, then G^* will contain regular expressions having star-height at least equal to the rank of G.

We have observed, however, that the elimination methods are all based on regular tautologies having analoques in linear algebra, but that not all regular tautologies have such analogues. This suggested two problems:

a) Can we invent new closure algorithms which do not have analogues in linear algebra? and

b) does the rank of a graph represent the "best" one can do using these new algorithms, or can we do better than the rank (i.e. obtain regular expressions for the entries $[G^*]_{ij}$ of star-height less than the rank of G)?

The main stumbling block to this approach is problem a), since it would appear extremely difficult to solve this problem in full generality. (Otherwise, no doubt, such algorithms would already have been published.) We were therefore obliged to seek a new closure algorithm which could be applied to particular classes of graphs, e.g. finite-state machines, each graph in the class being somehow naturally defined by a given language. Yet once again, for graphs such as the finite-state machine or semigroup machine, such algorithms seem impossible to find; thus we were forced to look for other "naturally defined" graphs to which such an algorithm could be applied. Cohen and Brzozowski [12] introduced the notion of a "subset automaton" but the difficulty in studying this class of graphs is that even for very simple regular languages the size of the "subset automaton" may be immense, thus precluding any empirical investigations.

1.2 The Relevance of Factor Theory

The class of graphs which we eventually decided to study are called "factor graphs". The idea of studying factor graphs came from reading McNaughton's paper [29] and the chapter in Conway's book [13] on factor theory. Let us use Conway's terminology and call G.H.K a <u>subfactorization</u> of a language Q if $G.H.K \subseteq Q$, and call H a <u>factor</u> of Q if there is no $H' \supset H$ such that $G.H'.K \subseteq Q$. (Thus H is in a sense maximal). In his paper, McNaughton presented an extremely useful technique for establishing a lower bound on the star-height of a given language. The technique involves spotting particular regular languages and showing that in any recogniser of Q these languages define nodes of the recogniser which are connected by arcs having loop complexity at least equal to the conjectured lower bound. Now, in general, subfactorizations of a language Q are mathematically unmanageable; but Conway showed that factors <u>are</u> manageable and, moreover, exhibit some remarkable properties. One such property particularly relevant to our aims is that the factors are all the entries in a matrix, denoted $\lceil Q \rceil$ and called the factor matrix, which is the closure $(C_{max} + L_{max})^*$ of a constant + linear matrix. Thus the factors naturally define a transition graph, <u>which is</u>, moreover, <u>a recogniser for Q</u>.

The matrix $C_{max} + L_{max}$, as it turns out, is not very useful for our purposes, but it is a stepping stone to proving that there is a unique <u>minimal</u> constant + linear matrix G_Q, which we call the factor graph of Q, such that G_Q^* $= \lceil Q \rceil$.

One of the main reasons for studying a problem like the star-height problem is the possible side-benefits that one can gain on the way. In trying to attack the problem using factor theory, we have been particularly on the lookout for such benefits; but also we are expressing a belief that a mathematical solution to the problem, which is not an impracticable "enumerative" solution, does exist. Rather disappointingly, the algorithm we shall present does not always yield a minimal star-height expression for a given regular language. Nevertheless we feel it is important as a contribution to our understanding of factor theory and because it offers a new approach to the problem, one which may well be more successful than earlier approaches using Eggan's theorem.

The presentation of the algorithm to determine G_Q^* occupies both this chapter and the next. This chapter is devoted to the fundamental properties of factors (due to Conway [13]) and to introducing the factor graph, G_Q, of a regular language Q and providing an algorithm to calculate G_Q. In the next chapter we introduce the notion of separability of factors and exploit this notion to derive an algorithm to calculate the closure G_Q^* of the factor graph. We also prove that the algorithm yields regular expressions for Q of star-height less than or equal to the rank of G_Q, and, as we demonstrate, in many cases strictly less than the rank of G_Q. Finally we discuss how the results could be extended in a further attack on the star-height problem.

2. ℓ-classes, r-classes and c-classes

We shall assume that the reader is familiar with the basic results on finite-state machines, to be found in Rabin and Scott [35], and the method of derivatives due to Brzozowski [3].

The purpose of this section is merely to summarise those results which we shall require later, and to define the ℓ, r and c-classes of a regular language Q.

2.1 Machine, Anti-machine and Semigroup

Let $Q \subseteq V^*$ be any language. Q naturally defines three equivalence relations on V^* - Q_ℓ, Q_r and Q_c - given by:

$$xQ_\ell y \Leftrightarrow (\forall z \varepsilon V^*,\ zx\varepsilon Q \Leftrightarrow zy\varepsilon Q)$$

$$xQ_r y \Leftrightarrow (\forall z \varepsilon V^*,\ xz\varepsilon Q \Leftrightarrow yz\varepsilon Q)$$

$$xQ_c y \Leftrightarrow (\forall u,v \varepsilon V^*,\ uxv\varepsilon Q \Leftrightarrow uyv\varepsilon Q).$$

These are, of course, the usual left-invariant equivalence relation, right-invariant equivalence relation and congruence relation introduced by Rabin and Scott [35].

The fundamental theorem linking these relations to regular languages is the following:

<u>Theorem 2.1</u> A language $Q \subseteq V^*$ is regular $\Leftrightarrow$ the relation Q_ℓ is of finite index $\Leftrightarrow$ the relation Q_r is of finite index $\Leftrightarrow$ the relation Q_c is of finite index.

<u>Definition</u> Let Q be a regular language. By theorem 2.1, each of the relations Q_ℓ, Q_r and Q_c partitions V^* into a finite number of equivalence classes. We shall call an equivalence class modulo Q_ℓ an <u>r-class</u> of Q, an equivalence class modulo Q_r an <u>ℓ-class</u> of Q and an equivalence class modulo Q_c a <u>c-class</u> of Q.

Note the peculiar switch: an equivalence class modulo Q_ℓ is an <u>r</u>-class of Q. The reason for this will become evident later.

We shall also write $\ell(x)$ for the ℓ-class containing x, $r(x)$ for the r-class containing x, and $c(x)$ for the c-class containing x.

Definition The machine of a regular language Q is the unique deterministic recogniser of Q having the least number of nodes.

The anti-machine of Q is the machine of $\overleftarrow{Q}$, where $\overleftarrow{Q}$ denotes the set of all words which are the reverse of words in Q.

Nodes of the machine and anti-machine will usually be called states.

The semigroup of Q is the quotient of the free semigroup V^* with respect to the congruence relation Q_c.

The machine and the ℓ-classes of Q, and the anti-machine and the r-classes of Q are connected by the following theorem.

Theorem 2.2 Let Q be a regular language. Let the states of the machine for Q be $\{\ell_1, \ldots, \ell_n\}$ and the states of the anti-machine be $\{r_1, \ldots, r_m\}$. Suppose that ℓ_1 and r_1 are the start states of the respective machines and let $x \varepsilon V^*$. Then we have:

(a) If x takes the start state ℓ_1 to state ℓ_i of the machine, then the ℓ-class containing x, $\ell(x)$, is the set of all words which also take state ℓ_1 to state ℓ_i.

(b) If $\overleftarrow{x}$ takes the start state r_1 to state r_j of the anti-machine, then the r-class containing x, $r(x)$, is the set of the reverse of all words which take state r_1 to state r_j.

Corresponding to the semigroup we can always construct a <u>semigroup machine</u>, whose states correspond to elements c_i of the semigroup, and where, for all $a \varepsilon V$, there is an arc labelled a from state c_i to c_j if $c_i \cdot c(a) = c_j$. Let c_1 be the identity element of the semigroup. We then have:

(c) If x takes the state c_1 to state c_t of the semigroup machine, then the c-class containing x, c(x), is the set of all words which also take state c_1 to state c_t.

<u>Corollary</u> The ℓ, r and c-classes of Q are regular if Q is regular.

Because of this theorem, we shall henceforth use the symbols ℓ_1, ℓ_2, ... to denote states of a machine for a regular language Q and also to denote the ℓ-classes of Q to which they correspond. (And similarly of course with the symbols r_1, r_2, ... and c_1, c_2, ...).

(2.2) <u>Derivatives, Anti-derivatives and Contexts</u>

Let us consider the relation Q_r. We note that any word $x \varepsilon V^*$ partitions V^* into two sets, denoted D_xQ and $\sim D_xQ$, where

$$D_xQ = \{y | xy \varepsilon Q\}$$
$$\sim D_xQ = \{y | xy \notin Q\} .$$

D_xQ is called the <u>derivative of Q with respect to x</u>.

We then have:

<u>Lemma 2.3</u> $x \; Q_r \; y \Leftrightarrow D_xQ = D_yQ$.

This is the basis of the method of derivatives for calculating the machine of a language Q [3].

Similarly the relation Q_ℓ leads one to define anti-derivatives: The anti-derivative of Q with respect to x, denoted $\reflectbox{D}_x Q$ is

$$\reflectbox{D}_x Q = \{y \mid xy \varepsilon \overleftarrow{Q}\} = \{y \mid \overleftarrow{y}\overleftarrow{x} \varepsilon Q\} .$$

<u>Lemma 2.4</u> $\quad x\ Q_\ell\ y \Leftrightarrow \reflectbox{D}_{\overleftarrow{x}} Q = \reflectbox{D}_{\overleftarrow{y}} Q$.

Finally, the relation Q_c partitions the set $V^* \times V^*$ into $C_x Q$, the <u>context</u> of x in Q, and $\sim C_x Q$ where

$$C_x Q = \{(u,v) \mid uxv \ \varepsilon\ Q\} .$$

<u>Lemma 2.5</u> $\quad x\ Q_c\ y \Leftrightarrow C_x Q = C_y Q$.

The following observation, although rather elementary, is quite important in the sequel.

<u>Theorem 2.6</u> (a) The word derivatives $D_x Q$ of a language Q are unions of r-classes of Q, where $D_x Q \supseteq r(y)$ if and only if $xy \ \varepsilon\ Q$.

(b) The reverse of anti-derivatives of Q, i.e. languages of the form $\overleftarrow{\reflectbox{D}_{\overleftarrow{y}} Q}$, are unions of ℓ-classes of Q, where $\overleftarrow{\reflectbox{D}_{\overleftarrow{y}} Q} \supseteq \ell(x)$ if and only if $xy \ \varepsilon\ Q$.

(c) The contexts $C_x Q$ of a language Q are unions of subsets of $V^* \times V^*$ of the form $\ell \times r$, where ℓ is an ℓ-class of Q and r is an r-class of Q, where $C_x Q \supseteq \ell(u) \times r(v)$ if and only if $uxv \ \varepsilon\ Q$.

<u>Proof</u> Let Q be a language and let $x \ \varepsilon\ V^*$.

Then $y \ \varepsilon\ D_x Q \Leftrightarrow xy \ \varepsilon\ Q \Leftrightarrow \overleftarrow{x} \ \varepsilon\ \reflectbox{D}_{\overleftarrow{y}} Q$.

But by lemma 2.4, $\reflectbox{D}_{\overleftarrow{y}} Q = \reflectbox{D}_{\overleftarrow{y'}} Q$ for all y' such that $y'\ Q_\ell\ y$.

$\therefore\ y \ \varepsilon\ D_x Q \Leftrightarrow y' \ \varepsilon\ D_x Q$ for all y' such that $y'\ Q_\ell\ y$.

i.e. $D_x Q = \sum_{y \varepsilon D_x Q} r(y)$, and part (a) is proved.

Part (b) is proved similarly.

Consider now $C_x Q$. The pair $(u,v) \varepsilon C_x Q \Leftrightarrow uxv \varepsilon Q$ $\Leftrightarrow v \varepsilon D_{ux} Q$ and $\overleftarrow{u} \varepsilon D_{\overleftarrow{xv}} Q$.

But then, by an identical argument to that above, this implies that $u'xv' \varepsilon Q$ for all $u' \varepsilon \ell(u)$ and $v' \varepsilon r(v)$.

i.e. $C_x Q \supseteq \ell(u) \times r(v)$.

Whence $C_x Q = \sum\limits_{(u,v) \varepsilon C_x Q} \ell(u) \times r(v)$, and we have proved (c).

Note that although the displayed unions are over an infinite set, the number of distinct terms is finite when Q is regular, and so the unions themselves may be taken over only a finite set of words.

3. The Fundamentals of Factor Theory

The following definitions are taken from Conway [13].

Definitions Let F, G, H, ... , K, Q denote arbitrary languages (not necessarily regular).

F.G...H...J.K is a subfactorization of Q if and only if

$$F.G...H...J.K \subseteq Q. \qquad (*)$$

$\overline{F}.\overline{G}...\overline{H}...\overline{J}.\overline{K}$ dominates it if it is also a subfactorization of Q and $F \subseteq \overline{F}$, $G \subseteq \overline{G}$, ... , $K \subseteq \overline{K}$.

A term H is maximal if it cannot be increased 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.

Next we state two lemmas, due to Conway, which are quite fundamental to future results. The proofs are quite simple and can be found in Conway's book [13].

Lemma 3.1 Any subfactorization of Q is dominated by some factorization in which all terms originally maximal remain unchanged.

Lemma 3.2 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.

We shall now give a characterisation of the factors of Q which gives some insight into their properties. Recall (§2) that an ℓ-class of Q is a right-invariant equivalence class, an r-class is a left-invariant equivalence class and a c-class is a congruence class of Q.

Theorem 3.3 The left factors of any language Q are either ϕ (the empty set) or are sums of ℓ-classes of Q. The right factors of Q are either ϕ or are sums of r-classes of Q and the factors are ϕ ~~or~~ are sums of c-classes of Q.

Corollary (Conway) A language Q is regular if and only if it has a finite number of factors. The factors are regular for regular Q.

Proof Let L be a left factor in the two term factorization $L.R \subseteq Q$ of Q. If $L \neq \phi$, let $x \varepsilon L$ and consider any $y \varepsilon \ell(x)$. Since $L.R \subseteq Q$, $R \subseteq D_x Q = D_y Q$ (by Lemma 2.3). Therefore $y.R \subseteq Q$, and so, since L is maximal, $y \varepsilon L$. Hence $L \supseteq \ell(x)$, and $L = \sum_{x \varepsilon L} \ell(x)$, i.e. L is a sum of ℓ-classes of Q. Similarly any non-empty right factor is a sum of r-classes of Q.

If H is any factor of Q it is the central term in a factorization $LHR \subseteq Q$ (lemma 3.2). If $H \neq \phi$, let $x \in H$. Then the set $C_xQ = \{(u,v) | uxv \in Q\} \supseteq L \times R = \{(u,v) | u \in L, v \in R\}$. But if $y \in c(x)$, $C_y\dot{Q} = C_xQ \supseteq L \times R$. Thus, as above, $y \in H$ and $H = \sum_{x \in H} c(x)$.

The corollary follows from the corollary to Theorem 2.2.

The above characterisation of the factors of Q is different to Conway's. The advantage will be seen later when we consider the problem of calculating the factors of Q.

From now on, unless otherwise stated, we shall only consider the case when Q is regular.

4. The Factor Matrix

Following Conway, let us index the left and right factors as $L_1, L_2, \ldots, L_q$ and $R_1, R_2, \ldots, R_q$ wherein corresponding factors (see lemma 3.2) 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 lemmas 3.1 and 3.2, 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 <u>factor</u> <u>matrix</u> of Q and is denoted $\lceil Q \rceil$.

Various properties of the factor matrix may be observed, some of which are summarised below.

Theorem 4.1

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

(ii) Q_{ij} is maximal in the subfactorizations $L_i \cdot Q_{ij} \subseteq L_j$ and $Q_{ij} \cdot R_j \subseteq R_i$. 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 $A_1 \cdot A_2 \ldots A_m \subseteq Q_{ij}$ is a subfactorization of Q_{ij}, then $\exists$ indices $k_1, k_2, \ldots, k_{m-1}$ such that $A_1 \subseteq Q_{ik_1}$, $A_2 \subseteq Q_{k_1k_2}$, $\ldots$, $A_m \subseteq Q_{k_{m-1}j}$.

Proof Although the proofs of all parts of this theorem can be found in Conway's book, the proof technique is so fundamental that it is worth repeating.

The proof of (i) is contained in the preamble to the theorem.

To prove (ii), we observe that the subfactorization $(L_iQ_{ij}) \cdot R_j \subseteq Q$ is dominated by $L_j \cdot R_j \subseteq Q$. Therefore $L_i \cdot Q_{ij} \subseteq L_j$ is a subfactorization of L_j in which Q_{ij} must be maximal. Similarly $Q_{ij} \cdot R_j \subseteq R_i$ is a subfactorization of R_i in which Q_{ij} is maximal. The rest follows from lemmas 3.1 and 3.2.

The indices s and t in part (iii) are chosen by the condition that $L_t \cdot R_t \subseteq Q$ dominates the subfactorization $Q \cdot e \subseteq Q$, and $L_s \cdot R_s \subseteq Q$ dominates the subfactorization $e \cdot Q \subseteq Q$. Then, by definition, $L_t \supseteq Q$; but also $Q \supseteq L_t \cdot R_t \supseteq L_t \cdot e = L_t$. Therefore $L_t = Q$. Similarly

$R_s = Q$. The index $s\ (t)$ is then unique, because all the left (right) factors are distinct. To prove that $Q_{st} = Q$, we note that Q_{st} is maximal in $L_s \cdot Q_{st} \cdot R_t \subseteq Q$ implies that it is also maximal in the subfactorization $L_s \cdot Q_{st} \subseteq L_t$. But $L_t = Q$, and R_s is maximal in $L_s \cdot R_s \subseteq Q$. Therefore $Q_{st} = R_s = Q$. Now $L_s \cdot Q \subseteq Q$ and $L_i \cdot R_i \subseteq Q$ are both factorizations of Q; so $L_s \cdot L_i \cdot R_i \subseteq Q$ is a subfactorization of Q in which L_i and R_i are maximal. Therefore, by definition of Q_{si}, $L_i = Q_{si}$. Similarly, $R_i = Q_{it}$.

Part (iv) can now be proved quite simply. We observe
(a) $Q_{ii} \supseteq e$ (Since $L_i \cdot R_i = L_i \cdot e \cdot R_i \subseteq Q$). and
(b) $Q_{ij} \supseteq Q_{ik} \cdot Q_{kj}$, for all $k = 1, 2, \ldots, q$. This follows because, by (ii), $L_k \supseteq L_i \cdot Q_{ik}$, and $R_k \supseteq Q_{kj} \cdot R_j$. Therefore $L_i \cdot (Q_{ik} Q_{kj}) \cdot R_j \subseteq L_k \cdot R_k \subseteq Q$. So, by definition, $Q_{ij} \supseteq Q_{ik} \cdot Q_{kj}$.

In matrix terms (a) and (b) are

(a)' $\lceil Q \rceil \supseteq E$, (b)' $\lceil Q \rceil \supseteq \lceil Q \rceil \cdot \lceil Q \rceil$.

Therefore $\lceil Q \rceil \supseteq E + \lceil Q \rceil \cdot \lceil Q \rceil$, and so by R1, $\lceil Q \rceil \supseteq \lceil Q \rceil^*$.

But $\lceil Q \rceil^* \supseteq \lceil Q \rceil$. Therefore $\lceil Q \rceil = \lceil Q \rceil^*$.

We shall prove (v) for the case $m = 2$; for $m > 2$ the result follows by simple induction. Suppose then that $A \cdot B \subseteq Q_{ij}$. Then $(L_i A) \cdot (B R_j) \subseteq Q$ is a subfactorization of Q and so must be dominated by some factorization $L_k \cdot R_k \subseteq Q$. I.e. $L_k \supseteq L_i \cdot A$ and $R_k \supseteq B \cdot R_j$. But then, by (ii), $A \subseteq Q_{ik}$ and $B \subseteq Q_{kj}$.

4.1 is an extremely interesting and powerful theorem, from which most results on factors can be deduced immediately. Particularly useful is 4.1 (iv), which we shall often apply in its alternative form (a) $Q_{ii} \supseteq e$ and (b) $Q_{ij} = \sum_k Q_{ik} \cdot Q_{kj}$.

Part (iii) tells us that the s th column of $\overline{|Q|}$ contains all the left factors and the t th row all the right factors, and the intersection of this row and column is the language Q itself. This and (iv), $\overline{|Q|} = \overline{|Q|}^*$, 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 <u>minimal</u> one.

Note, also, that (iii) does not imply that Q only occurs once in its factor matrix. For instance,

$$Q = (11)^* \quad \text{has factor matrix} \quad \overline{|Q|} = \begin{bmatrix} (11)^* & (11)^*1 \\ (11)^*1 & (11)^* \end{bmatrix}$$

in which Q occurs twice. (We mention this because there is a misprint in Conway's book [13,p 49], in which Conway says "The theorem does prevent Q from occurring twice in its matrix ...". This should, of course, read "does <u>not</u> prevent").

As we shall see, the combination of Theorems 3.3 and 4.1 (ii) is sufficient to enable one to calculate $\overline{|Q|}$. Various "brute force" methods can be used, but the method we give appears to be the most straightforward and easiest to apply.

5. <u>The Factor Graph</u>

Our objective in this section is to find a method of determining the factor matrix of a regular language Q. We shall prove 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 <u>factor graph</u> of Q.

The proof technique we use may seem rather round-about, and so it is worth while explaining the difficulty. Consider any element A of a regular algebra R. We shall call any X such that $X^* = A^*$ a <u>starth root</u> of A^*. Our aim is to prove that there is a unique minimal starth root of $\lceil Q \rceil$. In a <u>free</u> regular algebra it is quite easy to prove that there is always a unique minimal starth root of any element α^* in the algebra (see Brzozowski [4]). The proof, however, relies on length considerations and does not apply to all regular algebras. Indeed it is not generally true. Consider the matrix $M = \begin{bmatrix} e & e & e \\ e & e & e \\ e & e & e \end{bmatrix}$

This matrix has starth roots $A_1 = \begin{bmatrix} \phi & e & \phi \\ \phi & \phi & e \\ e & \phi & \phi \end{bmatrix}$ and

$A_2 = \begin{bmatrix} \phi & \phi & e \\ e & \phi & \phi \\ \phi & e & \phi \end{bmatrix}$, which are both minimal.

Thus there is no <u>unique</u> minimal starth root of M.

In order to prove that $\lceil Q \rceil$ nevertheless does have a unique minimal starth root we first prove that $\lceil Q \rceil$ has a starth root which is a constant + linear matrix, and then that this matrix can be reduced to one which is minimal.

<u>Lemma 5.1</u> $\exists$ unique <u>maximal</u> constant and linear matrices C_{max} and L_{max} such that $\lceil Q \rceil \geq (C_{max} + L_{max})^*$.

<u>Proof</u> Define C_{max} and L_{max} to be the unique maximal constant and linear matrices (respectively) such that

$\lceil Q \rceil \supseteq C_{max}$ and $\lceil Q \rceil \supseteq L_{max}$. Then $\lceil Q \rceil \supseteq C_{max} + L_{max}$ and, as $\lceil Q \rceil = \lceil Q \rceil^*$, $\lceil Q \rceil \supseteq (C_{max} + L_{max})^*$.

We shall now prove that the inequality in the above lemma can be changed to an equality.

<u>Theorem 5.2 (Conway)</u> Let C_{max} and L_{max} be the unique maximal constant and linear matrices of lemma 5.1 such that $\lceil Q \rceil \supseteq (C_{max} + L_{max})^*$. Then $\lceil Q \rceil = (C_{max} + L_{max})^*$.

<u>Proof</u> Suppose $x \in Q_{ij}$. If $x = e$ then (a) $x \in [C_{max}]_{ij}$, since C_{max} is maximal. Otherwise (b) $x = a_1 a_2 \dots a_m$ is a word of length $m \geq 1$, in which each a_p is a letter. But then applying Theorem 4.1(v), $\exists$ integers $k_1, k_2, \dots, k_{m-1}$ such that $a_1 \in Q_{ik_1}$, $a_2 \in Q_{k_1 k_2}$, ... , $a_m \in Q_{k_{m-1} j}$. But then $a_1 \in [L_{max}]_{ik_1}$, $a_2 \in [L_{max}]_{k_1 k_2}$, ... , $a_m \in [L_{max}]_{k_{m-1} j}$. I.e. $x \in [L_{max}.L^*_{max}]_{ij}$. But (a) and (b) imply $\lceil Q \rceil \subseteq C_{max} + L_{max}.L^*_{max}$. But, using lemma 5.1, $\lceil Q \rceil \subseteq C_{max} + L_{max}.L^*_{max} \subseteq (C_{max} + L_{max})^* \subseteq \lceil Q \rceil$. Hence the theorem.

We have already mentioned that in a free regular algebra R_F any event A^* has a unique minimal starth root. This is given by $(A \setminus E) \setminus (A \setminus E)^{2+*}$, where E is the unit element of R_F, $\setminus$ denotes set difference and x^{2+*} denotes $x^2 + x^3 + x^4 + \dots$. In the algebra $M_p(R_F)$, of $p \times p$ matrices over the free regular algebra R_F, the most we can say is that if $(A \setminus E) \setminus (A \setminus E)^{2+*}$ is a starth root then A^* does have a unique minimal starth root which is given by the above expression. More formally:

<u>Theorem 5.3</u> Let A be an element of $M_p(R_F)$ where R_F is a free regular algebra. Let $M_p(R_F)$ have unit element E.

Let $[B\backslash C]_{ij} = [b_{ij}\backslash c_{ij}]$ where $\backslash$ denotes set difference.
If $A^* = \{(A\backslash E)\backslash(A\backslash E)^{2+*}\}^*$ then $(A\backslash E)\backslash(A\backslash E)^{2+*}$ is the unique minimal starth root of A^*.

Proof Let $X = (A\backslash E)\backslash(A\backslash E)^{2+*}$. By assumption X is a starth root of A^*. Suppose Y is also a starth root. We must show that $X \subseteq Y$.

Suppose $w \in X_{ij}$.

Clearly $w \in Y^*_{ij} = [(Y\backslash E)^*]_{ij}$, because Y is a starth root of A^* and $A^* \supseteq X$. Hence $w \in [(Y\backslash E)^n]_{ij}$ for some n where, by definition of X, $n \geq 1$.

Now $Y \subseteq A^* = (A\backslash E)^*$.

Hence $Y\backslash E \subseteq (A\backslash E)^+$.

$\therefore$ $w \in [(Y\backslash E)^n]_{ij} \subseteq [((A\backslash E)^+)^n]_{ij}$.

But $w \in X_{ij} = [(A\backslash E)\backslash(A\backslash E)^{2+*}]_{ij}$

$\Rightarrow$ $n = 1$

$\Rightarrow$ $w \in [Y\backslash E]_{ij} \subseteq Y_{ij}$.

I.e. $X \subseteq Y$ and the theorem is proved.

Considering the matrix M mentioned at the beginning of this section, we find that

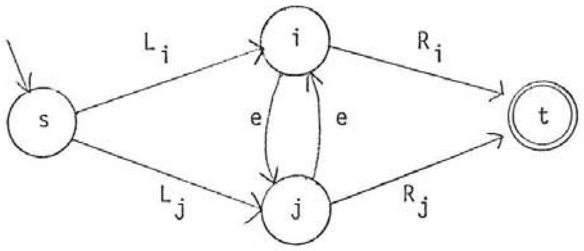
$$(M\backslash E)\backslash(M\backslash E)^{2+*} = \begin{bmatrix} \phi & \phi & \phi \\ \phi & \phi & \phi \\ \phi & \phi & \phi \end{bmatrix}$$

This is clearly not a starth root of M.

We note however that M has e-cycles which pass through more than one node. This cannot be true of the factor matrix as the next lemma states. This observation together with theorems 5.2 and 5.3 enable us to proceed to the proof of our main theorem.

Lemma 5.4 $C_{max}\backslash E$ is acyclic.

Proof Suppose $C_{max}\backslash E$ is cyclic. Then there must be two distinct nodes i and j such that $[C_{max}]_{ij} = e = [C_{max}]_{ji}$. Now consider the matrix $\lceil Q \rceil$. It will help if the reader refers to the figure below, which shows part of the matrix $\lceil Q \rceil$. The nodes labelled s and t are the nodes mentioned in Theorem 4.1.



We have indicated by labelled arrows that

$$Q_{si} = L_i, \quad Q_{it} = R_i$$
$$Q_{sj} = L_j \text{ and } Q_{jt} = R_j .$$

Since $\lceil Q \rceil \geq C_{max}$, $Q_{ij} \geq e$ and $Q_{ji} \geq e$.

This has also been indicated. We shall now prove that $L_i = L_j$ and $R_i = R_j$.

We have $\lceil Q \rceil = \lceil Q \rceil^*$, (4.1(iv)), and $\lceil Q \rceil \geq C_{max}$, by Theorem 5.2.

Therefore $\lceil Q \rceil = \lceil Q \rceil \cdot \lceil Q \rceil \geq \lceil Q \rceil \cdot C_{max}$.

Hence $Q_{si} \geq Q_{sj} \cdot [C_{max}]_{ji} = Q_{sj}$

and $Q_{sj} \geq Q_{si} \cdot [C_{max}]_{ij} = Q_{si}$.

Therefore $Q_{si} = Q_{sj}$ i.e. $L_i = L_j$.

Similarly, using $\lceil Q \rceil \geq C_{max} \cdot \lceil Q \rceil$, we get $R_i = R_j$.

But then nodes i and j cannot be distinct and we have a contradiction. Therefore the initial assumption is incorrect and $C_{max}\backslash E$ must be acyclic.

Corollary 5.5 Let $\lceil Q \rceil$ be a q×q matrix. Then $(C_{max} \backslash E)^p = N$ for all $p \geq q$.

Finally we come to the main theorem of this chapter.

Theorem 5.6 Let Q be a regular language, and let C_{max} and L_{max} be as defined in Theorem 5.2. 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 4.1 (iii)) is a recogniser for Q.

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

Proof Using Theorem 5.3, we need only prove that $G_Q^* = \lceil Q \rceil$ where $G_Q = ((C_{max} + L_{max}) \backslash E) \backslash ((C_{max} + L_{max}) \backslash E)^{2+*}$. In turn this only requires proving that $G_Q^* \geq (C_{max} + L_{max}) \backslash E$, since then $G_Q^* = (G_Q^*)^* \geq (C_{max} + L_{max})^* = \lceil Q \rceil$, by Theorem 5.2.

Let p1 : $w \in [(C_{max} + L_{max}) \backslash E]_{ij}$.

Then p2 : w has length 0 or 1.

Suppose p3 : $w \notin [G_Q^*]_{ij}$.

Then p4 : $w \notin [G_Q]_{ij}$

and so p5 : $w \in [((C_{max} + L_{max}) \backslash E)^{2+*}]_{ij}$.

Hence $\exists$ indices $i = k_1, k_2, \ldots, k_{m+1} = j$ and words $a_1, a_2, \ldots, a_m$ s.t. $m \geq 2$, $w = a_1 a_2 \ldots a_m$

and p1 : $a_h \in [(C_{max} + L_{max}) \backslash E]_{k_h k_{h+1}}$

and hence p2 : a_h has length 0 or 1, for all $h = 1, \ldots, m$.

Now for some h we must have

p3 : $a_h \notin [G_Q^*]_{k_h k_{h+1}}$

(otherwise $w \in [G_Q^*]_{ij}$).

Hence for this h

$$p4 : a_h \notin [G_Q]_{k_h k_{h+1}}$$

and so, for this h

$$p5 : a_h \in [((C_{max} + L_{max})\backslash E)^{2+*}]_{k_h k_{h+1}} .$$

Let this a_h be v. v has the same properties as w and so can in turn be expressed as the product of two or more words $b_1 b_2 \dots b_n$, where by the same argument one of the b_g's = u, say, also has the same properties as w. In this way we can express w as a product

$$w = y_1 y_2 \dots y_x$$

of an unbounded number x of words y_f, where

$$y_f \in [(C_{max} + L_{max})\backslash E]_{n_f n_{f+1}}$$

for some nodes $n_1, \dots ,n_{x+1}$. But the product of two linear matrices is either null or non-linear. Therefore at most one y_f has length one, and we conclude that $(C_{max}\backslash E)^p$ is non-null for all p. But this contradicts corollary 5.5. Hence the initial assumption that property p3 holds for w must be false. Hence $G_Q^* \supseteq (C_{max} + L_{max})\backslash E$, and by our earlier argument $G_Q^* = \boxed{Q}$. Finally, applying Theorem 5.3, G_Q is the minimal starth root of $\boxed{Q}$. The last part of the theorem follows immediately from Theorem 4.1 (iii), $Q = Q_{st} = [G_Q^*]_{st}$.

6. AN EXAMPLE

The previous results embody an algorithm for calculating $\boxed{Q}$, which we shall develop with the aid of an example. Essentially our aim is to calculate $C_{max} + L_{max}$; we could then use any one of the algorithms of chapter II to calculate

$\lceil Q \rceil$ from G_Q, using $\lceil Q \rceil = G_Q^*$ and
$G_Q = ((C_{max} + L_{max})\backslash E)\backslash((C_{max} + L_{max})\backslash E)^{2+*}$, although the next chapter will develop a rather better algorithm to determine G_Q^*.

Example 1: Let $Q = (b+a(aa^*b)^*b)^*$

6.1 Machine, Anti-machine and Semigroup

Theorem 3.3 suggests that we begin by calculating the machine, anti-machine and semigroup of Q. This we have done, using standard methods, in Figures 1(a), (b) and (c).

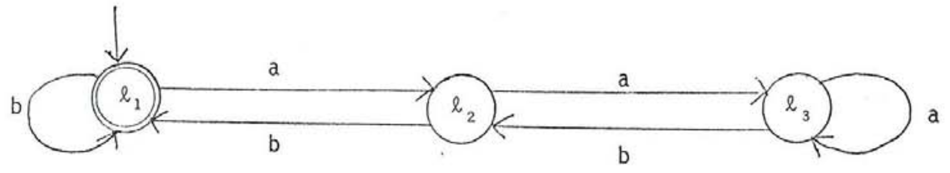


Fig. 1(a) Machine of Q.

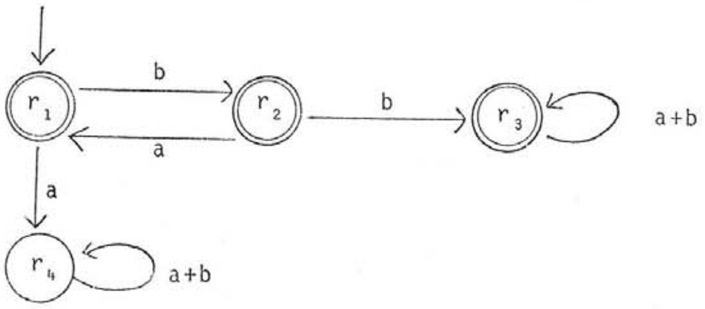


Fig. 1(b) Anti-machine of Q.

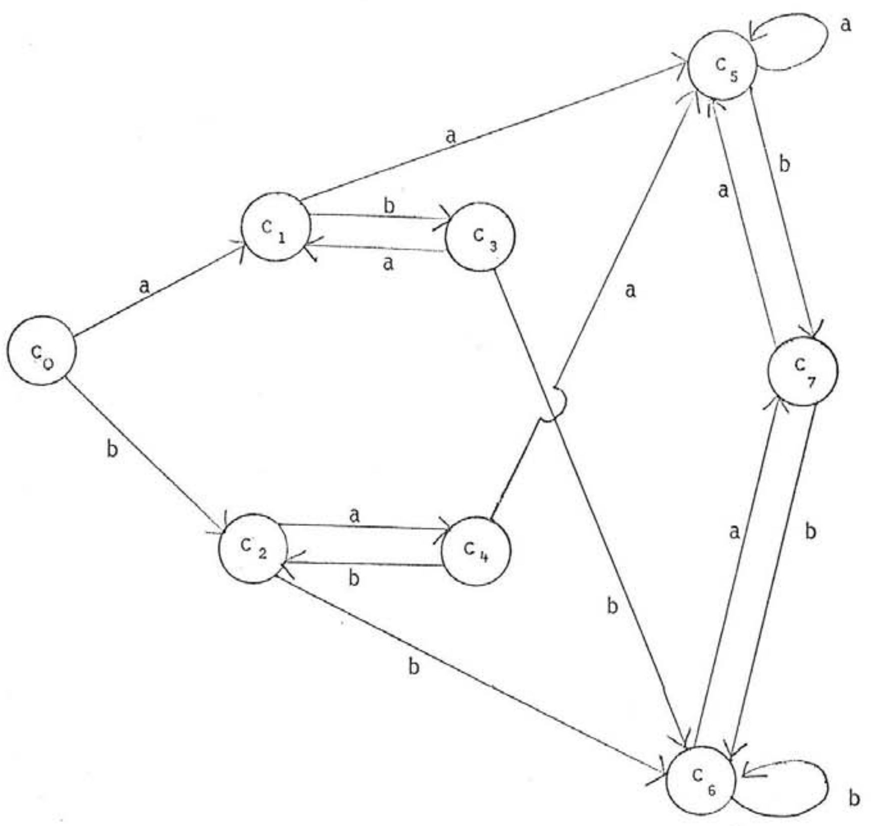


Fig. 1(c) Semigroup of Q.

In these figures we have labelled the nodes of the machine ℓ_1, ℓ_2 and ℓ_3, but, as stated after the corollary to theorem 2.2, we shall also use the symbols ℓ_1, ℓ_2, ℓ_3 to denote the sets $(b+a(aa^*b)^*b)^*$ $(=Q)$, $Qa(aa^*b)^*$ and $Qa(aa^*b)^*aa^*$, respectively, these being the right-invariant equivalence classes to which they correspond. Similarly r_1 is used to label a node of the anti-machine, but also denotes the set $(ab)^*$, this being the set of the reverse of all words which take node r_1 to itself in the anti-machine (=(ba)*) .

6.2 Calculating the Derivatives etc.

In tables 1(a) and (b) we have used Theorem 2.6 to determine the derivatives and anti-derivatives of Q as sums of r-classes and sums of ℓ-classes of Q, respectively. Thus in the first column of each table we have listed the ℓ- and r-classes, and the third column shows the corresponding derivative or (reverse of) anti-derivative as a union of r-classes or ℓ-classes of Q, as the case may be.

Node/ ℓ-class	Representative element	Derivative
ℓ_1	e	$r_1+r_2+r_3$
ℓ_2	a	r_2+r_3
ℓ_3	aa	r_3

Table 1(a) Machine.

Node/ r-class	Representative element	Reverse of anti-derivative
r_1	e	ℓ_1
r_2	b	$\ell_1 + \ell_2$
r_3	bb	$\ell_1 + \ell_2 + \ell_3$
r_4	a	ϕ

Table 1(b) Anti-machine.

A very important point to note here is that, since each ℓ-, r- or c-class consists of words which are <u>equivalent</u> with respect to some relation, the properties which follow from the equivalences can be found by considering <u>representative elements</u> of the equivalence classes only. In order to calculate the derivatives of Q as sums of r-classes we have chosen in column 2 of each table a representative element of each equivalence class. The choice is quite arbitrary. Then we have used Theorem 2.6 (a) directly to determine the derivative. For instance the class ℓ_2 has representative a

and since	$a \cdot e \notin Q$	(e is the representative of r_1),
and	$a \cdot a \notin Q$	(a ———— " ———— r_4),
but	$a \cdot b \in Q$	(b ———— " ———— r_2),
and	$a \cdot bb \in Q$	(bb ———— " ———— r_3),

the derivative corresponding to ℓ_2 is $r_2 + r_3$. Thus it is quite unnecessary in these calculations to calculate regular expressions representing the various ℓ-, r- and c-classes.

A similar table, illustrating part (c) of Theorem 2.6, could be constructed for the c-classes of Q. The part of this table for those c-classes containing e or a word consisting of a single letter is shown in Table 1(c). Once again column 1 lists the c-class and column 2 gives a representative element of each c-class. The third column expresses the corresponding context of Q, C_xQ (where x is the representative), as a union of direct products of the form $\ell_i \times r_j$. A simple way of calculating the appropriate entry is as follows. Suppose one is considering the c-class c_k having as representative the element x. Consider each state ℓ_i of the machine in turn. If under input x state ℓ_i goes to state ℓ_j, read off the entry in column 3 of the ℓ_jth row of Table 1(a). Suppose this is $r_{j_1} + r_{j_2} + \ldots + r_{j_h}$. Then add $\ell_i \times (r_{j_1} + \ldots + r_{j_h})$ to the entry already in the third column of c_k.

For example consider the class c_1 having representative a. Let us write $\ell_i \xrightarrow{x} \ell_j$ if input x takes state ℓ_i to state ℓ_j of the machine. Then we have

$\ell_1 \xrightarrow{a} \ell_2$ and ℓ_2 has derivative $r_2 + r_3$. Hence enter $\ell_1 \times (r_2+r_3)$
$\ell_2 \xrightarrow{a} \ell_3$ and ℓ_3 ——— " ——— r_3. —— " —— $\ell_2 \times r_3$
$\ell_3 \xrightarrow{a} \ell_3$ and ℓ_3 ——— " ——— r_3. —— " —— $\ell_3 \times r_3$.

Thus the entry in column 3 for c_1 is

$$\ell_1 \times (r_2 + r_3) + \ell_2 \times r_3 + \ell_3 \times r_3 .$$

The reader should ignore column 4 of Table 1(c) for the time being.

Node/ Congruence class	Representative element	Context Form 1.	Context Form 2.
c_0	e	$\ell_1 \times (r_1 + r_2 + r_3)$	$L_1 \times R_1 + L_2 \times R_2 + L_3 \times R_3$
		$+ \ \ell_2 \times (r_2 + r_3)$	$+ \ L_1 \times R_2 + L_2 \times R_3 + L_1 \times R_3$
		$+ \ \ell_3 \times r_3$	$+ \ L_4 \times R_i \quad (i = 1,2,3,4)$
c_1	a	$\ell_1 \times (r_2 + r_3)$	$L_3 \times R_3 + L_1 \times R_2$
		$+ \ \ell_2 \times r_3$	$+ \ L_2 \times R_3 + L_1 \times R_3$
		$+ \ \ell_3 \times r_3$	$+ \ L_4 \times R_i \quad (i = 1,2,3,4)$
c_2	b	$\ell_1 \times (r_1 + r_2 + r_3)$	$L_3 \times R_2 + L_2 \times R_1$
		$+ \ \ell_2 \times (r_1 + r_2 + r_3)$	$+ \ L_2 \times R_2 + L_3 \times R_2$
		$+ \ \ell_3 \times (r_2 + r_3)$	$+ \ L_1 \times R_2 + L_2 \times R_3 + L_1 \times R_3$

Table 1(c)

6.3 The Left and Right Factors

We can now deduce the left factors and right factors of Q from Table 1(a). If we consider any sum $\ell_{i_1} + \ell_{i_2} + \dots + \ell_{i_k}$ of ℓ-classes of Q, including the empty sum ϕ, and then determine those r-classes $r_{j_1} + \dots + r_{j_m}$ common to the third column of all the i_1th, i_2th, ... , i_kth rows of Table 1(a), then

$$(\ell_{i_1} + \ell_{i_2} + \dots + \ell_{i_k}) \cdot (r_{j_1} + \dots + r_{j_m}) \subseteq Q$$

will be a subfactorization of Q. By inspection of all such subfactorizations we can deduce those which are also factorizations of Q. Thus for our example we would get the following subfactorizations:

$(\ell_1 + \ell_2 + \ell_3) \cdot r_3$, $(\ell_2 + \ell_3) \cdot r_3$, $(\ell_1 + \ell_3) \cdot r_3$, $\ell_3 \cdot r_3$,

$(\ell_1 + \ell_2) \cdot (r_2 + r_3)$, $\ell_2 \cdot (r_2 + r_3)$,

$\ell_1 \cdot (r_1 + r_2 + r_3)$,

$\phi \cdot (r_1 + r_2 + r_3 + r_4)$.

Table 2 Subfactorizations of Q.

We have displayed these subfactorizations in such a way as to make it evident that only those in the first column are also factorizations of Q.

This information is summarised in Table 3, in which we have also named the left and right factors L_1, L_2, L_3, L_4 and R_1, R_2, R_3, R_4.

Left factors			Right factors		
$L_t =$	L_1	ℓ_1	$R_s =$	R_1	$r_1 + r_2 + r_3$
	L_2	$\ell_1 + \ell_2$		R_2	$r_2 + r_3$
	L_3	$\ell_1 + \ell_2 + \ell_3$		R_3	r_3
	L_4	ϕ		R_4	$r_1 + r_2 + r_3 + r_4$

Table 3 Left and right factors

In the table we have also indicated that the indices s and t of Theorem 4.1(iii) are both equal to 1. This is because, from the machine (Fig. 1(a)), $Q = \ell_1$, and from the anti-machine (Fig. 1(b)), $Q = r_1 + r_2 + r_3$; but $L_1 = \ell_1$ and $R_1 = r_1 + r_2 + r_3$.

6.4 Construction of $C_{max} + L_{max}$ and G_Q

The penultimate step in the construction of the factor graph is to construct $C_{max} + L_{max}$. In our example the graph of $C_{max} + L_{max}$ will have four nodes (see Fig. 2). In order to fill in the arc labels there are two approaches we can adopt.

(i) In Table 1(c), column 4, we have expressed the context C_xQ of each constant or linear term x (i.e. $x = e$ or $x \in V$) in all possible ways in terms of direct products of left and right factors of Q. There is then an arc labelled x from node i to node j of $C_{max} + L_{max}$ if and only if there is an entry $L_i \times R_j$ in Column 4 of Table 1(c) of the row corresponding to x.

(ii) Alternatively, we can apply Theorem 4.1(ii), which can be restated as, $x \in Q_{ij} \Leftrightarrow L_i x \subseteq L_j$. To do this we use the representation of L_i as $\ell_{i_1} + \ell_{i_2} + \dots + \ell_{i_k}$, given in Table 3. C_{max} can be calculated immediately using $[C_{max}]_{ij} = e \Leftrightarrow L_i \subseteq L_j$. To find L_{max} consider each element $x \in V$ in turn. Under input x the state ℓ_{i_m} of the machine goes to state ℓ_{h_m}, say. Thus $L_i x = (\ell_{i_1} + \ell_{i_2} + \dots + \ell_{i_k}) \cdot x \subseteq \ell_{h_1} + \ell_{h_2} + \dots + \ell_{h_k}$, and $[L_{max}]_{ij} \geq x$ for all those j such that

$$L_j \supseteq \ell_{h_1} + \ell_{h_2} + \dots + \ell_{h_k} .$$

Using either of these techniques, we get the graph of $C_{max} + L_{max}$ shown in Fig. 2.

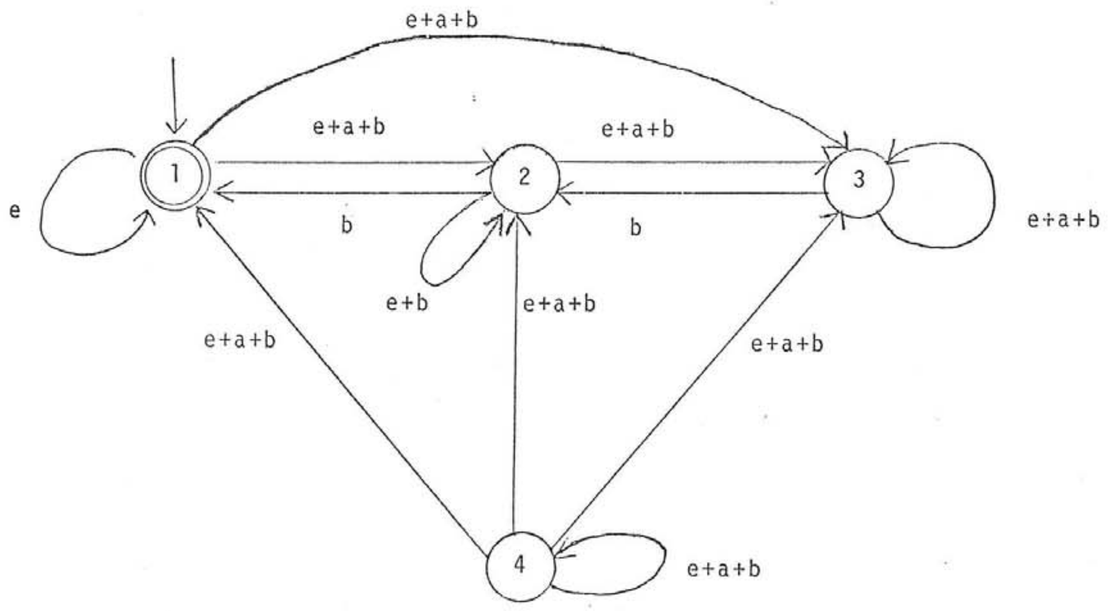


Fig. 2 $C_{max}+L_{max}$.

Finally to get the factor graph we remove those arc labels in $C_{max} + L_{max}$ which are in E or may be decomposed into paths in $((C_{max} + L_{max})\backslash E)^{2+*}$. The factor graph for this example is shown in Fig. 3. In both Figs. 2 and 3 we have indicated, in the usual way, that the graphs are recognisers of Q with start node s = 1 and terminal node t = 1.

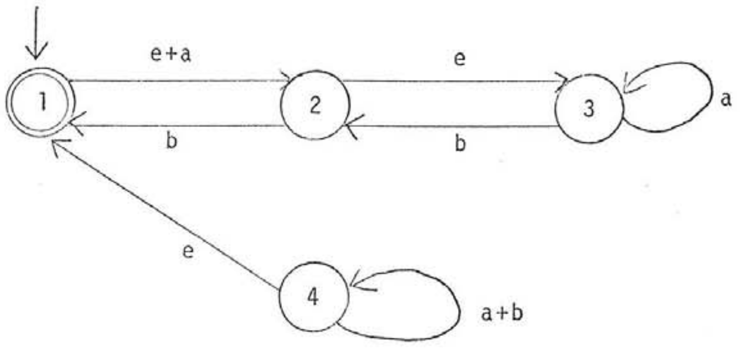


Fig. 3 The factor graph of Q.

6.5 The matrix $\boxed{Q}$

From G_Q we could now calculate $\boxed{Q} = G_Q^*$ using one of the standard methods of chapter II. However we can get the same information about $\boxed{Q}$ by determining each entry Q_{ij} as a sum of c-classes of Q. To do this we replace each entry in G_Q by the c-class of which it is a representative (see Fig. 1(c)). Thus G_Q is represented by

$$\begin{bmatrix} \phi & c_0 + c_1 & \phi & \phi \\ c_2 & \phi & c_0 & \phi \\ \phi & c_2 & c_1 & \phi \\ c_0 & \phi & \phi & c_1 + c_2 \end{bmatrix}$$

and then calculate G_Q^* in the algebra $M_q(\mathbb{R}(\underline{S}_Q))$ where $\underline{S}_Q$ is the semigroup of Q, and $\mathbb{R}(\underline{S}_Q)$ is the regular algebra generated by $\underline{S}_Q$ (c.f. I§2.3). For this example one can verify that G_Q^* is represented by

$$\begin{bmatrix} c_0+c_2+c_3+c_6 & \begin{matrix} c_0+c_1+c_2+c_3 \\ +c_4+c_6+c_7 \end{matrix} & S & \phi \\ c_2+c_6 & \begin{matrix} c_0+c_2+c_3 \\ +c_4+c_6+c_7 \end{matrix} & S & \phi \\ c_6 & \begin{matrix} c_2+c_3+c_6 \\ +c_7 \end{matrix} & S & \phi \\ S & S & S & S \end{bmatrix}$$

where S denotes the whole semigroup i.e. $S = \sum_{i=0}^{7} c_i$.

6.6 Some Remarks

There are various minor improvements one can make to the above method of calculating G_Q.

First of all, if it is only required to calculate G_Q, it is unnecessary to calculate the semigroup of the language. However, the representation of the matrix $\lceil Q \rceil$ in terms of c-classes, as in the last section, is (as we shall see) very useful and also much more informative than calculating regular expressions denoting each of the factors.

A second point concerns Tables 1(a) and (b). It should be noted that the third column of Table 1(b) can be deduced directly from the third column of Table 1(a) (or vice-versa), and thus gives redundant information.

This small amount of redundancy can, however, be quite useful in checking hand calculations and so is probably worth retaining.

Thirdly, we note that by length considerations,

$$G_Q = C_{min} + L_{min}, \text{ where}$$

$$C_{min} = (C_{max}\backslash E)\backslash(C_{max}\backslash E)^{2+*}$$

and $$L_{min} = L_{max}\backslash(L_{max} + C_{min})^{2+*} .$$

The above formulae suggest that one first calculates C_{max} and from it C_{min}, and then determines L_{max} and from it L_{min}. The sum of C_{min} and L_{min} is then the factor graph G_Q. In fact one rarely needs calculate C_{max} and L_{max} explicitly because one can remove arcs from these matrices by inspection as they are being constructed. Note also that the second method of calculating C_{max} and L_{max} (§6.4(ii)) is preferable to the first, and hence the construction of Table 1(c) is unnecessary - although it does add some insight into what is happening.

Finally, a minor technical nuisance in the study of factors is that ϕ may be a factor. In this example $L_4=\phi$ is a left factor, but ϕ is not a right factor. If ϕ is a factor then the factor graph can have up to two "useless" nodes, i.e. nodes such that there is no path from node s to the node, (for example node 4 of Fig.3), or no path from the node to node t. If we are interested in the graph G_Q as a recogniser for Q, we can always ignore these nodes and consider the resulting all-admissible recogniser for Q. In all future calculations we will take the liberty of disregarding this technical problem, and all the factor graphs we display will be all-admissible factor graphs.

7. An algorithm to calculate the factor graph

We are now in a position to summarise the steps in an algorithm to determine the (all-admissible) factor graph for a given regular language Q. We assume naturally that Q is given either as a regular expression or by a system of left (or right)-linear equations. Following the algorithm we have worked through another example, which shows explicitly the various steps of the algorithm.

Algorithm 1 To calculate the factor graph G_Q of a given regular language Q.

Step 1 Calculate the machine and anti-machine for the language Q. (Use the method of derivatives [3]). Label the states of the machine (anti-machine) $\ell_1, \ell_2, \ldots, \ell_m$ $(r_1, r_2, \ldots, r_{am})$ and use these labels to denote the corresponding ℓ-class (r-class).

Step 2 Construct two tables, the first listing the ℓ-classes of Q and the second the r-classes of Q. Each table has 3 columns. Construct first of all the first two columns of these tables, the first column containing simply a list of the labels ℓ_i (r_j) given to the ℓ-classes (r-classes) of Q, and the second column containing an arbitrary representative element of the corresponding class. The third column of each table is now constructed. In the first table this column represents the various derivatives of Q as unions of r-classes of Q, and in the second table it represents the reverse of the various anti-derivatives of Q as unions of ℓ-classes of Q. Suppose the ℓ-class ℓ_i has representative x_i and the r-class r_j has representative y_j. Then r_j appears as a term in the ℓ_ith row of Table 1 if and only if $x_i y_j \in Q$, and similarly ℓ_i appears as a term in the r_jth row of Table 2 if and only if $x_i y_j \in Q$.

Step 3 Deduce the corresponding left and right factors of Q and label them $L_1, L_2, \ldots, L_q, R_1, R_2, \ldots, R_q$. Find the unique indices s and t such that $Q = L_t = R_s$.

To do this, one considers all subsets $\{\ell_{i_1}, \ldots, \ell_{i_k}\}$ (excluding the empty subset) of the ℓ-classes of Q and finds for each subset those classes $r_{j_1}, \ldots r_{j_n}$ common to the ℓ_{i_1}th, ℓ_{i_2}th, ..., ℓ_{i_k}th entries in the third column of Table 1. One then has $(\ell_{i_1} + \ldots + \ell_{i_k})\cdot(r_{j_1} + \ldots + r_{j_n}) \subseteq Q$ is a subfactorization of Q in which $(r_{j_1} + \ldots + r_{j_n})$ is maximal. By inspecting all such subfactorizations one may deduce the left and right factors. $L_t = \ell_{t_1} + \ell_{t_2} + \ldots + \ell_{t_k}$ is that left factor such that the ℓ-class $\ell_{t_i} \subseteq L_t$ if and only if it corresponds to a terminal node of the machine for Q. Similarly $R_s = r_{s_1} + r_{s_2} + \ldots + r_{s_p}$ is that right factor such that the r-class $r_{s_j} \subseteq R_s$ if and only if it corresponds to a terminal node of the anti-machine for Q.

<u>Step 4</u> Calculate C_{max}. Whence deduce

$$C_{min} = (C_{max}\backslash E)\backslash(C_{max}\backslash E)^{2+*} .$$

$[C_{max}]_{ij} \geqq e$ if and only if $L_j \supseteq L_i$, and this can easily be deduced from the representation of L_i and L_j as unions of ℓ-classes of Q.

<u>Step 5</u> Calculate L_{max}. Whence deduce

$$L_{min} = L_{max}\backslash((C_{max} + L_{max})\backslash E)^{2+*} .$$

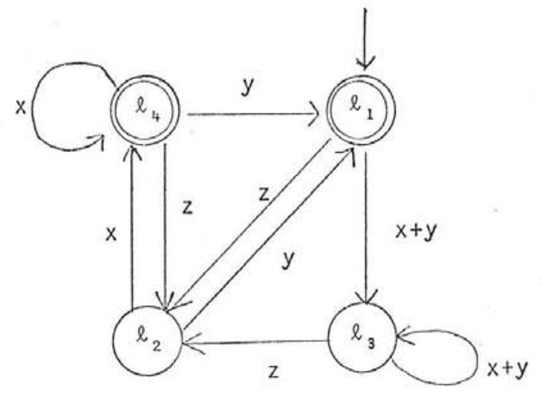
$[L_{max}]_{ij} \geqq a$ if and only if $a \in V$ and $L_i . a \subseteq L_j$. This can also be easily deduced from the representation of L_i and L_j as unions of ℓ-classes of Q and the knowledge that $\ell_k . a \subseteq \ell_j$ if and only if under input a the ℓ_kth state of the machine for Q goes to state ℓ_j.

Finally $G_Q = C_{min} + L_{min}$.

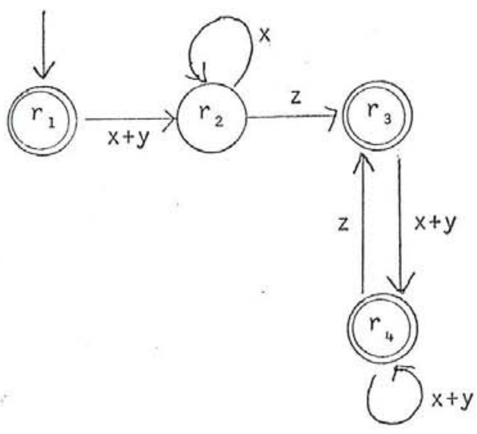
We shall now illustrate the various steps in the above algorithm by a second example.

<u>Example 2</u> $Q = [(x+y)^*zx^*(x+y)]^*$

<u>Step 1</u>



(All-admissible) machine



(All-admissible) anti-machine

Step 2.

ℓ-class	Representative	Derivative
ℓ_1	e	$r_1 + r_3 + r_4$
ℓ_2	z	$r_2 + r_4$
ℓ_3	x	$r_3 + r_4$
ℓ_4	zx	$r_1 + r_2 + r_3 + r_4$

Table 1.

r-class	Representative	Reverse of anti-derivative
r_1	e	$\ell_1 + \ell_4$
r_2	x	$\ell_2 + \ell_4$
r_3	zx	$\ell_1 + \ell_3 + \ell_4$
r_4	xzx	$\ell_1 + \ell_2 + \ell_3 + \ell_4$

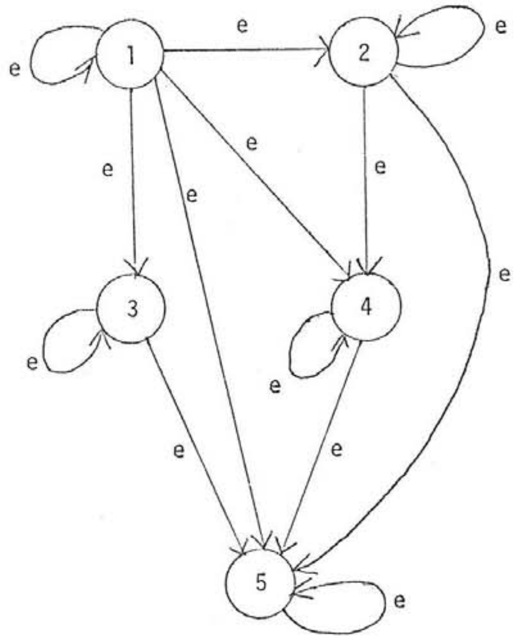
Table 2.

Step 3.

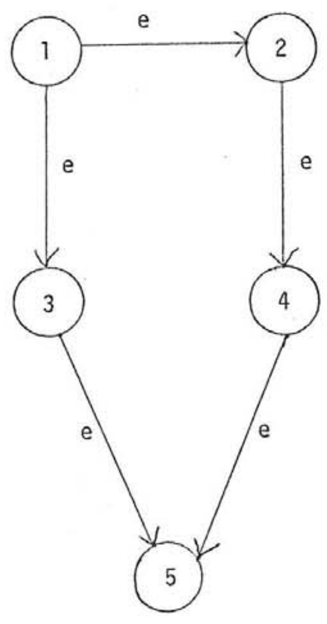
	Left factors			Right factors	
	L_1	ℓ_4		R_1	$r_1 + r_2 + r_3 + r_4$
$L_t =$	L_2	$\ell_1 + \ell_4$	$R_s =$	R_2	$r_1 + r_3 + r_4$
	L_3	$\ell_2 + \ell_4$		R_3	$r_2 + r_4$
	L_4	$\ell_1 + \ell_3 + \ell_4$		R_4	$r_3 + r_4$
	L_5	$\ell_1 + \ell_2 + \ell_3 + \ell_4$		R_5	r_4

Step 4

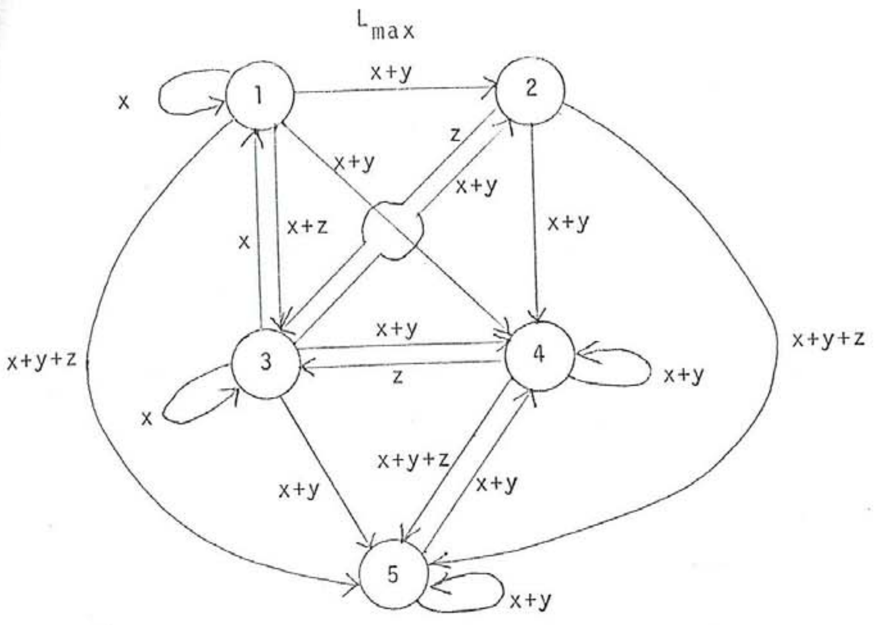
C_{max}



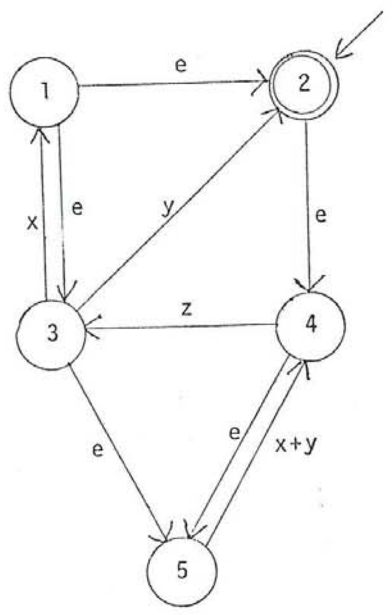
C_{min}



Step 5



(All admissible) Factor Graph $G_Q = C_{min} + L_{min}$



IV CALCULATING THE CLOSURE OF A FACTOR GRAPH

1. Introduction

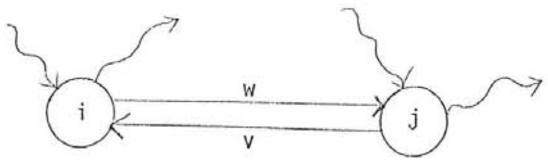
We recall that our original objective in studying Conway's factor matrix was to try to obtain a method of finding the star-height of a given regular language. It is not long, however, before one realises that this cannot be obtained directly from the factor graph for the language Q. Thus for the language $Q = (b + a(aa^*b)^*b)^*$ of Example 1 one obtains directly from the anti-machine for Q (see III §6, fig. 1(b)) the regular expression

$$Q = [(a + b)^*bb + b + e]\ (ab)^*$$

showing that Q has star-height one. However the factor graph of Q (III §6, fig. 3) has rank two.

Yet a very enigmatic feature of factor graphs, observed by examining just a few examples, is that very often one can see ad hoc ways of determining regular expressions for the languages which they recognise, which have star-height less than the rank of the factor graph. The purpose of this chapter is to develop a systematic way of finding the closure G_Q^* of the factor graph G_Q, which does have the property of often yielding expressions of star-height less than the rank of the graph G_Q.

The intuitive approach adopted to tackle this problem is based on the recursive nature of the definition of a language Q in terms of its factors (we use recursive here in the computer scientists sense, not the mathematicians). We observe that if there is a loop



such as the one shown above in the factor graph for Q we would get equations

$$Q_{it} = w \cdot Q_{jt} + \cdots$$
$$Q_{jt} = v \cdot Q_{it} + \cdots$$

in the system of equations which define $Q = Q_{st}$. In this way Q is defined recursively in terms of its factors. The question we ask is "when is a factor necessarily defined in terms of Q?" A clue to answering this question is given by Theorem 2.1 below, due to Conway, which states "factors of factors are themselves factors". This means that the relation "factor of" is a transitive relation, and so it can be naturally reduced to an equivalence relation on the factors, which we call "inseparable from". (Note that this is no different to considering the relation "is connected to" on nodes of a graph, and reducing it to "is strongly connected to", which is an equivalence relation on the nodes.) Examining the properties of factors further (section 3), we prove that the factor matrix of a factor F is a submatrix of $\lceil Q \rceil$, and, moreover, is equal to $\lceil Q \rceil$ if and only if F is inseparable from Q. Having made this observation an algorithm for determining G^*_Q (sections 4 and 5) which exploits separability of factors is then obvious. The remaining sections are then concerned with discussing the applicability of the algorithms to the star-height problem.

2. Inseparable Factors

Theorem 2.1 (Conway) Let Q be any language, and let F be a factor of Q. Then any factor of F is also a factor of Q.

Corollary The relation "factor of" is a reflexive and transitive relation on the factors of any language Q.

Proof If F is a factor of Q it is maximal in some subfactorization $LFR \subseteq Q$ of Q. If H is a factor of F it is also maximal in some subfactorization $GHJ \subseteq F$. But then H is maximal in the subfactorization $LGHJR \subseteq Q$ of Q and so is a factor of Q. The corollary follows because Q is a factor of Q, i.e. the relation is reflexive. (That "factor of" is transitive is merely a restatement of the above theorem.)

Definition 2.2 Let F and H be factors of any language Q. We say F is inseparable from H if and only if F is a factor of H and H is a factor of F. Otherwise we say F and H are separable.

Lemma 2.3 Inseparability is an equivalence relation on the factors of Q.

3. Factor Matrices of Factors

Let the matrix M have nodes $N = \{1,2,\ldots,n\}$ and let $N' \subseteq N$ be any subset of this set. Then we shall call the matrix M' derived from M by simply removing the rows and columns corresponding to nodes $i \notin N'$ the submatrix of M defined by N'. If $N' \neq N$ we say M' is a proper submatrix of M.

Consider, now, any factor F of Q. Then F is some entry Q_{ij} of $\lceil Q \rceil$ (III4.1(i)), and each two term product $Q_{ik} \cdot Q_{kj}$ is, by III4.1(iv), a subfactorization of Q_{ij} (i.e. $Q_{ik} \cdot Q_{kj} \subseteq Q_{ij}$). Moreover, by III4.1(v), all the L.R factorizations of $F = Q_{ij}$ are included in the subfactorizations $Q_{ik} \cdot Q_{kj} \subseteq Q_{ij}$. These observations are highly suggestive that the factor matrix $\lceil F \rceil$ of F is a submatrix of $\lceil Q \rceil$, and, indeed, we shall show in this section that this is the case. Complications arise inevitably in the proof because factors of Q do not necessarily appear uniquely in the factor matrix, but often appear repeatedly (as in example 1).

To avoid confusion we shall henceforth always need to use subscripts or superscripts to identify the factor under consideration. Thus we shall use N_Q to denote the set of nodes of the factor graph G_Q, s_Q and t_Q (where we previously used just s and t) for the nodes mentioned in Theorem III4.1 (iii), and so on.

The proof of the main theorem, that the factor matrix $\lceil F \rceil$ of a factor $F = Q_{ij}$ of Q is a submatrix of $\lceil Q \rceil$, follows from four simple lemmas. The first lemma recognises that F may occur more than once in $\lceil Q\rceil$, and so identifies unique nodes s_F and t_F such that $F = Q_{s_F t_F}$ and which will play the same role in $\lceil F\rceil$ as s_Q and t_Q (see III4.1(iii)). Following this we define a subset N_F of the nodes N_Q of the matrix $\lceil Q\rceil$, and in lemmas 3.3 and 3.4 show that $k\varepsilon\ N_F \Rightarrow Q_{s_F k}\cdot Q_{kt_F} \subseteq Q_{s_F t_F} = F$ is a factorization of F and that these include all the L·R factorizations of F. The final step is to show that if k and $m \varepsilon N_F$, Q_{km} is maximal in $Q_{s_F k}\cdot Q_{km}\cdot Q_{mt_F} \subseteq F$. Then by the definition of $\lceil F\rceil$ the submatrix of $\lceil Q\rceil$ defined by the set of nodes N_F is the matrix $\lceil F\rceil$.

<u>Lemma 3.1</u> Let $F = Q_{ij}$ be a factor of Q. Then $\exists$ indices s_F and t_F such that $F = Q_{s_F t_F}$ and $Q_{s_F s_F}$ and $Q_{t_F t_F}$ are both factors of F.

<u>Proof</u> By III4.1(ii), $L_i Q_{ij} \subseteq L_j$ is a subfactorization in which Q_{ij} is maximal. Let this be dominated by the factorization $L_{s_F}\cdot Q_{s_F j} \subseteq L_j$. (Note that III4.1(v) is being used implicitly here.) Then, by Lemma III3.1, $Q_{ij} = Q_{s_F j}$, and by III4.1(iii) the index s_F is uniquely defined. Now let t_F be that unique index defined by $Q_{s_F t_F}\cdot R_{t_F} \subseteq R_{s_F}$ is a factorization which dominates the subfactorization $Q_{s_F j}\cdot R_j \subseteq R_{s_F}$. Then also $Q_{s_F t_F} = Q_{s_F j}$ and hence $Q_{s_F t_F} = Q_{ij} = F$. To prove the last part, we note that, by construction, $L_{s_F}\cdot Q_{s_F t_F}\cdot R_{t_F} \subseteq Q$ is a factorization of Q. Thus, using III4.1(ii),

$$L_{s_F}\cdot Q_{s_F s_F}\cdot Q_{s_F t_F}\cdot Q_{t_F t_F}\cdot R_{t_F} \subseteq Q \text{ is a factorization,}$$

and hence $Q_{s_F s_F}$ and $Q_{t_F t_F}$ must be maximal in $Q_{s_F s_F} \cdot Q_{s_F t_F} \cdot Q_{t_F t_F} \leq Q_{s_F t_F}$ and so are factors of $Q_{s_F t_F} = F$.

Definition 3.2 The subset N_F of the set N_Q of nodes of the factor matrix $\boxed{Q}$ is defined by $k \varepsilon N_F \Leftrightarrow Q_{kt_F}$ is a factor of F and $L_k \cdot Q_{kt_F} \leq L_{t_F}$ is a factorization of L_{t_F}.

Lemma 3.3 $k \varepsilon N_F \Rightarrow Q_{s_F k} \cdot Q_{kt_F} \leq Q_{s_F t_F}$ is a factorization of F.

Proof By definition $k \varepsilon N_F$ implies $L_k \cdot Q_{kt_F} \leq L_{t_F}$ is a factorization, which implies, by III4.1(ii), that $L_{s_F} \cdot Q_{s_F k} \cdot Q_{kt_F} \leq L_{t_F}$ is a subfactorization in which $Q_{s_F k}$ is maximal.
But by the definition of t_F in the proof of lemma 3.1, $L_{s_F} \cdot Q_{s_F t_F} \leq L_{t_F}$ is a factorization. Therefore, $Q_{s_F k}$ must also be maximal in $Q_{s_F k} \cdot Q_{kt_F} \leq Q_{s_F t_F}$ and so is a left factor of F. Q_{kt_F} is a factor by assumption, so the lemma follows immediately.

Lemma 3.4 If $L^F \cdot R^F \leq F$ is a factorization of F, $\exists$ a unique node $k \varepsilon N_F$ such that $Q_{s_F k} = L^F$ and $Q_{kt_F} = R^F$.

Proof By III4.1(v), $L^F \cdot R^F \leq F = Q_{s_F t_F}$ implies $L^F \leq Q_{s_F p}$ and $R^F \leq Q_{pt_F}$ for some p. Moreover, as L^F and R^F are factors, the last two inequalities must be equalities. Suppose $L_p \cdot Q_{pt_F} \leq L_{t_F}$ is dominated by the factorization $L_k \cdot Q_{kt_F} \leq L_{t_F}$. Now, by III4.1(ii), $Q_{kt_F} = Q_{pt_F} = R^F$, and $L_k \geq L_p \Rightarrow$, by III4.1(ii), $Q_{pk} \geq e \Rightarrow Q_{s_F k} \geq Q_{s_F p} = L^F$. Hence $Q_{s_F k} \cdot Q_{kt_F} \leq F$ dominates $L^F \cdot R^F \leq F$; but, as the latter is a factorization, $Q_{s_F k} = L^F$ and $Q_{kt_F} = R^F$. Moreover, by definition 3.2, $k \varepsilon N_F$. Finally, k is unique follows directly from $Q_{kt_F} = Q_{pt_F}$ and $L_k \cdot Q_{kt_F} \leq L_{t_F}$ is a factorization of L_{t_F}.

Lemma 3.5 Let k and m ε N_F. Then Q_{km} is maximal in $Q_{s_F k} \cdot Q_{km} \cdot Q_{mt_F} \subseteq Q_{s_F t_F}$.

Proof By definition, m ε $N_F \Rightarrow L_m \cdot Q_{mt_F} \subseteq L_{t_F}$ is a factorization; hence, by III4.1(ii), $L_k \cdot Q_{km} \cdot Q_{mt_F} \subseteq L_{t_F}$ is a subfactorization in which Q_{km} is maximal. But $L_k \cdot Q_{kt_F} \subseteq L_{t_F}$ is also a factorization of L_{t_F}, from which we conclude that Q_{km} must be maximal in $Q_{km} \cdot Q_{mt_F} \subseteq Q_{kt_F}$. Thus, by lemma 3.3, Q_{km} is maximal in $Q_{s_F k} \cdot Q_{km} \cdot Q_{mt_F} \subseteq Q_{s_F t_F}$.

Theorem 3.6 F is a factor of Q$\Leftrightarrow$ the factor matrix $\overline{|F|}$ of F is a submatrix of the factor matrix $\overline{|Q|}$ of Q.

Proof Let F be a factor of Q. Then if we compare lemmas 3.3 to 3.5 with the definition of the factor matrix $\overline{|F|}$ of F at the beginning of Section III 4, we see immediately that the submatrix of $\overline{|Q|}$ defined by the set N_F is indeed $\overline{|F|}$. Conversely if $\overline{|F|}$ is a submatrix of $\overline{|Q|}$, F is an entry in $\overline{|Q|}$ and so is a factor of Q (see III4.1(i)).

Corollary 1 Let F and H be two factors of a regular language Q. Then F is inseparable from H

$\Leftrightarrow$ they have the same factor matrix

$\Leftrightarrow$ they have the same factor graph.

Proof F is a factor of H$\Leftrightarrow$ $\overline{|F|}$ is a submatrix of $\overline{|H|}$. H is a factor of F$\Leftrightarrow$ $\overline{|H|}$ is a submatrix of $\overline{|F|}$. Hence F is inseparable from H$\Leftrightarrow$ $\overline{|F|} = \overline{|H|}$. The rest follows from the uniqueness of the factor graph.

Corollary 2 Let F be a regular language, and let C^F_{max} and L^F_{max} be the maximal constant and linear matrices such that $(C^F_{max} + L^F_{max})^* = \overline{|F|}$. Then F is a factor of Q if and only if $C^F_{max} + L^F_{max}$ is a submatrix of $C^Q_{max} + L^Q_{max}$.

Proof If F is a factor of Q, $C^F_{max} + L^F_{max} \subseteq \lceil F \rceil$, which is a submatrix of $\lceil Q \rceil$. Thus by maximality of $C^Q_{max} + L^Q_{max}$, $C^F_{max} + L^F_{max}$ is a submatrix of it. $\Leftarrow$ is obvious.

Finally, we recall that inseparability of factors was defined as a symmetric closure of the relation "factor of". The other "half" of this relation - the anti-symmetric half - is a partial ordering on the classes of inseparable factors, or equivalently, by Corollary 1 of the last theorem, a partial ordering on the factor graphs of factors. This is now defined.

Definition 3.7 Let F and H be two factors of a regular language Q, and let G_F and G_H be their factor graphs. Then we define the relation $\preccurlyeq$ on the factor graphs of factors of Q by

$$G_F \preccurlyeq G_H \quad \text{iff } \lceil F \rceil \text{ is a submatrix of } \lceil H \rceil.$$

Theorem 3.8 $\preccurlyeq$ is a (reflexive) partial ordering on the factor graphs of factors of Q.

The proof is obvious.

4. Example 1 again

The above theorem immediately suggests a new method of calculating $\lceil Q \rceil$ when Q has a factor H which is separable from Q. For, using our knowledge that $\lceil H \rceil$ is a submatrix of $\lceil Q \rceil$, we can write

$$\lceil Q \rceil = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & \lceil H \rceil \end{bmatrix} \tag{1}$$

where C_{11} is a square matrix. The factor graph G_Q can be decomposed into corresponding submatrices

$$G_Q = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{2}$$

Using this notation the escalator method (II § 4.3) is given by the formulae

$$C_{11} = A_{11}^* + A_{11}^* A_{12} \overline{|H|} A_{21} A_{11}^* \tag{3}$$

$$C_{12} = A_{11}^* A_{12} \overline{|H|} \tag{4}$$

$$C_{21} = \overline{|H|} A_{21} A_{11}^* \tag{5}$$

where $\overline{|H|}$ is usually given as $\overline{|H|} = (A_{22} + A_{21} A_{11}^* A_{12})^*$. However $\overline{|H|}$ is the factor matrix of the language H, and so

$$\overline{|H|} = G_H^* \tag{6}$$

where G_H is the factor graph of H.

Formulae (3), (4), (5) and (6) form the basis of an algorithm to compute $\overline{|Q|}$. We shall first use these formulae to calculate the factor matrix of Example 1.

For ease of reference the all-admissible factor graph of $Q = Q_{11}$ is reproduced below.

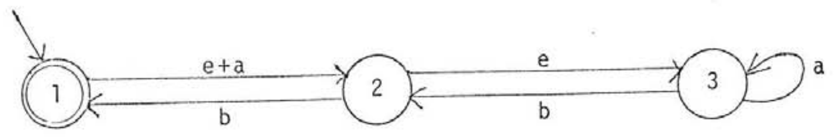


Fig. 1

The first step is to find the factor graphs of factors of Q. We already have an algorithm to do this, contained in the proof of Theorem 3.6. If $F = Q_{ij}$ is a factor of Q, this involves locating other entries $Q_{i'j'}$ which are also equal to F, and using lemma 3.1 to choose s_F and t_F. Then the L R factorizations of F are found by checking whether any one subfactorization $Q_{s_F p} \cdot Q_{p t_F} \subseteq F$ is dominated by another subfactorization $Q_{s_F k} \cdot Q_{k t_F} \subseteq F$; finally definition 3.2 is used to choose those $k \in N_F$. It is here that the representation of each entry in $G_Q^* = \lceil Q \rceil$ as a union of c-classes of Q is particularly useful. For this example we have already shown that G_Q^* may be represented by

$$\begin{bmatrix} c_0 + c_2 + c_3 + c_6 & c_0 + c_1 + c_2 + c_3 + c_4 + c_6 + c_7 & S \\ c_2 + c_6 & c_0 + c_2 + c_3 + c_4 + c_6 + c_7 & S \\ c_6 & c_2 + c_3 + c_6 + c_7 & S \end{bmatrix}$$

(Refer to III§6fig.1(c) for the meaning of $c_0, \dots, c_7$). Using this representation of G_Q^* to determine whether $Q_{ip} \cdot Q_{pj} \subseteq Q_{ij}$ dominates $Q_{ik} \cdot Q_{kj} \subseteq Q_{ij}$ it is only necessary to compare finite subsets of the set of elements in the semigroup against one another.

This example has been chosen for its simplicity. Only one factor, $Q_{33} = Q_{13} = Q_{23}$, appears more than once in the matrix, and this, from fig. 1, is obviously equal to $(a+b)^*$, and so has the factor graph shown below.

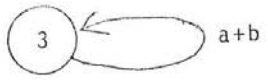


Fig. 2

All other factors Q_{ij} can be easily shown to have the same factor graph as $Q = Q_{11}$. For j=1 and for all i, one need only check that $Q_{i1} \cdot Q_{11} \leq Q_{i1}$ is not dominated by any subfactorization $Q_{ik} \cdot Q_{k1} \leq Q_{i1}$. This is clearly impossible since $Q_{11} = c_0 + c_2 + c_3 + c_6 \supseteq Q_{21} = c_2 + c_6 \supseteq Q_{31} = c_6$. Thus $Q = Q_{11}$ is a factor of Q_{i1}, and so they are inseparable. Similarly Q_{12} can be shown to be inseparable from Q. This only leaves Q_{22}, but since $Q_{12} \supset Q_{22}$ and $Q_{12} \supset Q_{32}$ ($c_1 \leq Q_{12}$, $c_1 \not\leq Q_{22} + Q_{32}$), Q_{12} is a factor of Q_{22}. But Q_{12} is inseparable from Q, hence so is Q_{22}. Now in order to use formulae (3), (4), (5) and (6), we write

$$G_Q = \left[\begin{array}{cc|c} b & e+a & e \\ \hline & b & a \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

A^*_{11} is calculated by a standard elimination method and found to be

$$A^*_{11} = \begin{bmatrix} (b+ab)^* & (b+ab)^*(e+a) \\ (b+ba)^*b & (b+ba)^* \end{bmatrix}$$

The factor H in (3) - (6) is Q_{33}, and its factor matrix is determined from fig.2.

$$\boxed{H} = [(a+b)^*]$$

We now have all the information necessary to apply the formulae (3) - (6), giving

$$\lceil Q \rceil = \begin{bmatrix} (b+ab)^*(e+a)(a+b)^*b(b+ba)^*b + (b+ab)^* & (b+ab)^*(e+a)(a+b)^*b(b+ba)^* + (b+ab)^*(e+a) & (b+ab)^*(e+a)(a+b)^* \\ (b+ba)^*(a+b)^*b(b+ba)^*b + (b+ba)^*b & (b+ba)^*(a+b)^*b(b+ba)^* + (b+ba)^* & (b+ba)^*(a+b)^* \\ (a+b)^*b(b+ba)^*b & (a+b)^*b(b+ba)^* & (a+b)^* \end{bmatrix}$$

The regular expressions appearing in $\lceil Q \rceil$ could be made much simpler had we used the knowledge that $Q_{33} = Q_{31} = Q_{32} = (a+b)^*$. However this is irrelevant to our aim which is simply to obtain regular expressions of smallest star-height. Indeed for this example we have achieved this aim, since all the expressions appearing in $\lceil Q \rceil$ are of star-height one. Moreover this is <u>strictly less than</u> the rank of the factor graph. The final expression for Q is

$$Q = Q_{11} = (b+ab)^*(e+a)(a+b)^*b(b+ba)^*b+(b+ab)^*$$

which simplifies to

$$Q = (a+b)^*b(b+ba)^*b+(b+ab)^* .$$

5. <u>An Algorithm for Calculating $\boxed{Q}$</u>

We shall now formulate the algorithm for calculating $\lceil Q \rceil$. In general the algorithm is not quite as simple as in the example above. In the above example all the factor graphs G_H of factors of Q were totally ordered by the relation $\preccurlyeq$ (there were only two!). Technical difficulties arise in the algorithm because in general the relation $\preccurlyeq$ is a partial ordering on the factor graphs associated with Q.

<u>Algorithm 2.</u> To calculate $\lceil Q \rceil$ for a given regular language Q.

<u>Step 1</u> Find the factor graphs G_H of all factors H of Q (including G_Q). A method of doing this is given in the next section. Associate with each factor graph G_H a subset $N_H = \{i_1, i_2, \ldots, i_h\}$ of the nodes $\{1, 2, \ldots, q\}$ of the factor graph G_Q, where the submatrix of $\lceil Q \rceil$ defined by the set N_H is the factor matrix $\lceil H \rceil$ of H. Note that for some factors H there may be more than one submatrix of $\lceil Q \rceil$ which is equal to $\lceil H \rceil$ (see e.g. the next example). For the purposes of exposition, we shall assume these factor graphs to be distinct.

<u>Step 2</u> Calculate the upper semi-lattice defined by the partial ordering $\preccurlyeq$ on the distinct factor graphs G_H of factors H of Q. We shall call G_H a <u>minimal</u> element of this lattice if there is no other factor graph G_F such that $G_F \prec G_H$. G_Q is of course the only maximal element.

<u>Step 3</u> Choose any path $G_Z \prec G_Y \ldots\ G_S \prec G_R \prec G_Q$ from a minimal element G_Z of the semi-lattice to the maximal element G_Q. This defines a sequence $N_Z \subset N_Y \subset \ldots \subset N_S \subset N_R \subset N_Q$ of the nodes of G_Q. Reorder the nodes of G_Q such that the nodes in the set $N_Q \backslash N_R$ are numbered from 1 to $|N_Q \backslash N_R|$, the nodes of $N_R \backslash N_S$ are numbered from $|N_Q \backslash N_R| + 1$ to $|N_Q \backslash N_S|$ etc. Within any of sets $N_R \backslash N_S$ the order is immaterial.

<u>Step 4</u> For the minimal element G_Z of the path calculate $G_Z^* = \lceil Z \rceil$ using a standard elimination method.

Step 5 Suppose the current factor matrix that has been calculated is $G_H^* = \lceil H \rceil$. If $G_H = G_Q$, stop; otherwise let G_F be the next point in the chain. Split G_F as shown below

$$G_F = \left[\begin{array}{c|c} A_{FF} & A_{FH} \\ \hline A_{HF} & A_{HH} \end{array}\right] \begin{array}{l} \left.\right\} \text{Nodes in } N_F \backslash N_H \\ \left.\right\} \text{Nodes in } N_H \end{array} \Bigg\} \text{Nodes in } N_F$$

and correspondingly define C_{FF}, C_{FH}, C_{HF} and C_{HH} by

$$G_F^* = \lceil F \rceil = \left[\begin{array}{c|c} C_{FF} & C_{FH} \\ \hline C_{HF} & C_{HH} \end{array}\right]$$

Compute A_{FF}^* using a standard elimination method.

Compute all entries of G_F^* using

$$C_{HH} = \lceil H \rceil \qquad \text{(which has already been calculated)}$$
$$C_{FF} = A_{FF}^* + A_{FF}^* A_{FH} \lceil H \rceil A_{HF} A_{FF}^*$$
$$C_{FH} = A_{FF}^* A_{FH} \lceil H \rceil$$
$$C_{HF} = \lceil H \rceil A_{HF} A_{FF}^* .$$

Step 6 Repeat Step 5.

The above algorithm requires that one calculate the factor graphs of factors of Q. Once the factor graph of Q has been calculated it is not necessary to repeat all the steps of the algorithm given in section 7 of Chapter III to find the factor graph of any factor F of Q. Instead one essentially uses the proof of lemma 3.1 to find nodes s_F and

and t_F such that $F = Q_{s_F t_F}$, and then definition 3.2 and corollary 2 to theorem 3.6 enable one to deduce $C^F_{max} + L^F_{max}$ directly from $C^Q_{max} + L^Q_{max}$. For ease of reference the steps in this algorithm are given below.

<u>Algorithm 3</u> To calculate factor graphs G_H of factors H of Q.

Suppose $H = Q_{ij}$ is the (i,j)th entry of $\lceil Q \rceil$.

<u>Step 1</u> Calculate $C^Q_{max} + L^Q_{max}$ and deduce G_Q.

<u>Step 2</u> Calculate $(C^Q_{max} + L^Q_{max})^*$ in the algebra $M_q(\mathcal{R}(\underline{S}_Q))$. ($\mathcal{R}(\underline{S}_Q)$ is the regular algebra generated by the semigroup $\underline{S}_Q$ of the language Q). In other words calculate each entry of $\lceil Q \rceil$ as a union of c-classes of Q. Let this matrix be denoted $\lceil C(Q) \rceil$.

In the following steps, in order to check that $Q_{k\ell} \geq Q_{mn}$, one checks that

$$C(Q)_{k\ell} = c_{i_1} + c_{i_2} + \dots + c_{ix} \geq C(Q)_{mn} = c_{j_1} + c_{j_2} + \dots + c_{jy}.$$

This involves comparing two finite sets for set inclusion.

<u>Step 3</u> Consider $H = Q_{ij}$, and consider all nodes i' such that $Q_{ij} = Q_{i'j}$. Let s_H be that node i' such that L_{s_H} is maximal. Now consider $Q_{s_H j}$ and all nodes j' such that $Q_{s_H j} = Q_{s_H j'}$. Let t_H be that node j' such that R_{t_H} is maximal.

<u>Step 4</u> We now have $H = Q_{s_H t_H}$. Compare all subfactorizations $Q_{s_H k} \cdot Q_{k t_H} \leq H$ and $Q_{s_H m} \cdot Q_{m t_H} \leq H$ for one dominating the other; thus deduce the right factors $Q_{k t_H}$ of H.

<u>Step 5</u> For all k such that $Q_{k t_H}$ is a right factor of H, let k' be that node such that $Q_{k t_H} = Q_{k' t_H}$ and $L_{k'}$ is maximal.

Let N_H be the set of all such k'.

Step 6 $C^H_{max} + L^H_{max}$ is the submatrix of $C^Q_{max} + L^Q_{max}$ defined by the set of nodes N_H. Calculate G_H using

$$G_H = ((C^H_{max} + L^H_{max})\backslash E)\backslash((C^H_{max} + L^H_{max})\backslash E)^{2+*}.$$

Needless to say in practical applications it is not necessary to go to quite these lengths to calculate G_H, and various ad hoc techniques, such as were used in example 1, can be acquired with practice.

6. Two More Examples

Consider $Q = a(a+b)^*b(a+b)^*a$.

We shall apply algorithm 2 to determine the factor matrix $|\overline{Q}|$.

Step 1 The factor graph and semigroup of Q are shown below.

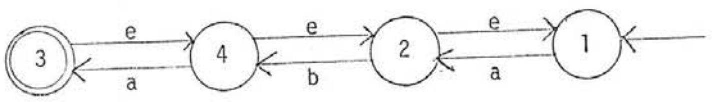


Fig. 3 Factor Graph of $Q = Q_{13}$.

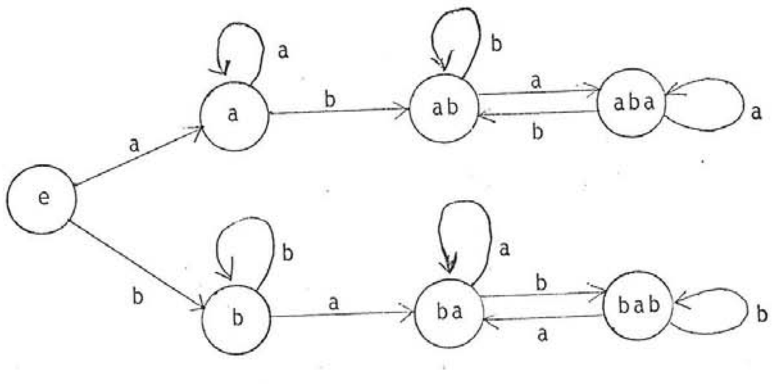


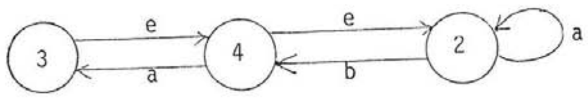
Fig. 4 Semigroup of Q.

The matrix $\overline{|C(Q)|}$ which exhibits each entry of $\overline{|Q|}$ as a union of c-classes of Q is easily found to be:

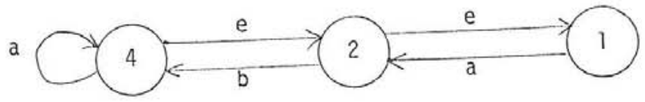
$$\overline{|C(Q)|} = \begin{bmatrix} e+a+ab+aba & a+ab+aba & aba & aba+ab \\ S & S & aba+ba & b+ab+ba+bab+aba \\ S & S & e+a+ba+aba & S \\ S & S & a+ba+aba & S \end{bmatrix}$$

where S denotes the whole semigroup.

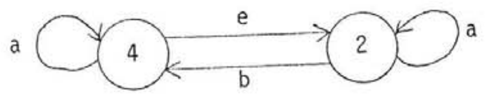
Applying algorithm 3 we can deduce the following factor graphs for factors Q_{ij} of Q.



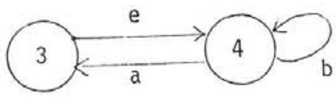
(a) Factor graph of Q_{23}



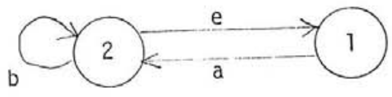
(b) Factor Graph of Q_{14}



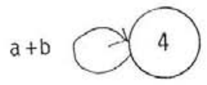
(c) Factor graph of Q_{24}



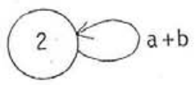
(d) Factor Graph of Q_{43}



(e) Factor Graph of Q_{12}



(f) Factor Graph of Q_{44}



(g) Factor Graph of Q_{22}

Fig. 5

<u>Step 2</u> The semi-lattice defined by the relation $\precsim$ is shown graphically below. Each node contains a representative element of the class of inseparable factors to which the node corresponds, together with the set of nodes which define the submatrix of $\boxed{Q}$ which equals the particular factor matrix.

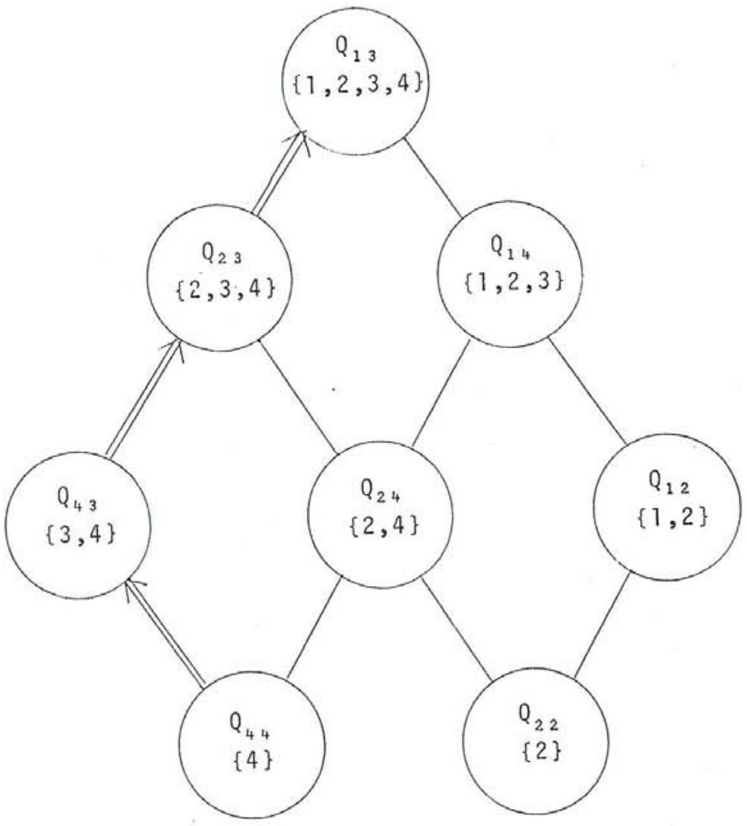


Fig. 6 Semi-lattice $\precsim$

Step 3 The particular path from bottom to top of this semilattice, which we will use in the calculation of $\lceil Q \rceil$, has been arrowed. The node numbering has already been chosen to meet the requirements of step 3 of algorithm 1.

Step 4 Clearly $\lceil Q_{44} \rceil = [(a+b)^*]$. (from fig. 5(f)).

Step 5 Repeating step 5 of the algorithm we get successively

$$\lceil Q_{43} \rceil = \begin{bmatrix} e+(a+b)^*a & (a+b)^* \\ (a+b)^*a & \lceil Q_{44} \rceil \end{bmatrix} \quad \text{(from fig. 5(d))}$$

$$\lceil Q_{23} \rceil = \begin{bmatrix} a^*+a^*b(a+b)^*a^* & a^*b(a+b)^*a & a^*b(a+b)^* \\ (a+b)^*a^* & \lceil Q_{43} \rceil & \\ (a+b)^*a^* & & \end{bmatrix}$$

(from fig.5(a))

$$\lceil Q \rceil = \begin{bmatrix} e+a(a^*+a^*b(a+b)^*a^*) & a(a^*+a^*b(a+b)^*a^*) & aa^*b(a+b)^*a & aa^*b(a+b)^* \\ a^*+a^*b(a+b)^*a^* & & & \\ (a+b)^*a^* & & \lceil Q_{23} \rceil & \\ (a+b)^*a^* & & & \end{bmatrix}$$

(From Fig.3).

In each of these matrices we have only shown the new entries in the matrix. The final expression for Q is $Q_{13} = aa^*b(a+b)^*a$.

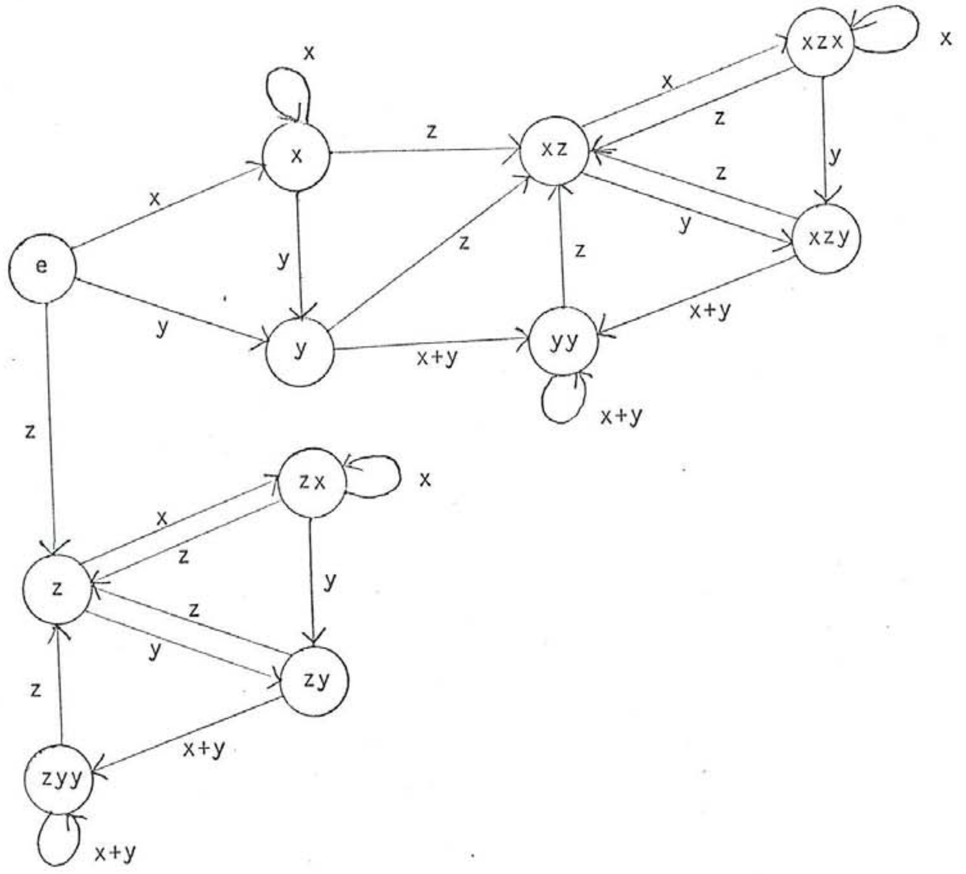
Remarks. This example is instructive for two reasons. Firstly it illustrates that in general one has a choice of path through the semilattice. Different paths will usually give different regular expressions for the language Q, although in this case all paths yield expressions of the same star height.

Whether there are examples where two different paths through the semi-lattice yield different star-height expressions I do not know. In any case one can always determine the star height that a particular path will give and choose one which is optimal. Secondly, all regular expressions appearing in $\boxed{Q}$ are of star height one, yet the rank of the factor graph is two! Indeed we shall show later that the algorithm always yields regular expressions having star-height less than or equal to the rank of the factor graph of Q.

<u>Example 2</u> (continued)

Let us return to example 2 (see pages 111-114). The language considered is $Q = [(x+y)^*zx^*(x+y)]^*$, and its factor graph is reproduced in fig. 8(b) below. As we are only interested in deriving a regular expression for the language Q, which is the (2,2)th entry of the factor matrix, we shall not calculate the whole factor matrix using algorithm 2 but only Q_{22}. To do this we apply steps 1 to 3 of algorithm 2 as before, but then apply steps 4 and 5 in the reverse order. This results in a system of equations for Q_{22} which can then be solved to deduce a regular expression for the language $Q = Q_{22}$.

The semigroup of Q is shown in Fig. 7, and in table 1 we show the factor matrix as a union of congruence classes of Q.



Fig, 7. Semigroup of Q.

$$\begin{bmatrix}
\begin{array}{c} e \\ +zy+zx \\ +xzx+xzy \end{array} & zx+xzx & \begin{array}{c} z+zx \\ +xz+xzx \end{array} & M-xz & M+z \\
\begin{array}{c} e+x \\ +zx+xzx \\ +y+zy+xzy \end{array} & \begin{array}{c} e+x \\ +zx+xzx \end{array} & \begin{array}{c} e+x \\ +zx+xzx \\ +z+xz \end{array} & M-xz & M+z \\
\begin{array}{c} y+x \\ +xzx+xzy \end{array} & \begin{array}{c} x \\ +xzx \end{array} & \begin{array}{c} e+x \\ +xz+xzx \end{array} & N-e-xz & N \\
\begin{array}{c} zy+zx \\ +xzx+xzy \end{array} & zx+xzx & \begin{array}{c} z \\ +xz+xzx \end{array} & M-xz & M+z \\
xzx+xzy & xzx & xz+xzx & N-e-xz & N
\end{bmatrix}$$

where

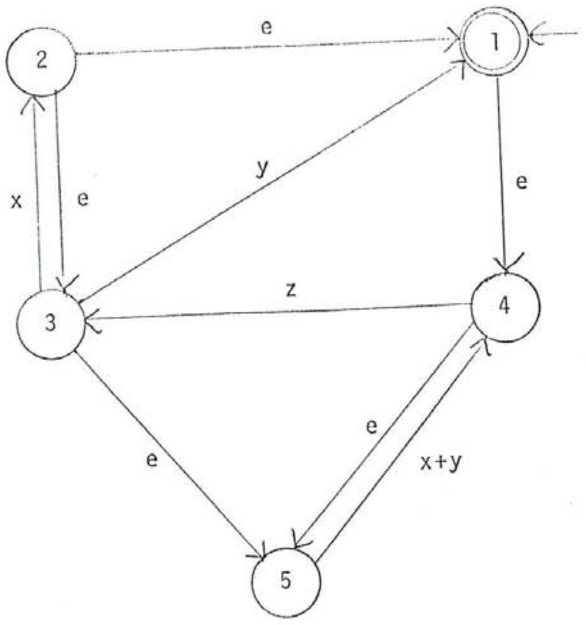
$M = e + x + y + xz + xzx + xzy + zx + zy + zyy + yy$

$N = e + x + y + xz + xzx + xzy + yy$.

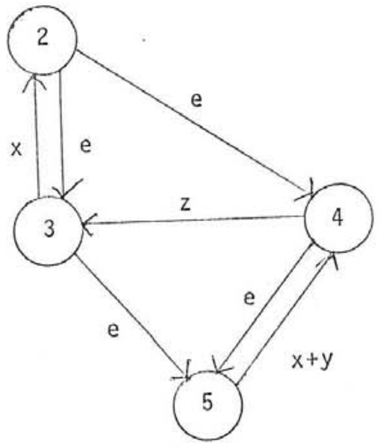
Table 1. The Matrix $\lceil C(Q) \rceil$

We now find that there are three factor graphs associated with the language Q. The first is the factor graph of Q, which for convenience we have reproduced below, and the second and third are factor graphs for the languages $\{Q_{33}, Q_{34}, Q_{43}\}$ and $\{Q_{44}, Q_{54}, Q_{45}, Q_{55}\}$, respectively.

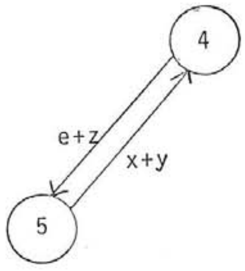
The ordering $\preccurlyeq$ on the factor graphs is total and so there is no question of a choice of path through the semi-lattice.



(a) Factor Graph of Q



(b) Factor Graph of Q_{33}, Q_{34}, Q_{43}



(c) Factor Graph of Q_{44}, Q_{45}, Q_{54} and Q_{55}

Fig. 8

Applying the last step of algorithm 2 requires us to split G_Q in the manner indicated below:

$$G_Q = \left[\begin{array}{c|cccc} & & & e & \\ \hline e & & e & & \\ y & x & & & e \\ & & z & & e \\ & & & x+y & \end{array}\right] = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

(Cf. figs. 8(a) and (b)).

$Q = [G_Q^*]_{11}$ is then calculated as

$$Q = A^*_{11} + [A^* \cdot B \cdot G_H^* \cdot C \cdot A^*]_{11}$$

where G_H is the second factor graph, i.e.

$$G_H = \begin{bmatrix} & e & e & \\ x & & & e \\ & z & & e \\ & & x+y & \end{bmatrix} .$$

A^*_{11} is just e, and so we can write down an equation for Q, viz:

$$\begin{aligned} Q &= \underbrace{e}_{A^*_{11}} + \underbrace{e}_{A^*_{11}} \cdot \underbrace{e}_{B_{14}} \cdot \underbrace{Q_{43}}_{[G_H^*]_{43}} \cdot \underbrace{y}_{C_{31}} \cdot \underbrace{e}_{A^*_{11}} \\ &\quad + \underbrace{e}_{A^*_{11}} \cdot \underbrace{e}_{B_{14}} \cdot \underbrace{Q_{42}}_{[G_H^*]_{42}} \cdot \underbrace{e}_{C_{21}} \cdot \underbrace{e}_{A^*_{11}} \\ &= e + Q_{43}y + Q_{42} . \end{aligned} \qquad (1)$$

In the above equation we have indicated how each term arises. All other terms in the product $A^*\cdot B\cdot G_H^*\cdot C\cdot A^*$ are null. In order to calculate Q_{43} and Q_{42} we apply the same procedure to G_H. (See figs. 8(b) and (c)). First we write

$$G_H = \left[\begin{array}{cc|cc} & e & e & \\ x & & & e \\ \hline & z & & e \\ & & x+y & \end{array}\right] = \left[\begin{array}{c|c} K & L \\ \hline M & N \end{array}\right] \text{, say.}$$

Q_{43} and Q_{42} are then calculated using

$$Q_{43} = [G_F^*\cdot M\cdot K^*]_{43}$$

$$Q_{42} = [G_F^*\cdot M\cdot K^*]_{42}$$

where G_F is the minimal factor graph (fig. 8(c)):

$$G_F = \begin{bmatrix} & e+z \\ x+y & \end{bmatrix} .$$

By inspection $K^* = \begin{bmatrix} x^* & x^* \\ x^*x & x^* \end{bmatrix}$

$$\text{thus } Q_{43} = \underbrace{Q_{44}}_{[G_F^*]_{44}}\cdot\underbrace{z}_{M_{43}}\cdot\underbrace{x^*}_{K_{33}^*} = Q_{44}zx^* \qquad (2)$$

$$\text{and } Q_{42} = \underbrace{Q_{44}}_{[G_F^*]_{44}}\cdot\underbrace{z}_{M_{43}}\cdot\underbrace{x^*x}_{K_{32}^*} = Q_{44}zxx^* . \qquad (3)$$

Finally Q_{44} is calculated directly from the factor graph G_F (Fig. 8(c)). One easily obtains

$$Q_{44} = (x+y+zx+zy)^* \quad . \tag{4}$$

Using back-substitution, equations (1), (2), (3) and (4) are solved to give

$$\begin{aligned} Q &= e+(x+y+zx+zy)^*zx^*y \\ &\quad +(x+y+zx+zy)^*zx^*x \\ &= e+(x+y+zx+zy)^*zx^*(x+y) \quad . \end{aligned}$$

Note that once again we obtain an expression for Q which has star-height strictly less than the rank of the factor graph.

7. Final Theorem

We have observed in the previous examples that the algorithm for determining the closure G_Q^* yields expressions for Q which are of star-height strictly less than the rank of the factor graph G_Q. The algorithm requires that one use an elimination method to determine certain closures A_{FF}^* and the closure G_H^* of a factor graph G_H which is minimal with respect to the ordering $\preccurlyeq$. We shall now prove that, provided the order of elimination of nodes used in the determination of the various matrices A_{FF}^* and G_H^* is optimal with respect to the star-height of the resulting regular expressions, the algorithm always yields expressions for Q of star-height less than or equal to the rank of the factor graph G_Q of Q. The proof follows rather simply from the following theorem.

Theorem 7.1 Let Q be a regular language with factor graph G_Q, and let H be a factor of Q, with factor graph G_H. Then $\mathrm{rank}(G_H) \le \mathrm{rank}(G_Q)$.

We shall in fact prove more than this, namely that for any factor H of Q there is some graph G_H' such that $G_H \subseteq G_H' \subseteq C^H_{max} + L^H_{max}$, and G_Q is pathwise homomorphic to G_H'.

Suppose that the graph G_Q has nodes N_Q, that $N_H \subseteq N_Q$ is the set of nodes of G_H, and $H = Q_{ij}$ where $i,j \in N_H$ (i.e. $i = s_H$ and $j = t_H$). The next lemma is a necessary preliminary to defining a mapping $\gamma\colon N_Q \to N_H$.

Lemma 7.2 Let $p \in N_Q$. Then $\exists$ a unique node $m_p \in N_H$ such that (i) $Q_{im_p}\cdot Q_{m_pj} \subseteq Q_{ij}$ dominates $Q_{ip}\cdot Q_{pj} \subseteq Q_{ij}$ and

(ii) if $m' \in N_H$ also has the property that

$Q_{im'}\cdot Q_{m'j} \subseteq Q_{ij}$ dominates $Q_{ip}\cdot Q_{pj} \subseteq Q_{ij}$

then

(a) $Q_{im_p} \supseteq Q_{im'}$ and (b) $Q_{m'm_p} \supseteq e$.

Proof Let $\{n_1, n_2, \ldots n_r\} \subseteq N_H$ be all those nodes in N_H such that $Q_{in_k}\cdot Q_{n_kj} \subseteq Q_{ij}$ dominates $Q_{ip}\cdot Q_{pj} \subseteq Q_{ij}$, $k = 1,2,\ldots r$. (Obviously the set is non-empty.)

Then $(Q_{in_1} + Q_{in_2} + \ldots + Q_{in_r})\cdot(Q_{n_1j} \cap Q_{n_2j} \cap \ldots \cap Q_{n_rj}) \subseteq Q_{ij}$ dominates $Q_{ip}\cdot Q_{pj} \subseteq Q_{ij}$, and is itself dominated by $Q_{im_p}\cdot Q_{m_pj} \subseteq Q_{ij}$ for some $m_p \in N_F$. But m_p clearly satisfies (i) and (ii) (a). Part (ii) (b), follows directly from Theorem III 4.1(ii), since

$Q_{im'}$ and Q_{im_p} are both left factors of $H = Q_{ij}$ and $Q_{im_p} \supseteq Q_{im'}$.e by (ii) (a). Finally uniqueness of m_p follows from (ii) (b) and the acyclicity of $C^H_{max}\backslash E$ (Lemma III 5.4).

Now we may define a mapping $\gamma:N_Q \to N_H$ by:$\gamma(p) = m_p$ where m_p for any $p\varepsilon N_Q$ is the unique node of N_H defined by lemma 7.2 above.

We now extend γ to be a pathwise homomorphism in a rather trivial manner. We define the graph $G_H^{\prime}$ to have nodes N_H, and an arc labelled a from node k to node m $(k,m\varepsilon N_H)$ if and only if there is an arc a from some node $p\varepsilon\gamma^{-1}(k)$ to some node $r\varepsilon\gamma^{-1}(m)$ in the graph G_Q. Finally γ is extended to be a mapping from arcs of G_Q into arcs of $G_H^{\prime}$ by : if a labels an arc from node p to node r in G_Q then γ maps it into the arc labelled a from $\gamma(p)$ to $\gamma(r)$ in $G_H^{\prime}$.

By the construction of $G_H^{\prime}$, γ is a pathwise homormorphism and so we deduce from McNaughton's pathwise homormorphism theorem that:

<u>Lemma 7.3</u> $\text{rank}(G_H^{\prime}) \leq \text{rank}(G_Q)$.

Lemma 7.5 will state that $G_H \subseteq G_H^{\prime}$, from which Theorem 7.1 follows immediately. In order to prove this we prove the following lemma.

<u>Lemma 7.4</u> Let p and r be any two nodes of G_Q and let $\gamma(p) = k$ and $\gamma(r) = m$. Then $Q_{km} \supseteq Q_{pr}$.

<u>Proof</u> Consider the subfactorization $Q_{ip}\cdot Q_{pj} \subseteq Q_{ij}$ which is dominated by $Q_{ik}\cdot Q_{kj} \subseteq Q_{ij}$ (by the definition of γ and lemma 7.2(i)).

Now $Q_{kj} \supseteq Q_{pj} \supseteq Q_{pr} \cdot Q_{rj}$.

Hence $\exists\ m' \varepsilon N_H$ such that

$$Q_{pr} \subseteq Q_{km'} \tag{1}$$

and

$$Q_{rj} \subseteq Q_{m'j} \ . \tag{2}$$

But $Q_{rj} \subseteq Q_{m'j} \Rightarrow (Q_{ir} + Q_{im'}) \cdot Q_{rj} \subseteq Q_{ij}$.

Suppose this subfactorization is dominated by $Q_{im''} \cdot Q_{m''j} \subseteq Q_{ij}$ where $m'' \varepsilon N_H$. Then $Q_{im''} \supseteq Q_{im'}$ and as these are both left factors of $H = Q_{ij}$, by III4.1(ii),

$$Q_{m'm''} \supseteq e \ . \tag{3}$$

But, also, $Q_{im''} \cdot Q_{m''j} \subseteq Q_{ij}$ dominates $Q_{ir} \cdot Q_{rj} \subseteq Q_{ij}$.

$\therefore$, by lemma 7.2 (ii) (b),

$$Q_{m''m} \supseteq e \ . \tag{4}$$

Now, by (3) and (4), $Q_{m'm} \supseteq e$. (5)

Hence $Q_{km} \supseteq Q_{km'} \cdot Q_{m'm} \supseteq Q_{km'}$, by (5)

$\supseteq Q_{pr}$, by (1).

<u>Lemma 7.5</u> $G_H \subseteq G'_H$

<u>Proof</u> In order to prove this lemma, we first prove

(a) $G'_H \subseteq C^H_{max} + L^H_{max}$, and

(b) $G^*_H \subseteq G'^*_H$.

(a) Suppose $a \ \varepsilon\ [G'_H]_{km}$. Then by construction of G'_H, $\exists$ nodes p and r $\varepsilon\ N_Q$ such that $\gamma(p)=k, \gamma(r)=m$ $(k, m \varepsilon N_H)$, and $a \ \varepsilon\ [G_Q]_{pr}$.

Thus, by lemma 7.4, $a \in Q_{km}$. Hence $a \in [C^H_{max} + L^H_{max}]_{km}$.
I.e. $G'_H \subseteq C^H_{max} + L^H_{max}$.

(b) Now let us prove $G^*_H \subseteq G'^*_H$. Since G^*_H is a submatrix of $\overline{|Q|} = G^*_Q$, a word $w \in [G^*_H]_{km}$ if and only if there is a sequence of nodes $k = p_0, p_1, \ldots p_n = m$ with each $p_r \in N_Q$ and such that $w = a_1 a_2 \ldots a_n$, where $a_r \in [G_Q]_{p_{r-1}p_r}$.
But then $a_r \in [G'_H]_{\gamma(p_{r-1})\gamma(p_r)}$, and $\gamma(p_0) = k$, and $\gamma(p_n) = m$.
I.e. $w \in [G'^*_H]_{km}$. Thus $G^*_H \subseteq G'^*_H$.

Now from (a) and (b) and Theorems III 5.6 and 5.2, $\overline{|H|} = G^*_H \subseteq G'^*_H \subseteq (C^H_{max} + L^H_{max})^* = \overline{|H|}$. Hence $G^*_H = G'^*_H = \overline{|H|}$.
Finally, since G_H is the unique minimal starth root of $\overline{|H|}$ (Theorem III 5.6), $G_H \subseteq G'_H$ and the lemma is proved.

We may now deduce Theorem 7.1 directly from lemmas 7.3 and 7.5 since $G_H \subseteq G'_H$ trivially implies $\text{rank}(G_H) \leq \text{rank}(G'_H)$, which by lemma 7.3, $\leq \text{rank}(G_Q)$.

Corollary (to Theorem 7.1) With a suitable ordering of the nodes of the factor graph G_Q, algorithm 2 yields a regular expression for the language Q which is of star-height less than or equal to the rank of the factor graph G_Q.

Proof The algorithm requires that one determine G^*_H for a factor graph G_H which is minimal with respect to the ordering $\preccurlyeq$. By the above corollary the rank of G_H does not exceed the rank of G_Q and so with a suitable ordering of its nodes the escalator method yields regular expressions for the entries of

G_H^* all of which have star height not exceeding the rank of G_Q. Also required is that one determine A_{FF}^* for a number of graphs A_{FF}. Each such graph is a subgraph of a factor graph G_F and hence also has rank not exceeding the rank of G_Q. Thus once again one can obtain regular expressions for the entries in A_{FF}^* of star-height less than or equal to the rank of G_F. Since these are the only two cases where starred expressions are introduced by the algorithm the corollary holds.

As we have already demonstrated in examples 1, 2 and 3 the regular expressions obtained by the algorithm may well have star height _strictly_ less than the rank of G_Q.

The proof of Theorem 7.1 is very useful in hand calculations to search for factor graphs of factors of Q which are separable from Q. The idea is to endeavour to eliminate nodes from the factor graph G_Q by "coalescing" them with other nodes of the factor graph. To eliminate node p by "coalescing" it with node k, one simply converts any arc a from some node i to node p into an arc a from node i to node k, and any arc a from node p to some node j into an arc a from node k to node j. Suppose after eliminating a number of nodes from G_Q, this results in a graph G' having nodes $N' \subseteq N_Q$.

The next step is to "prune off" arcs of G', i.e. in effect construct $G = (G'\backslash E)\backslash(G'\backslash E)^{2+*}$. This graph G will then be a factor graph only if $G \leq (C_{max}^Q + L_{max}^Q)$, which can easily be checked by inspection. Note, however, that G is not necessarily a factor graph of Q (see example 7, section 8), although all factor graphs can be obtained in this way.

Thus the method is not fail-safe and ultimately resort must be made to Algorithm 3. Nevertheless it is undoubtedly a useful aid to rough calculations.

Example 3 illustrates the process quite well. In order to obtain the factor graph of Q_{23} one eliminates node 1 by coalescing it with node 2, the graph for Q_{14} is obtained by coalescing node 3 with node 4, the graph for Q_{24} by coalescing node 3 with node 4 and node 1 with node 2, and so on.

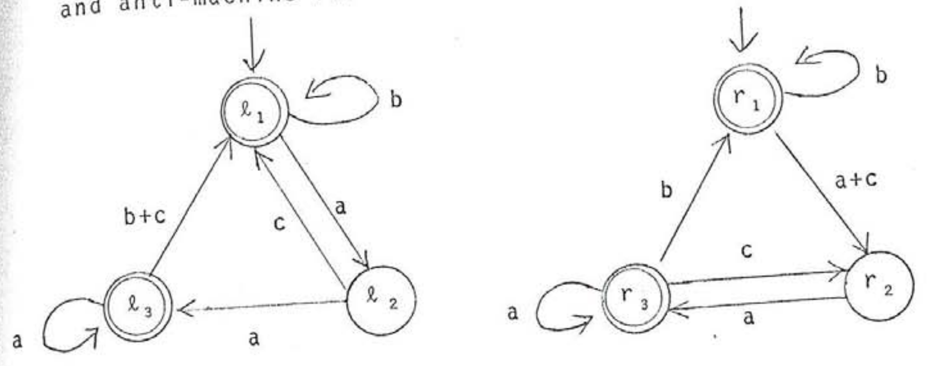
8. Empirical Results

We would have liked, of course, to end this chapter with a theorem to the effect that algorithm 2 always yields a minimal star-height expression for the language Q. Indeed for a long time we thought that this could well be true. Just by taking a very large selection of regular languages which have appeared in the literature, and laboriously calculating factor graphs we achieved almost 100% success in arriving at minimal star-height forms for these languages.

If algorithm 2 did always yield a minimal star-height expression for any regular language Q, a necessary condition would be that, for those languages Q all of whose factors are inseparable from Q, the star-height of Q would equal the rank of the factor graph of Q. In our empirical investigation we eventually found an example which appeared in McNaughton's paper [29] which refuted this condition, thus showing that algorithm 2 does not always give the minimal star-height expression for a language Q. The example follows.

Example 4 (Refutation of conjecture that algorithm 2 always yields a minimal star-height expression.)

Consider $Q = (b + aa + ac + aaa + aac)^*$. The machine and anti-machine for this language are shown below.



Machine Anti-machine

Fig. 9

If we use algorithm 1 we find that

$$D_{\ell_1}Q = r_1 + r_3$$

$$D_{\ell_2}Q = r_2 + r_3$$

and $D_{\ell_3}Q = r_1 + r_2 + r_3$.

Thus the L.R factorizations of Q are

$$\begin{aligned} L_1 \times R_1 &= (\ell_1 + \ell_2 + \ell_3) \times r_3 \\ L_t = L_2 \times R_2 = R_s &= (\ell_1 + \ell_3) \times (r_1 + r_3) \\ L_3 \times R_3 &= (\ell_2 + \ell_3) \times (r_2 + r_3) \\ L_4 \times R_4 &= \ell_3 \times (r_1 + r_2 + r_3) \quad . \end{aligned}$$

The factor graph of Q can now be determined:

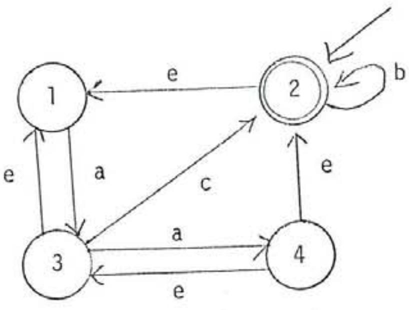


Fig. 10

In order to determine whether Q has any factors whose factor matrix is a submatrix of $\lceil Q \rceil$ we calculate the semigroup of Q. The semigroup machine is shown in the next diagram in which nodes, or equivalently c-classes of Q, are labelled by a representative element of the corresponding c-class.

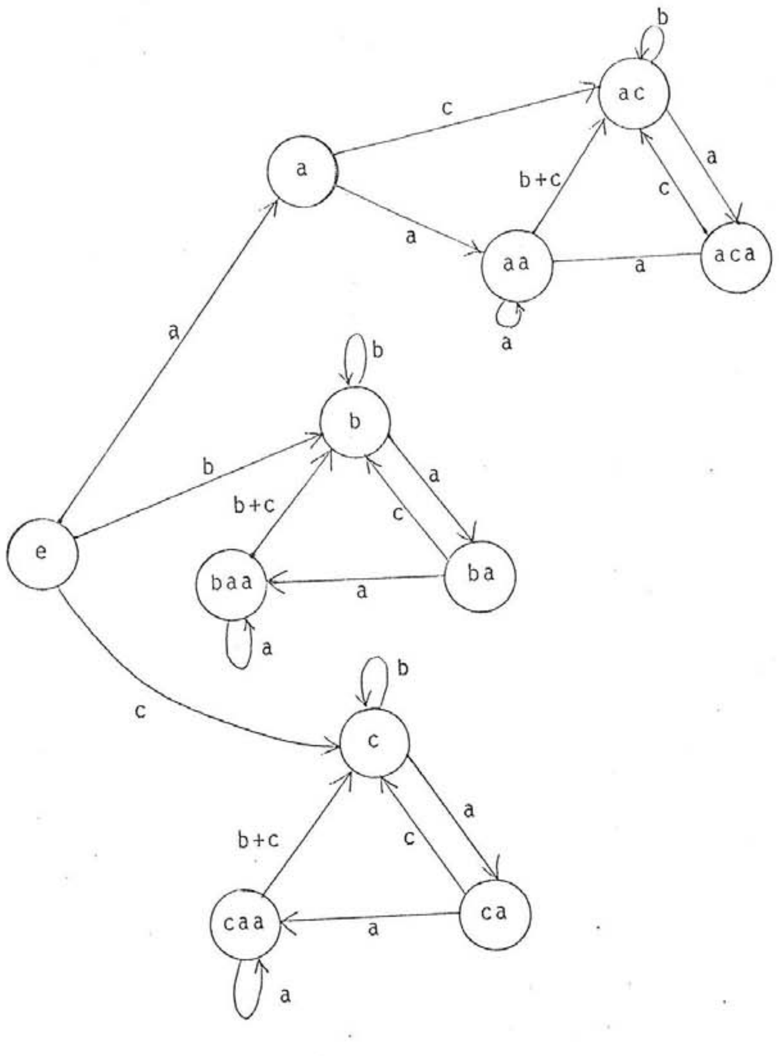


Fig. 11

Using the semigroup multiplication we can now determine each entry in $G_Q^* = \lceil Q \rceil$ as a union of c-classes of Q. Thus we get:

$$\lceil Q \rceil = \begin{bmatrix}
\begin{array}{l} e+a+aa \\ +ac+aca \end{array} & ac+aa & \begin{array}{l} a+aa \\ +aca \end{array} & aa \\
\begin{array}{l} e+a+aa \\ +ac+aca \\ +b+ba+baa \end{array} & \begin{array}{l} e+aa \\ +ac \\ +b+baa \end{array} & \begin{array}{l} a+aa \\ +aca \\ +ba+baa \end{array} & \begin{array}{l} aa \\ \\ +baa \end{array} \\
\begin{array}{l} e+a+aa \\ +ac+aca \\ +c+ca+caa \end{array} & \begin{array}{l} a+aa \\ +ac \\ +c+caa \end{array} & \begin{array}{l} e+a+aa \\ +aca \\ +ca+caa \end{array} & \begin{array}{l} a+aa \\ \\ +caa \end{array} \\
\begin{array}{l} e+a+aa \\ +ac+aca \\ +b+ba+baa \\ +c+ca+caa \end{array} & \begin{array}{l} e+a+aa \\ +ac \\ +b+baa \\ +c+caa \end{array} & \begin{array}{l} e+a+aa \\ +aca \\ +ba+baa \\ +ca+caa \end{array} & \begin{array}{l} e+a+aa \\ \\ +baa \\ +caa \end{array}
\end{bmatrix}$$

Finally by inspection of the above matrix we can verify that there are no indices i,j,p and k such that

$$Q_{ip} \subseteq Q_{ik} \quad \text{and} \quad Q_{pj} \subseteq Q_{kj} ,$$

and hence all factors are inseparable from Q.

The rank of G_Q is two whereas we already have an expression for Q which has star-height one. Thus for this example algorithm 2 fails to give a minimal star-height expression.

We conclude this section with a brief discussion of some of the "more interesting" examples studied by other authors.

Example 5 This example appears as example 6.5 in the paper by Cohen and Brzozowski [12].

Consider the language Q defined by the following machine.

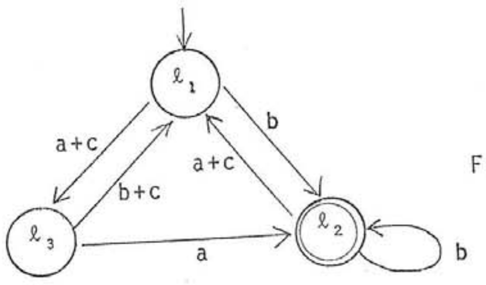


Fig. 12 Machine

Its anti-machine is given in the next diagram, following which we show a sequence of factor graphs of the language Q and some of its factors.

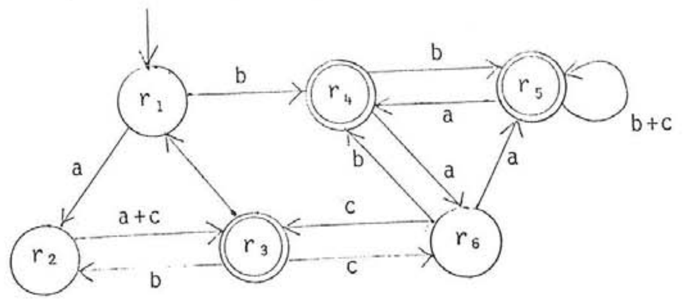


Fig. 13 Anti-machine

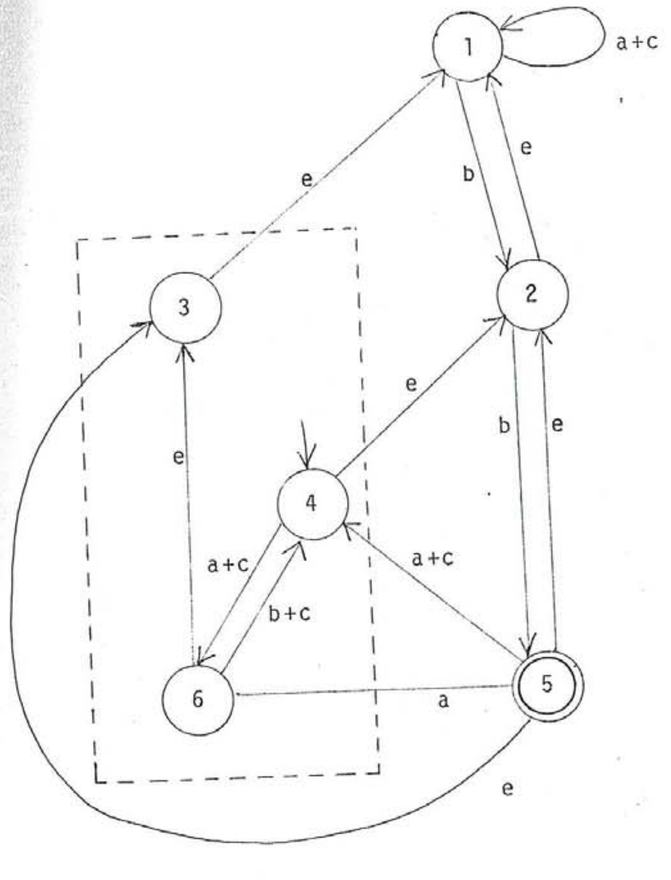


Fig. 14 Factor Graph of $Q = Q_{45}$

Matrix indicated by dotted lines = A, say.

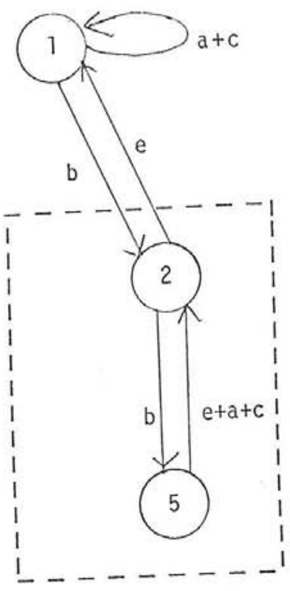


Fig. 15 Factor Graph of Q_{22}

Matrix indicated by dotted lines = B, say.

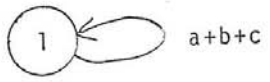


Fig. 16 Factor Graph of Q_{11}

If we apply algorithm 2 to the above sequence of factor graphs we would have to calculate Q_{11} (which is trivially $(a+b+c)^*$ - see fig. 16) and the closures B^* and A^*, where we have indicated the matrices A and B by dotted lines (figs. 14 and 15). Both of these matrices have rank 1 and so we deduce that Q has star-height 1.

This example was introduced by Cohen and Brzozowski to illustrate the difficulties inherent in a method they propose for finding the star-height of a regular language, their method being an enumerative one viz. calculate all subgraphs of a sequence of graphs of a specific type and determine whether each such graph is a recogniser for Q. In contrast using factor theory we are able to determine the star-height of this language directly, without any enumeration being involved.

Note also that neither the machine nor anti-machine have rank one - astute readers may have criticised our earlier examples on this point.

Example 6 (McNaughton [29],p314). The language

$$Q = \{x^*(x + z)\ x^*(y + z)\}^*$$

was proved by McNaughton to have star-height 2. His proof technique is to consider subfactors of a language and show that the complexity of their interconnections in any recogniser of Q must be above some value. In this case let G be any recogniser of Q, and consider those nodes N of G such that a word w in the subfactor x*zyz takes node N to some node N' in G. Let A be the set of all such nodes. Consider also the set B of all nodes M of G such that a word v in the subfactor zyzx* takes some node M' to node M in G. Then one easily proves that the sets A and B are disjoint but are strongly connected, and that they each define some strongly connected component of G of rank at least one. This then allows one to prove that the language has star-height at least two.

Examination of the machine or anti-machine for this language yields no insight which would lead to the above determination of the language's star-height. However if we study the factor graph (shown below - fig. 17)

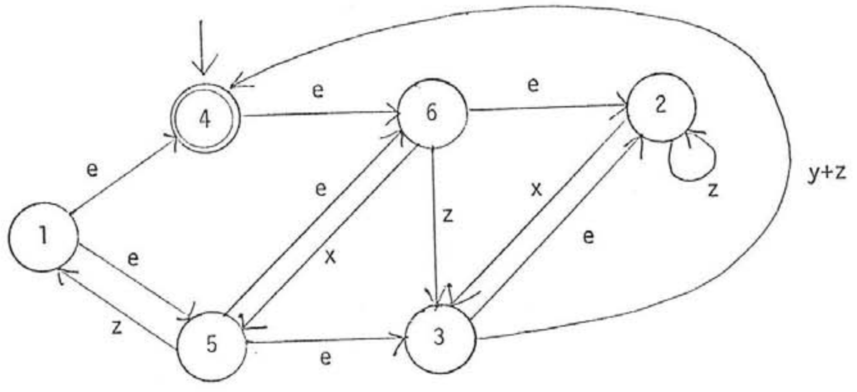


Fig. 17

we notice that we can eliminate nodes 1 and 4 to obtain the following factor graph in which all factors are inseparable.

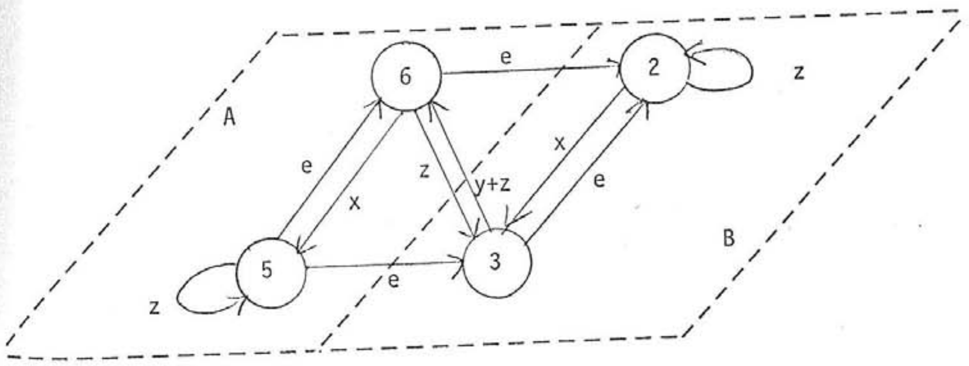


Fig. 18

A and B define two (almost) symmetrical halves of this graph (see fig. 18), but are distinguished by the asymmetry of the graph.

Example 7 Let Q = $(a + (a + b)c^*(c + d))^*$.

This example was also proved by McNaughton [29,p315] to have star-height two. The above expression is identical to what one would obtain from the factor graph (fig. 19).

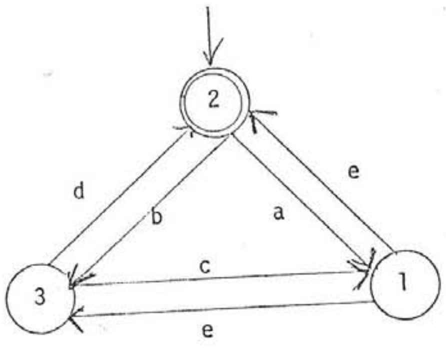


Fig. 19

All factors of Q are inseparable from Q. Note that the factor graph is pathwise homomorphic to the following graph, but which is not a factor graph for any language.

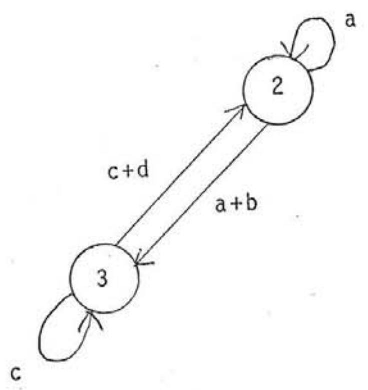
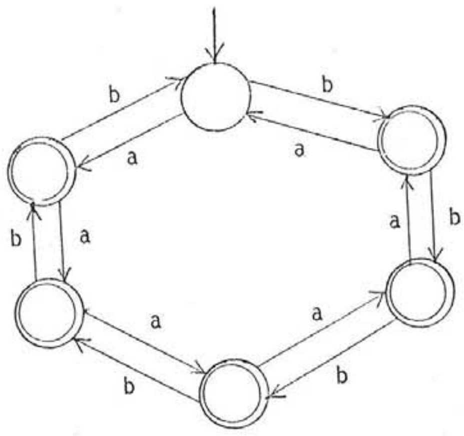


Fig. 20

CONCLUSIONS

In his book [13,p123] Conway advises readers to avoid calculating the factor matrix of any language Q. While disagreeing with this advice, for the simple reason that one cannot expect to make any advances in the theory of factors without doing such calculations, we should nevertheless mention two drawbacks to any practical use of factor theory.

The first is that calculations with factors inevitably tend to be rather long. With practice the method given in Chapter III to calculate the factor matrix $\overline{|Q|}$ is quite simple and straightforward, most of the effort in fact going into calculating the machine and anti-machine for Q. But, by hand, the work is tedious and rather prone to error, as well as involving quite a large amount of computation. Moreover, as soon as one begins calculating factor graphs of factors, the effort involved really does become quite daunting. (An obvious case for programming on a computer!) The second and much more significant reason, is that the factor graph may well have an extremely large number of nodes, even for languages having quite simple finite-state machines. For instance the language recognised by the finite-state machine below has a factor graph having 64 ($=2^6$) nodes.



Indeed, for any $n \geq 3$, the subset of $(a+b)^*$ consisting of all words such that the number of a's minus the number of b's is not congruent to 0 mod n has 2^n nodes in its factor graph (the above example is the case $n = 6$). Also if the language Q has a factor graph of a manageable size, the factor graph of $\sim Q$ will often be quite unmanageable. However these problems are not peculiar to factor theory, and would appear to be a fact of life when handling regular languages. If anything, they illustrate just how much we don't know about regular languages!

In retrospect, the algorithm given in chapter IV for calculating the closure G_Q^* of G_Q is based on very simple ideas. The essential ingredients of the algorithm are the following:

(a) there is a transitive relation "is a factor of" on the entries in G_Q^* .

(b) the relation "is a factor of" can be reduced to an equivalence relation "is inseparable from" on the entries in G_Q^* .

(c) each equivalence class modulo "inseparability" defines a graph G_H such that G_H^* is a submatrix of G_Q^* .

By way of comparison, consider any graph G and consider the following sequence of ideas.

(a)' there is a transitive relation "is connected to" on the nodes of the graph G.

(b)' the relation "is connected to" can be reduced to the equivalence relation "is strongly connected to" on nodes of G.

(c)' each equivalence class modulo "is strongly connected to" defines a subgraph of the graph G (in fact a section of G).

The process of deducing an algorithm to calculate G_Q^* is thus a very familiar one, and one inevitably seeks to extend it to other recognisers of Q. For example entries in M*, where M is the machine of Q, are derivatives of Q and derivatives of derivatives are derivatives. Thus "is a derivative of" is a transitive relation on the derivatives of Q. Unfortunately in this case no additional benefit is gained by considering a symmetric closure of "is a derivative of" since if S and T are derivatives of Q, S is a derivative of T and T is a derivative of S if and only if the two nodes to which they correspond (cf Theorem III 2.2) are in the same section of the machine of Q. Similarly analysis of the anti-machine or the semigroup machine yields no improvement on the usual elimination methods for finding their closure.

One possibility remains. We have observed that factors of a language Q are unions of c-classes of Q which are maximal in some subfactorization of Q. Instead of considering maximal unions of c-classes we could consider arbitrary unions of c-classes. Using similar techniques to those given in Chapter III, we could construct "c-class graphs" for Q. In a c-class graph, G, entries in G* are unions of c-classes of Q and each node can be labelled $(\ell_{i_1}+\ldots+\ell_{i_k}) \times (r_{j_1}+\ldots+r_{j_m})$ where the ℓ_i's are ℓ-classes and the r_j's are r-classes of Q. The only requirement is that

$$(\ell_{i_1}+\ldots+\ell_{i_k})\cdot(r_{j_1}+\ldots+r_{j_m})\subseteq Q$$

is a <u>sub</u>factorization of Q. Using some results of McNaughton and Papert [31], it is very easy to prove that unions of c-classes of unions of c-classes of Q are themselves unions of c-classes of Q. We thus have the beginnings of an analysis similar to the one given here for the factor graph. Moreover, in studying c-class graphs one can benefit much more from previous work on the star-height problem than we have been able to. For the "pure-group events" studied by McNaughton [28] a c-class graph is clearly what he would call a "μ-graph". Cohen and Brzozowski [12] also study "subset automata" which are particular cases of c-class graphs.

Very severe practical problems arise however, as there is usually not just one, but many c-class graphs for a particular language Q (one of which is the factor graph). Problems which we mentioned above in studying factor graphs are thus accentuated tremendously. Needless to say, we have no empirical results to date as to how successful this could be, and there is still a lot to be done before the star-height problem is solved.

An intermediate problem which is probably worth tackling before embarking on an investigation of c-class graphs is the following. Let F and H be two regular languages, and suppose F is a union of c-classes of H and H is a union of c-classes of F. Is the star-height of F equal to the star-height of H? An answer of no to this question would make us very sceptical of pursuing this line of investigation, but we would guess that the answer is more likely to be yes.

It is important to note that algorithm 2 cannot have an analogue in linear algebra. Considering yet again example 1; from the factor graph of the language Q (fig. 1,p.123) we get the following system of equations defining Q:

$$\begin{aligned} Q &= (e+a)P + e \\ P &= bQ + eT \\ T &= aT + bP \quad , \end{aligned}$$

where $P = Q_{21}$ and $T = Q_{31}$. Substituting $e = 1$, $a = -\frac{1}{2}$, $b = 1$ in these equations and solving (in linear algebra) for Q, we get

$$Q = -2 \quad .$$

However if we replace m^* by $(1-m)^{-1}$ and substitute the same values for e,a and b in

$$Q = (b+ab)^*(e+a)(a+b)^*b(b+ba)^*b + (b+ab)^*$$

(IV § 4) we get

$$Q = 6 .$$

If the method did have an analogue in linear algebra, the two values would agree, which they evidently do not.

The main conclusion to be drawn from the work presented here is that one should not be content with the simple elimination methods of Chapter II, and more effort should be put into developing new closure algorithms. The approach we would suggest for doing this would be the same as that used here. That is, investigate one particular class of graphs, develop closure algorithms using properties particular to this class and finally seek to extend the algorithms to have more general applicability. Moreover, let us not see any hypothetical solutions to the star-height problem which simply involve "enumeration" until all other possible hypotheses have definitely been exhausted.

We have had little to say in this thesis about other applications of Conway's factor theory, but we would anticipate that it will become much more important as a theoretical tool in the study of regular languages. Conway introduces it as a method of studying approximations to regular languages, and goes on to use it extensively in his study of the biregulators (Note: biregulators are usually called "generalised sequentia machines".) Some connections between factors and the semigrou of a language have been pointed out, and we feel that the direction of future research on factors would be to investigate such connections more fully. Note that it is always possible to discover the semigroup multiplication directly from the matrix $C^Q_{max} + L^Q_{max}$, and so in studying the factor matrix one loses no information about the semigroup. Indeed much more information is contained in the factor matrix. The semigroup is often too coarse a description of a language, since it is invariant under relabelling of the start and terminal nodes of the machine. However under such relabelling the factor matrix changes quite substantially. For these reasons we would expect factor theory to yield more insight into those problems which have traditionally been tackled by studying the semigroup of the language [30].

But these are just a few suggestions for further work on regular languages. The study of regular languages is a particularly attractive area for further research since it offers quite a large number of unsolved or inadequately solved problems. (See [34] for further discussion.) Moreover problems in regular algebra are usually expected to be solvable - but they are clearly very difficult and very challenging.

REFERENCES

1. Aho, A.V. and Ullman, J.D.
   "The Theory of Parsing, Translation and Compiling", vol.I.
   Prentice Hall, Englewood Cliffs, N.J. 1972.

2. Backhouse, R.C. and Carré, B.A.
   "Regular algebra applied to path-finding problems"
   J. Inst. Maths. Applics. 15 (1975) pp.161-186.

3. Brzozowski, J.A.
   "Derivatives of regular expressions"
   J.A.C.M. 11, 4 (Oct. 1964) pp.481-494.

4. Brzozowski, J.A.
   "Roots of star events"
   J.A.C.M. 14, 3 (July 1967) pp.466-477.

5. Bunch, J.R.
   "Complexity of sparse elimination"
   In
   "Complexity of sequential and parallel numerical algorithms"
   Traub, J.F. (Ed.), Academic Press, New York and London (1973).

6. Carré, B.A.
   "An algebra for network routing problems"
   J. Inst. Maths. Applics. 7 (1971), pp.273-294.

7. Carré, B.A.

"A matrix factorization method for finding optimal paths through networks"

I.E.E. Conf. Publ. No.51 (Computer Aided Design) 1969, pp.388-397.

8. Carré, B.A.

"An elimination method for minimal-cost network flow problems"

In

"Large sparse sets of linear equns."

(Proc. IMA Conf., Oxford, 1970) Ed. J.K. Reid, Academic Press, London.

9. Cohen, R.S.

"Rank non-increasing transformations on transition graphs"

Inf. and Control 20 (1972), pp.93-113.

10. Cohen, R.S.

"Star-height of certain families of regular events"

J. Comp. Syst. Scs. 4, 3 (1970), pp.281-297.

11. Cohen, R.S.

"Techniques for establishing star-height of regular sets"

Math. Systs. Th. 5, 2 (1971) pp.97-114.

12. Cohen, R.S. and Brzozowski, J.A.

"General properties of star-height of regular events"

J. Comp. Syst. Scs. 4, 3 (1970), pp.260-280.

13. Conway, J.H.

"Regular algebra and finite machines"

Chapman and Hall, London 1971.

14. Dantzig, G.B.

"All shortest routes in a graph"

Theory of Graphs (International Symposium, Rome 1966), Gordon and Breach, New York, pp.91-92.

15. Dejean, F. and Schützenberger, M.P.

"On a question of Eggan"

Info. and Control 9, (1966) pp.23-25.

16. Dijkstra, E.W.

"A note on two problems in connexion with graphs"

Numerische Mathematik, 1 (1959), pp.269-271.

17. Dreyfus, S.E.

"An appraisal of some shortest path algorithms"

Operat. Res. 17, 3 (May 1969), pp.395-412.

18. Eggan, L.C.

"Transition graphs and the star-height of regular events"

Michigan Math. J. 10 (1964) pp.385-397.

19. Floyd, R.W.

"Algorithm 97, shortest path"

Comm. Assoc. Comp. Mach. 5, 6 (June 1962).

20. Fontan, G.

"Sur les performances d'algorithmes de recherche de chemins minimaux dans les graphes clairsemés"

Rep. Lab. d'Auto. et d'Analyse des Systèmes, Toulouse (1974).

21. Fox, L.
"An introduction to numerical linear algebra"
Clarendon Press, Oxford (1964).

22. Ginzburg, A.
"Algebraic theory of automata"
Academic Press, New York 1968.

23. Grassin, J. and Minoux, M.
"Variations sur un algorithme de Dantzig avec application à la recherche des plus courts chemins dans les grands reseaux"
Rev. Fr. d'Auto. Info. & Rech. Oper. 1 (March 1973).

24. Hartmanis, J. and Stearns, R.E.
"Algebraic structure theory of sequential machines"
Prentice Hall, Englewood Cliffs, N.J. (1966).

25. Hoffman, A.J. and Winograd, S.
"Finding all shortest distances in a network"
IBM J. Res. & Devel. 16, 4 (July 1972), pp.412-414.

26. Householder, A.S.
"Principles of numerical analysis"
McGraw Hill, New York 1953.

27. Johnson, D.B.
"Algorithms for shortest paths"
Technical Report 73-169, Dept. of Computer Science, Cornell University (May 1973).

28. McNaughton, R.
"The loop complexity of pure-group events"
Info. and Control, 11 (1967), pp.167-176.

29. McNaughton, R.
"The loop complexity of regular events"
Infor. Sciences 1 (1969), pp.305-328.

30. McNaughton, R. and Papert, S.
"Counter-free automata"
Res. Monograph no.65, M.I.T. Press, Cambridge, Mass. and London.

31. McNaughton, R. and Papert, S.
"The syntactic monoid of a regular event"
In Arbib, M.A. (Ed.)
"Algebraic theory of machines, languages and semigroups"
Academic Press, New York (1968).

32. Munro, I.
"Efficient determination of the transitive closure of a directed graph"
Info. Proc. Letters (1971) pp.56-58, North Holland Publ. Co.

33. Murchland, J.D.
London School of Economics Report LSE-TNT-26 (1967).

34. Rabin, M.O.
"Mathematical theory of automata"
In J.J.T. Schwartz (Ed.)
"Mathematical aspects of computer science"
Amer. Math. Soc. Proc. in Applied Maths. 19, (1967).

35. Rabin, M.O. and Scott, D.
"Finite automata and their decision problems"
IBM J. Res. Dev. 3, 114-125 (1959).

36. Rodionov, V.V.

USSR Comp. Math. and Math. Phys. 8, (1968), pp.336-343.

37. Rose, D.J. and Bunch, J.R.

"The role of partitioning in the numerical solution of sparse systems"

In

"Sparse matrices and their applications"

Rose, D.J., Willoughby, R.A. (Eds.) Plenum Press, New York and London (1972), pp.177-187.

38. Salomaa, A.

"Two complete axiom systems for the algebra of regular events"

J.A.C.M. 13, 1 (Jan. 1966), pp.158-169.

39. Salomaa, A.

"Theory of automata"

Pergamon Press, Oxford (1969).

40. Scott, D.

"The lattice of flow diagrams"

In

"Symposium on semantics of algorithmic languages"

Ed. E. Engeler: Springer-Verlag, Berlin (1971).

41. Strassen, V.

"Gaussian elimination is not optimal"

Numer. Math. 13 (1969), pp.354-356.

42. Tewarson, R.P.

"Sparse matrices"

Academic Press, New York and London (1973).

43. Warshall, S.

"A theorem on boolean matrices"

J.A.C.M., 9 (1962), pp.11-12.

44. Yen, J.Y.

"Finding the lengths of all shortest paths in N-node non-negative - distance complete networks using $\frac{1}{2}N^3$ additions and N^3 comparisons"

J.A.C.M., 19, 3 (July 1972), pp.423-424.

45. Yen, J.Y.

"An algorithm for finding shortest routes from all source nodes to a given destination in general networks"

Quart. Appl. Maths. 27 (1970), pp.526-530.