AN EXAMINATION OF COUNTER-FREE AUTOMATA

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Sonal Pratik Patel

Fall 2010

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Sonal Pratik Patel:

An Examination of Counter-Free Automata

[illegible]

Carl Eckberg, Chair
Department of Computer Science

[illegible]

William Root
Department of Computer Science

J. Carmelo Interlando

J. Carmelo Interlando
Department of Mathematics and Statistics

June 8, 2010

Approval Date

ABSTRACT OF THE THESIS

An Examination of Counter-Free Automata
by
Sonal Pratik Patel
Master of Science in Computer Science
San Diego State University, 2010

This thesis is an extensive examination of a graduate level monograph, Counter-Free Automata by McNaughton and Papert, to develop a deep understanding of the subject. This book is about many subclasses of type 3 languages and it proves all of these subclasses are equivalent. We have explored chapters 1 to 8 and 10. The real goal of this thesis is to solve as many problems at the end of each chapter in the book as possible.

The thesis rehashes the discussion in the text of the nine chapters mentioned above. It is to be noted the text has no solution in the back of the book nor do any solutions appears to be available online. Since half or more of the problems have been solved in this undertaking, in effect a substantial beginning of a solution manual has been produced.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I am heartily thankful to my advisor, Dr. Carl Eckberg, whose encouragement, supervision and support from the preliminary to the concluding level enabled me to develop an understanding of the Counter-Free Automata. Without his enormous help and time, this would not have been possible.

CHAPTER 1

INTRODUCTION AND SUMMARY

This chapter reviews some of the basic definitions about finite state automata. We can find some basic concepts like event, alphabet, word, star operator and Boolean operators. It also explains about state graphs and regular expressions.

The main concept of this chapter is to explain and define "*noncounting events*" which is a very important subclass of the class of regular events. A regular event is always recognized by a finite state machine.

The only significant presented presentation of non-counting events occurs in Counter-Free Automata by McNaughton and Pappert. This monograph explores a large number of equivalent characterizations of this subclass of the type 3 language.

The thesis will express these notions in the author's own words, and as many problems as possible from this text will be solved. Each text chapter will be explained in a corresponding thesis chapter, but will include as many solutions as possible to problems at the end of the text chapter. So most solved exercises correspond to text exercises.

In many places, the wording of concepts and explanations in thesis closely follows that in Counter-Free Automata by McNaughton and Pappert and therefore the attribution to this text will not be explicitly made in various places.

There are actually more significant characterizations of Counter-Free Automata than of finite automata in general.

The state graph for Figure 1.1 represents the event $(a^*ba^*b)^*a^*$, which is the set of words having an even number of b's. The state graph for Figure 1.2 has the event $(ab \cup ba)^*a$, which is the set of all words of odd length e ending in a and whose $(2i - 1)$th and $(2i)$th letter differ from each other, for each $i \leq (e - 1)/2$. The state graph Figure 1.3 represents the event $(a \cup b)^*aaa$, which is the set of all words ending in aaa.

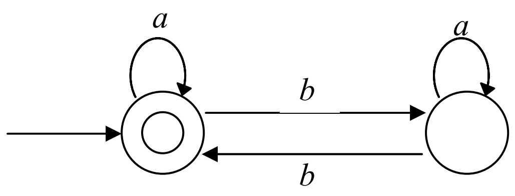

Figure 1.1. Automaton for $(a^*ba^*b)^*a^*$.

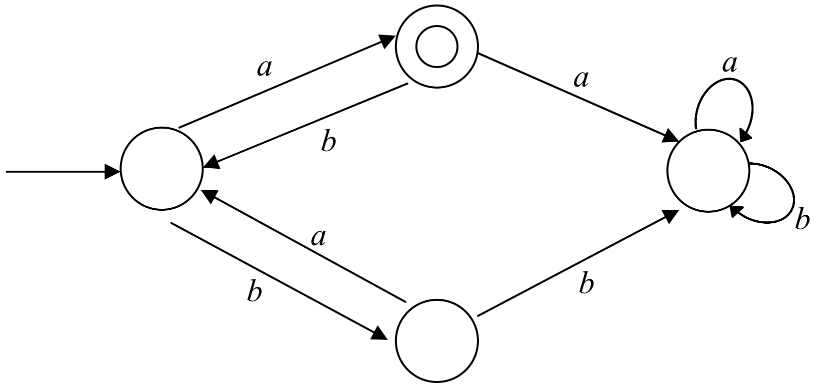

Figure 1.2. Automaton for $(ab \cup ba)^*a$.

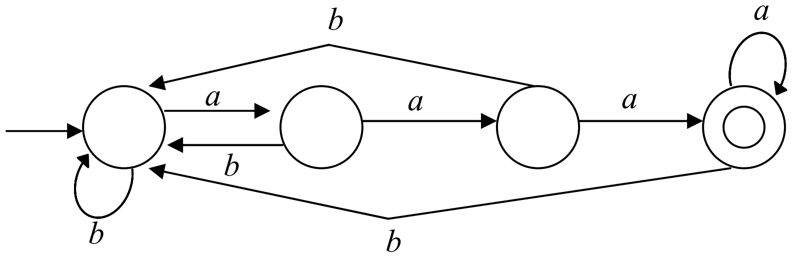

Figure 1.3. Automaton for $(a \cup b)^*aaa$.

Figure 1.4 represents $(a^*ba^*b)^*a^*aaa$, the set of words having an even number of b's and also ending with three consecutive a's. Finally, we note that $\sim((a^*ba^*b)^*a^*) = (a^*ba^*b)^*a^*ba^*$ is represented by Figure 1.5; in English, it is the set of all words with an odd number of b's.

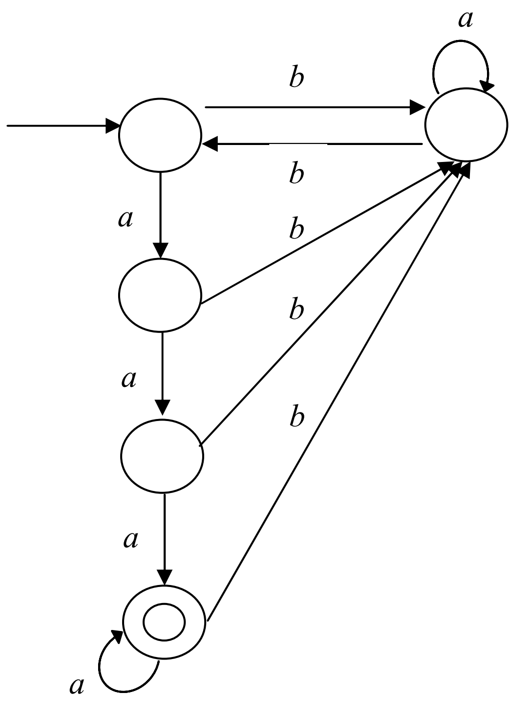

Figure 1.4. Automaton for $(a^*ba^*b)^*a^*aaa$.

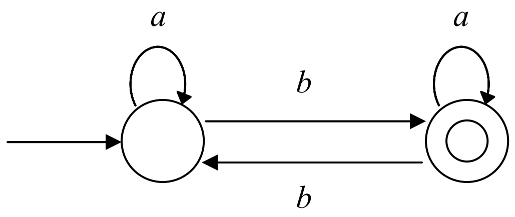

Figure 1.5. Automaton for $(a^*ba^*b)^*a^*ba^*$.

In automata theory the task of computation is to examine a word over an alphabet. The word is written on a tape from left to right, each letter occupying a square of the tape.

Finite state devices can determine three kinds of operations. First, it can check whether a certain pattern occur or not at any place in the word. Second, it can determine position or pattern of certain pattern in word. Third, it can count how many times something occurs. Certainly, counting for a finite state device must be modulo an integer.

There are two kinds of counting operations finite state devices can perform. One is counting modulo an integer greater than unity, which is just discussed above. The other is counting to a threshold, in which it can determine at least how many occurrences of certain patterns are in the word. From now onwards, the word "counting" will mean counting modulo an integer ≥ 2. In this section the class of those events is discussed in which, there is no need for counting operations. We will define the noncounting event as follows.

A *noncounting event* is a regular event L such that, for some n and for all words U, V and W over its alphabet, $UV^{n+x}W \in$ L if and only if $UV^{n}W \in$ L, for all positive integers x. The symbol *NC* is used for the class of all noncounting events.

We now present solution to many of the problems at the end of Chapter 1 of the text. We generally denote languages over alphabet ∑ by L, L_1, L_2, etc.

The following are exercise solutions from the book.

1. Prove that the class *NC* is unaltered if, in its definition, the clause "$UV^{n+x}W \in$ L if and only if $UV^{n}W \in$ L, for all positive integer x" is replaced by the clause "$UV^{n+1}W \in$ L if and only if $UV^{n}W \in$ L."
   Ans: By the definition of *NC*, for some n and for all words U, V, and W over its alphabet,
   $UV^{n+x}W \in \text{L} \Leftrightarrow UV^{n}W \in \text{L}$ for all $x > 0$
   Case 1: If $x = 1$ then we have,
   $UV^{n+1}W \in \text{L} \Leftrightarrow UV^{n}W \in \text{L}$
   which shows that the class of *NC* is unaltered.
   Case 2: if $x > 1$ then,
   $UV^{1}W \in \text{L} \Leftrightarrow UV^{n+1}W \in \text{L}$, whence
   $(UV)V^{n}W \in \text{L} \Leftrightarrow (UV)V^{n+1}W \in \text{L} \Leftrightarrow UV^{n+2}W \in \text{L}$, etc.
   which also shows that the class of *NC* is unaltered.

2. Prove that if L $\in NC$ then $L^r \in NC$, where $L^r = \{ W \mid W^r \in L \}$, where W^r is W written backwards.
Ans: $L^r = \{W \mid W^r \in L\}$
By the definition of *NC*, for some *n* and for all words *U*, *V*, and *W* over its alphabet,
if $UV^{n+x}W \in L \Leftrightarrow UV^nW \in L$ for all x > 0
then,
$W^r (V^r) V^{n+x}U^r \in L^r \Leftrightarrow W^r(V^r)^nU^r \in L^r$, which says the same thing.

3. Which of the events represented by the graphs Figures 1.1 through 1.5 above are noncounting ? Justify your answers.
Ans:
Figure 1.1 accepts set of words containing an even number of *b*'s. Now let, $L = \{b^{2n} \mid n \geq 0\}$. Assume $\exists\, n_0\, \forall\, U, \forall\, V, \forall\, W\ [\forall\, x > 0\ \ UV^{n+x}W \in L \Leftrightarrow UV^nW \in L]$. Consider $b^{2n0} = \lambda\, b^{n0}\, b^{n0} \in L$, where $U = \lambda$, $V = b^{n0}$, and $W = b^{n0}$. If $x = 1$, then $\lambda\, b^{n0+1}\, b^{n0} \notin L$, since n_0 is arbitrary L is not *NC*.
Figure 1.3 accepts the set of words ending with *aaa*'s. Let $L = \{a^naaa \mid n \geq 0\}$. Assume $\exists\, n_0\, \forall\, U, \forall\, V, \forall\, W\ [\forall\, x > 0\ \ UV^{n+x}W \in L \Leftrightarrow UV^nW \in L]$. Let $a^{n0}aaa = \lambda\, a^{n0}\, aaa \in L$, where $U = \lambda$, $V = a^{n0}$, and $W = aaa$. If $x = 1$, then $\lambda\, a^{n0+1}\, aaa \in L$, since n_0 is arbitrary L is *NC*.
Figure 1.4 accepts set of words having even number of *b*'s and also ending with *aaa*'s. Now let, $L = \{b^{2n}aaa \mid n \geq 0\}$. Assume $\exists\, n_0\, \forall\, U, \forall\, V, \forall\, W\ [\forall\, x > 0\ \ UV^{n+x}W \in L \Leftrightarrow UV^nW \in L]$. Consider $b^{2n0}\, aaa = b^{n0}\, b^{n0}\, aaa \in L$, where $U = b^{n0}$, and $V = b^{n0}$, $W = aaa$. If $x = 1$, then $b^{n0}\, b^{n0+1}aaa \notin L$, since n_0 is arbitrary L is not *NC*.
Figure 1.5 accepts set of words containing odd number of *b*'s. Now let, $L = \{b^{2n+1} \mid n \geq 0\}$. Assume $\exists\, n_0\, \forall\, U, \forall\, V, \forall\, W\ [\forall\, x > 0\ \ UV^{n+x}W \in L \Leftrightarrow UV^nW \in L]$. Consider $b^{2n0+1} = b^{n0}\, b^{n0}\, b \in L$, where $U = b^{n0}$, and $V = b^{n0}$, $W = b$. If $x = 1$, then $b^{n0}\, b^{n0+1}\, b \notin L$, since n_0 is arbitrary L is not *NC*.

4. Prove that the class *NC* is closed under concatenation, intersection, union, and complementation.
Ans: See Theorem 2.1 from Chapter 2.

5. Show that the event of Figure 1.1 has as an infinite subset an event that is noncounting.
Ans: Fig. 1.1 represents the event $L = (a^*ba^*b)^*a^*$, which is the set of words having an even number of *b*'s and it is not noncounting. Now let $L' = a^*$ where, L' represents the set of words having zero *b*'s. Clearly, L' is a subset of L. Also L' is a noncounting event. For example, let $L = \{a^n \mid n \geq 0\}$. Assume $\exists\, n_0\, \forall\, U, \forall\, V, \forall\, W\ [\forall\, x > 0\ \ UV^{n+x}W \in L \Leftrightarrow UV^nW \in L]$. Consider $a^{n0} = \lambda\, a^{n0}\, \lambda \in L$, where $U = \lambda$, and $V = b^{n0}$, $W = \lambda$. If $x = 1$, then $\lambda\, a^{n0+1}\, \lambda \in L$, since n_0 is arbitrary L is *NC*.

6. Follow the instruction of Exercise 5 for the events of Figures 1.2 through 1.5. (The solution will be trivial for those events that are themselves noncounting.)
Ans:
Figure 1.2 represents the event $L = (ab \cup ba)^*a$ which is the set of all words of odd length *e* ending in *a* and whose (2*i* – 1)th and (2*i*)th letter differ from each other, for each $i \leq (e - 1)/2$. Now let $L' = (ab)^*a$. Where L' is an infinite subset of L and clearly, L' is noncounting. For example, let $L = \{(ab)^n\, a \mid n \geq 0\}$. Assume $\exists\, n_0\, \forall\, U, \forall\, V, \forall\, W\ [\forall\, x >$

0 $UV^{n+x}W \in L \Leftrightarrow UV^{n}W \in L$]. Consider $(ab)^{n0}\, a = \lambda\, (ab)^{n0}\, a \in L$, where $U = \lambda$, and $V = (ab)^{n0}$, $W = a$. If $x = 1$, then $\lambda\, (ab)^{n0+1}a \in L$, since n_0 is arbitrary L is *NC*.
The state graph of Figure 1.3 represents the event $(a \cup b)^{*}aaa$, which is the set of all words ending in *aaa*. Let L' = $(a)^{*}aaa$ and L' is an infinite subset of L. L' is noncounting. Let L = $\{a^{n}aaa \mid n \geq 0\}$. Assume $\exists\, n_0\, \forall\, U, \forall\, V, \forall\, W$ [$\forall\, x > 0\;\; UV^{n+x}W \in L \Leftrightarrow UV^{n}W \in L$]. Let $a^{n0}aaa = \lambda\, a^{n0}\, aaa \in L$, where $U = \lambda$, $V = a^{n0}$, and $W = aaa$. If $x = 1$, then $\lambda\, a^{n0+1}\, aaa \in L$, since n_0 is arbitrary L is *NC*.

Figure 1.4 represents $(a^{*}ba^{*}b)^{*}a^{*}aaa$, the set of words having an even number of *b*'s and also ending with three consecutive *a*'s. Let L' = $a^{*}aaa$. Clearly, L' is an infinite subset of L and it is noncounting. Consider same example of Figure 1.3.

Figure 1.5 represents $(a^{*}ba^{*}b)^{*}a^{*}ba^{*}$, the set of all words with an odd number of *b*'s. Now let L' = ba^{*}. Where L' is an infinite subset of L and also noncounting. For example, let L = $\{(ba^{n} \mid n \geq 0\}$. Assume $\exists\, n_0\, \forall\, U, \forall\, V, \forall\, W$ [$\forall\, x > 0\;\; UV^{n+x}W \in L \Leftrightarrow UV^{n}W \in L$]. Consider $b\, a^{n0} = b\, a^{n0}\, \lambda \in L$, where $U = b$, and $V = a^{n0}$, $W = \lambda$. If $x = 1$, then $b\, a^{n0+1}\, \lambda \in L$, since n_0 is arbitrary L is *NC*.

7. Prove that if L is a noncounting event over an alphabet with a single letter, then either L is finite or L is cofinite (that is, has a finite complement).
Ans: If L is finite, let $n_0 = max\, \{\, |w| : w \in L\}$. Then $n_0 + 1$ is clearly a correct magic number. If L is cofinite and $\sum^{*}$ - L has longest word length n_0, then $n_0 + 1$ is clearly a magic number to show L is noncounting.
Let L be noncounting, $L \subseteq a^{*}$, and suppose L is not finite, and n_0 is a magic number for L. Since L is not finite $\exists\, \lambda\, a^{n0}\, a^{m} \in L$ for some $m \geq 0$. Where $a^{m+n0+x} \in L\;\; \forall\, x \geq 1$. So the complement of L is finite. i.e L is cofinite.

8. Give an example of an infinite regular event having the property that no infinite regular subset is noncounting.
Ans: Without loss of generality Let $a \in \sum$ and Let L' = $\{a^{2n} : n \geq 1\}$
Let $L' \subseteq L$, L' infinite. Assume $\exists$ *a* magic number n_0 making L' noncounting. Since L' is infinite let $n_1 \geq n_0$ where $a^{2n1} \in L'$, So $(a)^{n1}a^{n1} \in L$. Thus if L' is noncounting, we have ($u = \lambda$, $v = a$, w = a^{n1}) $a_1^{n+1}a^{n1} \in L$ also, but this is impossible.

9. Prove that, for any infinite noncounting event L_1, there exists a homomorphic image L_2 of L_1 such that L_2 is not noncounting.
Ans: Let $\sum = \{a_1, a_2, \ldots, a_n\}$ and let $h(a_i) = a^{2}$, $\forall\, i \geq 1$. Let L be infinite. Therefore h(L) is an infinite subset of $(a^{2})^*$. We are done by the proof of exercise 1.8.

10. Prove that, if L_2 is the image of L_1 under a one-to-one homomorphism and if L_2 is noncounting then L_1 is noncounting. Show that the hypothesis that the homomorphism be one-to-one is necessary.
Ans: Let h : $L_1 \rightarrow L_2$ be 1-1 and onto homomorphism.
Let *n* be magic number for L_2 noncounting.
We try *n* for L_1
Let $uv^{n}w \in L_1$, and *x* be the positive integer number.
To show $uv^{n+x}w \in L_1$
We have $h\, (uv^{n+x}w\,) \in L_2$
So $h(u)\, h(v)^{n+x}\, h(w) \in L_2$

Hence $h(u)\, h(v)^n\, h(w) \in L_2$
So $h(uv^n w) \in L_2$
So $uv^n w \in L_2$
Let $h : \sum \rightarrow \in$ counter example for one to one map.

11. Prove that for any alphabet $\sum_1$ there is a homomorphism ψ of the set of words over $\sum_1$ into the set of words over $\sum_2 = \{a, b\}$ such that for all events L_1 over $\sum_1$, if L_1 is noncounting then $\psi(L_1)$ is also noncounting. Make sure your proof is rigorous.
Ans: Let $L \in NC$ and let L have magic number n_0 with respect to $\sum_1$, now that L is not finite.
We can assume $|\sum_1| \geq 3$. If $\sum_1 = \{a_1, \ldots, a_{n0}\}$ we can define the $h(a_1) = bab$, $h(a_2) = ba^2b$, $h(a_3) = ba^3b$, etc. This is a monomorphic map from $\sum_1^*$ into $\{a, b\}^*$. For simplicity let $\sum_1 = \{a_1, a_2, a_3\}$.
The magic number for $h(L)$ can probably be n_0, but max $h(a_i) = 5$. So, let $n_1 = 5$. We assume $n_1 \geq 2$ and $|\sum_1|$ as well now let $Z' = u'v'^{n1}w' \in h(L)$ without loss of generality assume $v' \neq \lambda$.
Case 1: $u' = h(u)$, $v' = h(v_1)$, $w' = h(w)$ then $z' = h(u\, v_1^{n1}\, w)$ and thus $u\, v_1^{n1}\, w \in L$ because on $\sum_1^*$ h is one to one, and Z is for of something in L.
But then $\forall\, x \geq 0 \;\; u\, v_1^{n1+x}\, w \in L$ since $L \in NC$, whence $u'\, v'^{n1+x}\, w' \in h(L)$ end case.
Case 2: $v' \in a^+ \cup b^+$, then $u'\, v'^{n1}\, w'$ cannot possibly be in $h(L)$. since $n_1 \geq 2\,|\sum_1|$ and $u'v'^{n1}w' \in h(L)$ means the max n_0 of course a's is $|\sum_1|$ and max n_0 of b's is 2.
Case 3: $Z' = u'\, v'^{n1}\, w' = h(Z)$, $Z \in L$ so $Z' \in \{bab, baab, baaab\}^+$. Since we are not in Case 1, v' is not h of something, and v' has both a's and b's in it, since not Case 2.
Now $v' \notin \{ab, ba, aab, baa, aaab, baaa\}$ for some reasoning as in Case 2 (b's). Thus v' states with a suffix of one of $S = \{bab, baab, baaab\}$ and ends with a prefix of some element of S.
E.g. $v' = abbaabbabbaa$. We examine this example clearly $u'\, v'^{n1}\, w' = u''\, u'''\, v'^{n1}\, w''\, w'''$ where u'' and w''' are in $h(\sum_1^*)$ and u'' and w'' complete the leading ab and trailing baa of u' to complete a single element of S. Note that v' can? begin or end with an a of S but not both.

CHAPTER 2

LOCAL TESTING AND ORDER

There are two interesting subclasses of regular events; “finite events” and “definite events”. A finite set of words is known as a *finite* event. Let $\sum^*$ be the set of all words over the alphabet $\sum$. $\sum^*$ is a “free monoid” under concatenation generated by $\sum$. Let α and β be such finite events, if $\delta = F\alpha \cup \beta$ then δ is a *definite* event. In simple words, a definite event is an event having property that, for some positive integer k, membership of a given word in the event can be tested by looking at only the last k letters if the word is longer than k. This concept generalizes to a class of locally testable events; in which tests of length k can be done on all parts of the word, not only at the right end.

Let $L_k(W)$, $R_k(W)$, $I_k(W)$ be, respectively, the left end segment of W of length k, the right end segment of W of length k, and the set of interior segments of W of length k. These are defined only when W has length of k or more. If W has length k, then $L_k(W) = R_k(W) = W$. If W has length k or $k + 1$, then $I_k(W)$ is the empty set. An event L is *k- testable*, if for all W and W' of length k or more, if $L_k(W) = L_k(W')$, $R_k(W) = R_k(W')$, and $I_k(W) = I_k(W')$, then either both W and W' are in L or neither are. An event is *locally testable* if it is *k-testable* for some positive integer k. we denote these languages by LT_k and LT respectively.

Some examples of locally testable events over the alphabet {0, 1} are as follows:

1. The set of words in which a 011 never occurs.
2. The set of all words beginning with 000 and ending with 111.
3. The set of all words length 5 or more that have no occurrence of 00.

4. The set of all words w such that either w is null word, or else w is of length 8 or more and contains both 000 and 111 as interior segments.
5. To learn how the machines for Locally Testable events work, see the book (Counter-free automata, by McNaughton, Papert).

There is a smaller class of locally testable events which is known as "*locally testable events in the strict sense*." An event L is *k –testable in the strict sense* if there are sets α, β, and γ such that, for all w of length k or more, $W \in$ L if and only if $L_k(W) \in \alpha$, $I_k(W) \subseteq \beta$, $R_k(W) \in \gamma$. Then L is *locally testable in the strict sense* if L is *k – testable in the strict sense*, for some $k \geq 1$. We denote these languages by $LTSS_k$ and *LTSS* respectively.

Some examples of locally testable events in the strict sense are as follows:

1. The set of words in which a 011 never occurs.
2. The set of all words in which 011 never occurs except possibly at the left end.
3. The set of all words of length 3 or more that have no occurrence of 000.

In simple words, an event L is *k*- testable in the strict sense if there are three independent test that one can make on segments of length *k* such that, if the left length-*k* segment of a word *W* passes the first test, every interior length-*k* segment passes the second test, and the right length-*k* segment of *W* passes the third test, then *W* is in the event, and otherwise not. Note that one can process *W* word left to right and do a "quick reject" for L locally testable in the strict sense based on non-membership in β.

Considering closure properties, for each *k*, the class of *k*-testable events is closed under the Boolean operations; and the class of locally testable events is closed under the Boolean operations. The class of *k*-testable events in the strict sense and the class of locally testable events in the strict sense are not closed under union or complementation but closed under intersection. The class of *k*-testable events (locally testable events) is the closure of the class of k-testable events in the strict sense (locally testable events in the strict sense) under the Boolean operations.

There is another class of noncounting events, known as *LTO*, which is the smallest class of events that contains all the locally testable events and is closed under the Boolean operations and concatenation. Here, concatenation is the means of getting in the notion of order, which means the words an *LTO* event contains should be concatenation of words in some particular order. For example, let L be the set of all words over {0, 1} that contain at least one occurrence of the segment 00 with at least one occurrence of the segment 11 somewhere to its right. In this event, the more appearance of 00 and 11 is not sufficient; they must appear in a certain order. Now L ∈ *LTO*. For L = $\alpha\beta$, where α and β are respectively, the set of words having a segment 00 somewhere and the set of words having a 11 somewhere. Obviously, α and β are locally testable. On the other hand, L itself is not locally testable. For suppose it were k-testable, and consider $W = (10)^k(01)^k(10)^k$ and $W' = 1(01)^k(10)^k(01)^k0$. Then $L_k(W) = L_k(W')$, $R_k(W) = R_k(W')$, and $I_k(W) = I_k(W')$, and yet $W \in$ L and $W' \notin$ L.

There is a simple frequently used method to prove that an event L is not locally testable. The proof will consists in merely specifying schematically words W_k and W'_k such that (1) $W_k \in$ L, (2) $W'_k \notin$ L, (3) $L_k(W_k) = L_k(W'_k)$, (4) $R_k(W_k) = R_k(W'_k)$, and (5) $I_k(W_k) = I_k(W'_k)$.

Although an event in *LTO* is nocounting such an event may be capable of counting to a threshold. For example, the set β of words having at least five 0's is in *LTO*: for the set of α of words having at least one 0 is locally testable; then $\beta = \alpha\ \alpha\ \alpha\ \alpha\ \alpha$ and so $\beta \in LTO$. But as in Chapter 1, to say that β is noncounting is to say that β does not count modulo an integer.

Theorem 2.1: LTO $\subseteq$ NC.

Proof: The proof begins by showing (a) that if α is locally testable, then $\alpha \in NC$. It concludes by showing that, for all $\alpha, \beta \in NC$, (b) $\alpha \cup \beta \in NC$, (c) $\sim\alpha \in NC$, and (d) $\alpha\beta \in NC$.

1. Suppose that, for some k, α is k-testable. It must be shown that there is an n such that, for all U, V, W and $x > 0$, $UV^{n+x}W \in \alpha$ if and only if $UV^nW \in \alpha$. Take $n = k + 1$. If $V = \lambda$, the matter is trivial. If $V \neq \lambda$ then, since the length of V is at least 1 for all x, $L_k(UV^{k+1}W) = L_k(UV^{k+1+x}W)$, $R_k(UV^{k+1}W) = R_k(UV^{k+1+x}W)$, and $I_k(UV^{k+1}W) = I_k(UV^{k+1+x}W)$, which the reader can verify with some reflection. Thus the result follows.
2. If $\alpha \in NC$, then $\sim\alpha \in NC$ by definition.
3. For (b) and (d) suppose $\alpha, \beta \in NC$. Then there exist p and q such that, for all U, V, W and $x > 0$, $UV^{p+x}W \in \alpha$ if and only if $UV^pW \in \alpha$, and $UV^{q+x}W \in \beta$ if and only if $UV^qW \in \beta$.
4. Take $n = max(p, q)$. Then for all x, $UV^{n+x}W \in \alpha \cup \beta$ if and only if either $UV^{n+x}W \in \alpha$ or $UV^{n+x}W \in \beta$; which is so if and only if either $UV^nW \in \alpha$ or $UV^nW \in \beta$; which is so if and only if $UV^nW \in \alpha \cup \beta$.
5. To show that $\alpha\beta \in NC$, take $n = p + q + 1$; and suppose first that $UV^nW \in \alpha\beta$. Then $UV^nW = XY$, where $X \in \alpha$ and $Y \in \beta$. Since $n = p + q + 1$, either p of the V's in UV^nW are part of X or q of the V's are part of Y. Thus, either $X = UV^pW_0$ or $Y = U_0V^qW$. Thus, for all $x > 0$, either $UV^{p+x}W_0 \in \alpha$ or $U_0V^{q+x}W \in \beta$, and thus $UV^{n+x}W \in \alpha\beta$.

Now suppose, for an arbitrary $x > 0$, $UV^{n+x}W \in \alpha\beta$. Let $UV^{n+x}W = XY$, where $X \in \alpha$ and $Y \in \beta$. Then, by reasoning similar to the preceding paragraph, either (Case I) $X = UV^{p+x}W_0$, or (Case II) $Y = U_0V^{q+x}W$, or (Case III) $X = UV^{p+y}W_0$ and $Y = U_0V^{q+z}W$, where $y + z = x$. For Case I, $UV^pW_0 \in \alpha$, since $\alpha \in NC$. Hence $UV^nW \in \alpha\beta$, since $UV^nW = UV^pW_0Y$. For case II, $U_0 V^qW \in \beta$, since $\beta \in NC$. Hence, $UV^nW \in \alpha\beta$, since $UV^nW = XU_0V^qW$. For Case III. $UV^pW_0 \in \alpha$ and $U_0V^qW \in \beta$, since both α and $\beta \in NC$. Thus, $UV^nW = UV^pW_0U_0V^qW \in \alpha\beta$.

The examples of different events are represented by Figures 2.1 to 2.7.

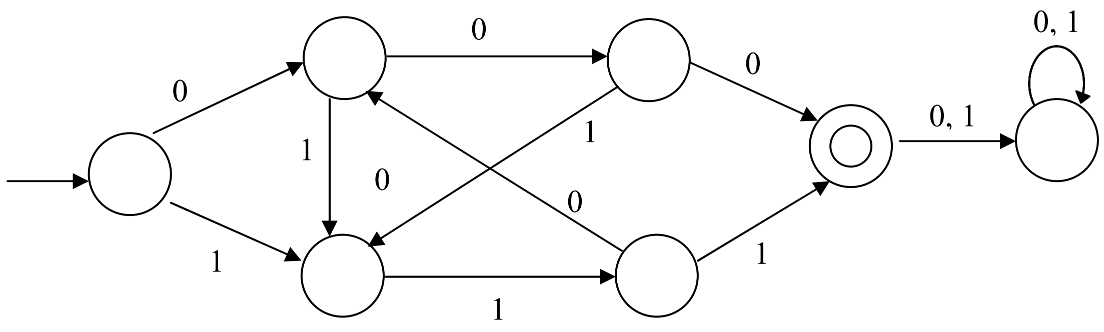

Figure 2.1. Automaton for words end in 000 or 111.

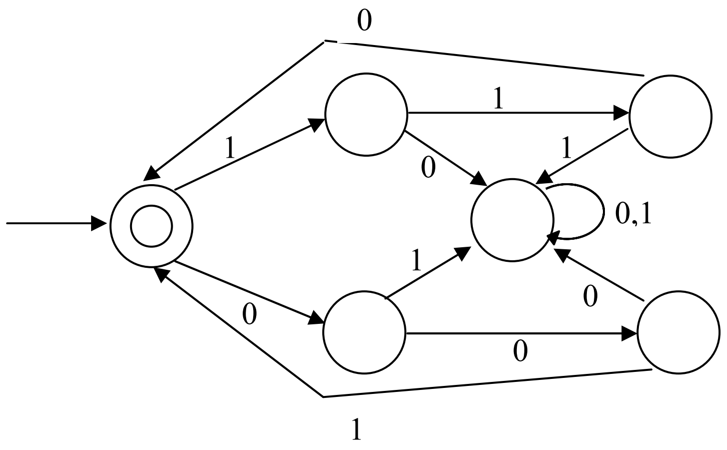

Figure 2.2. Automaton for words end in 110 or 100.

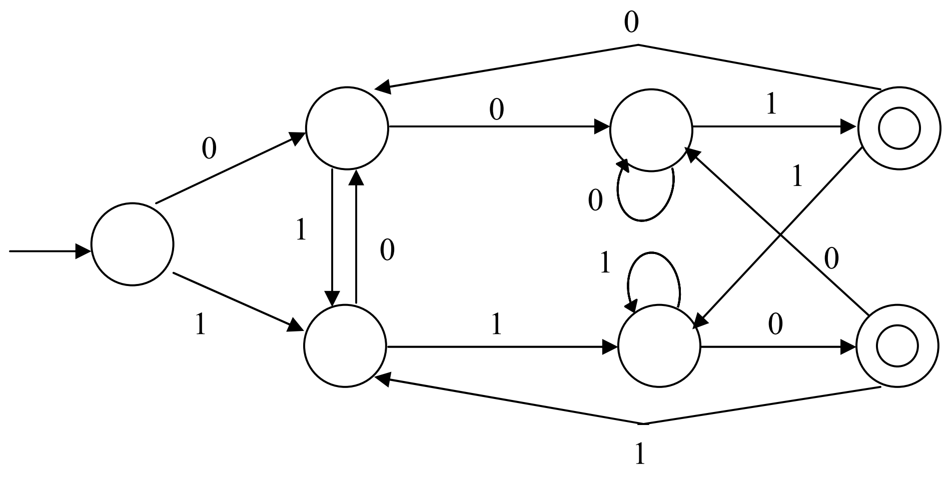

Figure 2.3. Automaton for (110 ∪ 001)*.

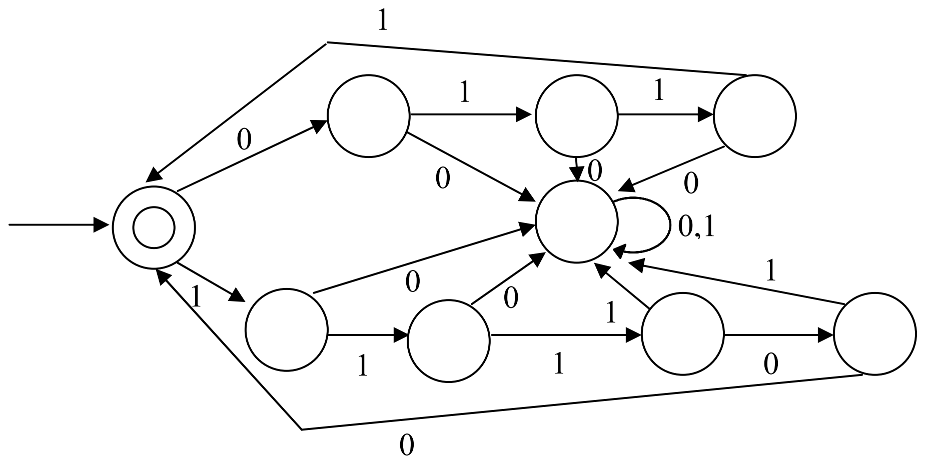

Figure 2.4. Automaton for (0111 ∪ 11100)*.

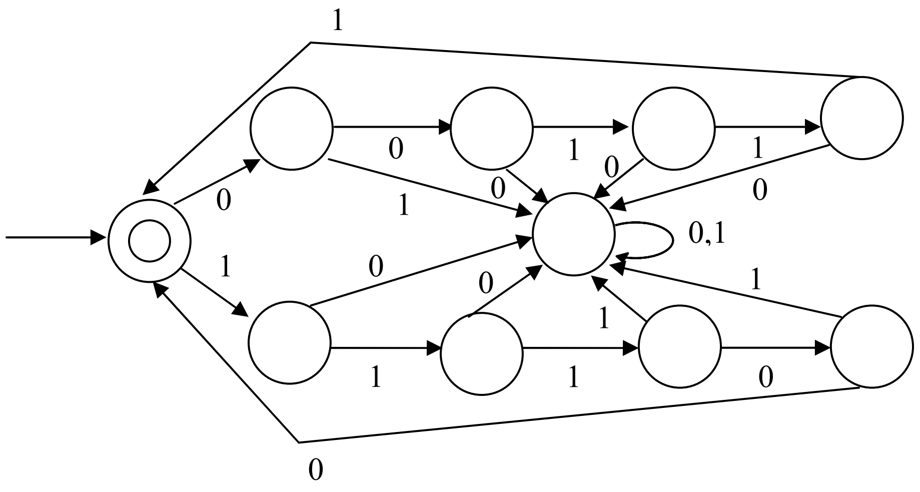

Figure 2.5. Automaton for (00111 ∪ 11100)*.

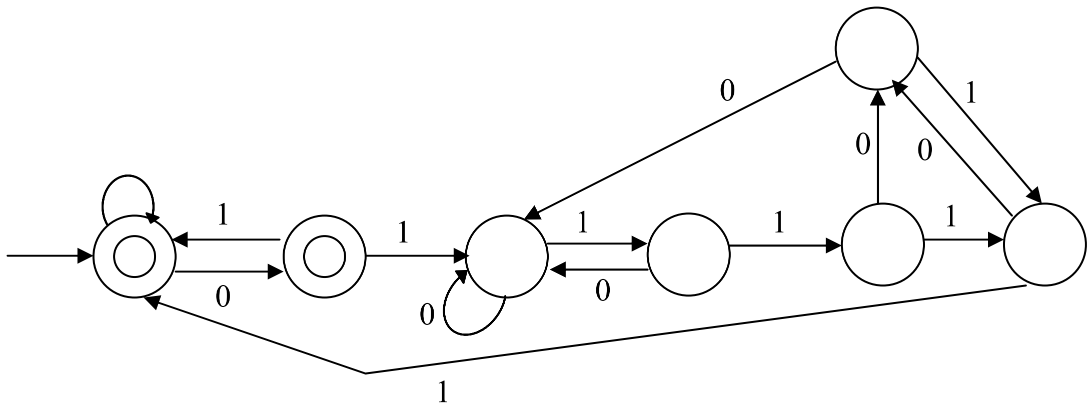

Figure 2.6. Automaton for words end in 11, 01 or 01.

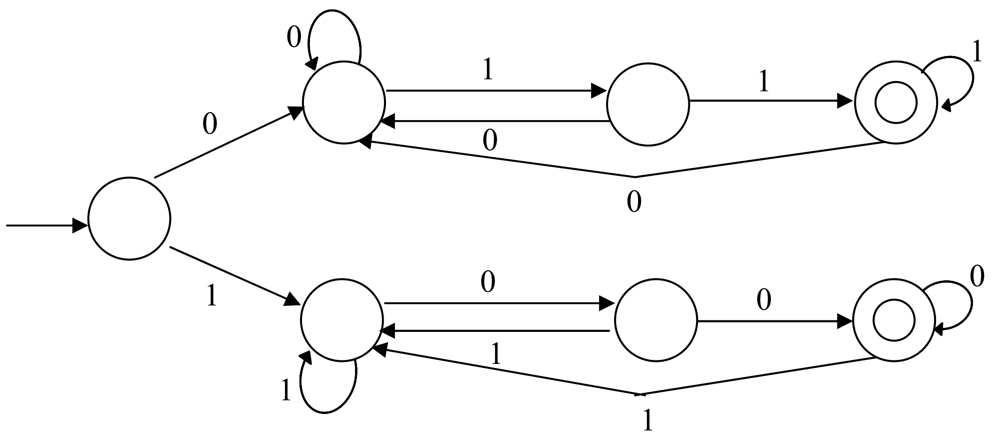

Figure 2.7. Automaton for words end in 0111* or 1000*.

The following are exercise solutions from the book.

1. Decide whether each of the Figures 2.1-2.7 represents a locally testable event.
   Ans:
   Figure 2.1 represents the event, whose words end in 000 or 111, but this true of no proper prefix. It is locally testable for any *W* and *W*' for *k* = 3.
   Figure 2.2 represents the event, whose words end in 001 or 110. It is locally testable event for any *W* and *W*' and for any *k* = 3. It satisfies the definition of locally testable.
   Figure 2.3 represents the event (110 ∪ 001)*. It is locally testable for any *W* and *W*' for *k* = 3.
   Figure 2.4 represents the event (0111 ∪ 11100)*. It is locally testable for any *W* and *W*' for *k* = 4.
   Figure 2.5 represents the event (00111 ∪ 11100)*. It is locally testable for any *W* and *W*' for *k* = 5.
   Figure 2.7 represents the event, whose words end in 0111* (if words start with 0) or 1000* (if words start with 1). For *k* = 3 for any *W*, *W*' it satisfies the definition of locally testable.

2. Prove that the class of locally testable events (*k*-testable events) in the strict sense is not closed under the Boolean operations.
   Ans: Complement:
   Let $\sum = \{0, 1\}$
   Let L_1 = Set of all words of length 3 or more that have no occurrence of 000.
   To see that L_1 is locally testable in the strict sense, let *k* = 3. Then let each of α, β, γ be all three letter words except for 000. If $|W| \geq 1$, then $W \in L_1$ if and only if its length 3 substrings belong to α, β, γ as appropriate.
   Thus L_1 is locally testable language in the strict sense.
   Let, $\sim L_1$ is the complement of L_1.
   Then $\sim L_1$ contains all the words which are not in L_1, which includes words of length 2 or less.
   We show that, $\sim L_1$ is not locally testable in the strict sense. Let magic number be $k \geq$

3, consider α, "initial" strings on length k. α must include all strings with substring 000 or we might "miss" elements of $\sim L_1$. But we must reject long words in $\sim L_1$ whose occurrence of 000 comes later.

So, $LTSS_k$ is not closed under complement.

Therefore the class of locally testable languages in the strict sense is not closed under Boolean operations.

3. Prove that the class of locally testable events (k-testable events) is closed under the Boolean operations.

Ans: Union:

Let L_1 and L_2 be locally testable languages.

Now let, $L = L_1 \cup L_2$.

Assume $L_1 \in LT_{k1}$, $L_2 \in LT_{k2}$.

Now then "test" for what be $W \in L_1$, or $W \in L_2$ is to complete $L_{k2}(W)$, $R_{k2}(W)$, $I_{k2}(W)$) and determine if that is an "accepting " triple of L_1 or an accepting triple of L_2, clearly the union of the accepting triples of L_1 and the accepting triples of L_2 will be set of accepting triples for L.

Complement:

Let L_1 be a locally testable language, and assume $L_1 \in LT_k$,

We must show that $\sim L_1$ is also locally testable. So there must be a k-test which recognizes $\sim L_1$, but $k' = k$ will work. For let W, $W' \in \sum^*$.

Then $L_k(W) = L_k(W')$, $R_k(W) = R_k(W')$, $I_k(W) = I_k(W')$ means $W \in L_1 \Leftrightarrow W' \in L_1$, whence $W \in \sim L_1 \Leftrightarrow W' \in \sim L_1$.

∴ L_1 is closed under complement.

Intersection:

Let L_1 and L_2 are locally testable languages.

Then $\sim L_1$ and $\sim L_2$ are also locally testable languages.

Locally testable languages are closed under union.

So, $\sim (L_1 \cup L_2) \in LT$, By De Morgan's law

$$\begin{aligned} \sim L_1 \cap \sim L_2 &= \sim(\sim (L_1 \cap L_2)) \\ &= \sim(\sim L_1 \cup \sim L_2) \\ &= L_1 \cap L_2 \end{aligned}$$

Therefore locally testable languages are closed under intersection.

4. Prove that the class of the locally testable events (k-testable events) is the Boolean closure of the class of locally testable events (k-testable events) in the strict sense.

Ans: Let $L \in LT_k$. Then a word of length $h \geq k$ is in $L \Leftrightarrow$ the trio (U, S_k', V) is accepting where U = first k letters, V = last k letters and S_k' = set of "interior" length k substrings, so $L = \cup \{W \mid W \text{ has trio } t\}$ where t is accepting. So we are done if we show $L_t = \{W \mid W \text{ has trio } t = (U, S_k', V)\} \in CL\ (LTSS_k, \cup, \cap, \sim)$.

In $LTSS_k$ terms we need $\alpha = \{U\}$, $\gamma = \{V\}$, for β we start with S_k' which is some set of length k strings. Let $\beta = \{W_1, \ldots, W_t\}$ and let $\beta_i = \beta - \{W_i\}$, then $(\mathbb{P}(\beta) - \cup^t_{i=1} \mathbb{P}(\beta_i)) = \beta$. By $L_{(\alpha, \beta i, \gamma)}$ we mean that $LTSS_k$ language given by the strict sense trio $(\alpha, \beta_i, \gamma)$, i. e. $W \in L \Leftrightarrow L_k(W) \in \alpha$, $R_k(W) \in \gamma$, $I_k(W) \subseteq \beta$. So, $L_t = L_{(\alpha, \beta, \gamma)} - \cup^t_{i=1} L_{(\alpha, \beta i, \gamma)} \in CL(LTSS_k, \cup, \cap, \sim)$.

5. Let CLA_k be the class of all events L having the property that, for any W, W' of length k or more, if $\{ L_k(W), R_k(W)\} \cup I_k(W) = \{ L_k(W'), R_k(W')\} \cup I_k(W')$, then either both W and W' are in L or neither are. Give an example of an event L ∈ CLA_1 but such that the event L· L is not locally testable.
Ans: Example:
By the definition of CLA_k class events having the property that, for any W, W' of length k or more, if $\{L_k(W), R_k(W)\} \cup I_k(W) = \{ L_k(W'), R_k(W')\} \cup I_k(W')$, then either both W and W' are in L or neither are.
We can let $L = 0^+ \cup 1^+$, so each word contains just 0 or 1 and complement, not including λ, has words for which the "letter set" is {0,1} hence L ∈ CLA_1, but L·L seems to be in LT since $L\cdot L = 00^+ \cup 11^+ \cup 0^+1^+ \cup 1^+0^+$ and each piece is in LT. Intuitively, a length 2 "window" passing over $W \in 0^+1^+$ sees the strings 00, 01, 11 and no others and that seems to show $0^+1^+ \in LT_2$, with modest labor.
So, we try CLA_2 instead.
Let $L = 00^+11^+$. If $W \in$ L, then if $k = 2$ we have $L_k(W) = \{00\}$, $R_k(W) = \{11\}$, and $I_k(W)$ has 01 and may also have 00 and/or 11 in any case $\{L_k(W), R_k(W)\} \cup I_k(W) = \{00, 11, 01\}$ for each $W \in$ L. Moreover words of length ≥ 2 in ~L must be "different", as is easily seen.
But $L\cdot L = 00^+ 11^+ 00^+ 11^+$ and a width 2 window, for $W \in$ L·L, encounters 00, 01, 11, 10 while starting with 00 and ending with 00. But so does 001100110011 ∉ L·L. So $L_k = \{00\}$, $R_k = \{11\}$, $I_k = \{00, 01, 10, 11\}$ does not characterize L·L, and larger k's suffers a similar fate for some $W \in$ L·L and $W' \in$ ~L·L. Whence L·L is not locally testable.

6. Let $CLA = \cup^{\infty}_{k=1} CLA_k$, where CLA_k is defined in above exercise. Which of the examples listed at the end of Chapter 1 are in CLA?
Ans: By the definition of $CLA = \cup^{\infty}_{k=1} CLA_k$, where CLA_k is defined in above exercise. None of the example listed at the end of Chapter 1 are in CLA. Because to satisfy the definition of CLA, all the words from each events have to satisfy the condition for all positive integer $k \geq 1$. Therefore none of the example is in CLA.

7. Find an example of an infinite sequence of events $\alpha_1, \alpha_2, \ldots$, all over a common alphabet, such that (1) $\alpha_i \subseteq \alpha_{i+1}$ for each i, (2) each α_i is locally testable and infinite, but (3) $\cup^{\infty}_{i=1} \alpha_i$ is not a regular event.
Ans: Example: Let $\sum = \{a,b\}$. Now we have to find an infinite sequence of events $\alpha_1, \alpha_2, \ldots$, where $\alpha_i \subseteq \alpha_{i+1}$ for each i and each α_i is LT and infinite, but also $\cup^{\infty}_{i=1} \alpha_i$ is not a regular event.
Let $\alpha_1 = b^+ \cup \{ab\}$. This is LT and infinite.
Same as α_1, let $\alpha_2 = \alpha_1 \cup \{a^2 b^2\}, \ldots, \alpha_{i+1} = \alpha_i \cup a^{i+1} b^{i+1}\}$ for $i \geq 1$ and so on. As we know the sequence of events satisfies $\alpha_1 \subseteq \alpha_2 \ldots \subseteq \alpha_i$. But $(\cup_i \alpha_i) \cap a^+b^+ = \{a^i b^i \mid i \geq 1\}$ is not regular, and thus $\cup_i \alpha_i$ is not regular.

8. Follow the same instructions as in Exercise 7, except to make $\cup^{\infty}_{i=1} \alpha_i$ regular, but not a noncounting, event.
Ans: Example:
Let $\sum = \{a,b\}$
It is same as the above example except $\cup^{\infty}_{i=1} \alpha_i$ is regular but not a noncounting

event.

Let $\alpha_1 = b^+ \cup \{a^2\}$, as we can see α_1 is regular *LT* and infinite event. But it is not noncounting event.

Same as α_1, let $\alpha_2 = b^+ \cup \{a^4\}, \ldots, \alpha_i = b^+ \cup \{a^{2i} \mid i \geq 1\}$ and so on. We know the sequence of events satisfies $\alpha_1 \subseteq \alpha_2 \ldots \subseteq \alpha_i$ and $\{a^{2i} \mid i \geq 1\}$ is not noncounting.

9. Follow the same instructions as in Exercise 7, except to make $\bigcup^{\infty}_{i=1} \alpha_i$ a noncounting, but not a locally testable, event.
Ans: Example:
Let $\alpha_1 = \{W \in \{0, 1\}^* \mid W$ contains substring 00 but does not contain substring 11$\}$. Let $\alpha_i = \alpha_1 \cdot \{W \mid |w| \leq i$ and W contains 11$\}$. For $i \geq 2$ clearly each word in α_i has 00 followed by 11. Now $\bigcup^{\infty}_{i=1} \alpha_i$ is all words that contain a 00 before they contain 11, and also contain 11. As a regular expression we have $(\Sigma^* - \Sigma^*\, 11\, \Sigma^*) \cdot \{00\} \cdot (\Sigma^*\, 11\, \Sigma^*)$. This is clearly in *LTO* and hence in *NC*. But it is not in *LT*.

CHAPTER 3

REGULAR EXPRESSIONS

Regular expressions are a compact form of representing regular sets of words using letters of an alphabet, special symbols for the empty set and null word, and signs for the various operations on sets of words. Regular expressions with all the Boolean operators are known as “general regular expressions” and those without the sign of intersection and complementation known as “restricted regular expressions”. The precise definition of “*general regular expression* (g.r.e) over an alphabet $\sum$” is a set of strings defined as follows: (1) Any letter of the alphabet is a g.r.e. (2) λ (the null word) and φ (the empty set) symbols are g.r.e.’s. If ψ and χ are g.r.e.’s then so are the strings (3) $(\psi) \cup (\chi)$ (union), (4) $(\psi) \cap (\chi)$ (intersection), (5) $\sim(\psi)$ (the complementation with respect to the set of all words over $\sum$), (6) $(\psi)(\chi)$ (the concatenation of ψ with χ), and (7) $(\psi)^*$ (closure, iteration, star closure, or just plain “star”) are g.r.e.’s. A *restricted regular expression* is a g.r.e. with no occurrence of “$\cap$” or “~”. A *star-free expression* is a g.r.e. with no occurrence of “*”. At this point we can say that “*SF*” is the class of those events that can be represented by star-free expression in the obvious way, since each regular expression clearly denotes some $L \subseteq \sum^*$.

In terms of set theory the definition of *SF* could be said that *SF* is the smallest class of events that (1) includes the empty set, the unit set of the null word, and the unit set of any word of unit length, and (2) is closed under the Boolean operations and concatenation.

The star has two roles in the class of restricted regular expressions: it permits counting and it also is the only operator that can serve to introduce infinite sets. In the class of g.r.e., however, infinite sets of words can be introduced without the aid of stars. And so, if a g.r.e. has a star which cannot be eliminated, we shall know that the event the expression represents is not noncounting. The only function of the star is to introduce counting contexts.

Theorem 3.1. $SF \subseteq LTO$

Proof: We can get a recursive definition of "star-free expression" from the recursive definition of "general regular expression" just by deleting part (7). To prove this Theorem we need to show that the definition of star-free expression satisfies the definition of *LTO*. Now, the events represented by letters of the alphabet, λ, and φ are in *LTO*, since all of these events are locally testable. Let ψ and χ are events represented by star free expressions and if they are in *LTO* then the events represented by $\psi \cup \chi$, $\psi \cap \chi$, $\sim \chi$, and $\psi\chi$ are also in *LTO*, since *LTO* is closed under Boolean operations and concatenation.

There are some defined notations to express the star-free expressions. Let $\sum$ be an alphabet. The symbol "*F*" is formally defined as $\sim\varphi$, which is $\sum^* - \varphi = \sum^*$, and thus it is clear that $\sum^*$ can be definable without the star. We will see that there are many regular-expression events, expressed with the use of star, that can be expressed without using the star. To show that it is possible we shall define some new operators defined in terms of Boolean operators and concatenation.

Definition: $\neg\, \alpha = \sim (F\alpha F)$. The set $\neg\, \alpha$, as defined, is the set of all words in which no member of α occurs as a segment. In simple words, there is an α, which is the set of some words, and $\neg\, \alpha$ means that it is the set of such words in which no word of α occurs as a segment.

Definition: Btw $(\alpha, \beta, \gamma) = \neg\, (\alpha\, (\neg\, \beta)\, \gamma)$. This is the set of all words in which, whenever there is an α occurrence (that is, an occurrence of a segment which is a member of

α) followed somewhere by a γ occurrence, there is a β occurrence somewhere between: the β occurrence must begin after the end of the α occurrence and must end before the beginning of the γ occurrence.

Definition: $\alpha \rightarrow \beta = {\sim}[F\alpha({\sim}(\beta F))]$. This is the set of words in which, for every α occurrence, a β occurrence immediately follows. This operator will be used mostly in such cases where α and β are either single words or finite events containing a small number of relatively short non-null words. Thus $0 \rightarrow 11$ is the set of all words in which a zero is always followed immediately by two 1's. And $(00 \cup 11) \rightarrow 22$ is the set of all words in which a double zero or double one is always immediately followed by a double two. If α and β are finite events then clearly $\alpha \rightarrow \beta$ is a locally testable event.

Definition: $\alpha \leftarrow \beta = {\sim}({\sim}(F\alpha)\beta F)$. This is the set of all words in which every β occurrence is immediately preceded be an α occurrence. Again, same as the above definition, this operator is mostly used when α and β are finite events, in which case $\alpha \leftarrow \beta$ represents a locally testable event.

Now, we can define $LT_k(\alpha)$ to be the set of all words W such that every segment (left end, right end, and interior) of W of length k is in α. Let the finitely many words of length k not in α be $X_1, X_2, \ldots, X_n$. Then $LT_k(\alpha)$ can be defined by the g.r.e

$\neg(X_1 \cup X_2 \cup \ldots \cup X_n)$.

Note that every word over the alphabet of length k-1 or less is in $LT_k(\alpha)$, since it has no segments of length k.

LT_k provides an easy way to represent some k-testable events. For example, let L be the set of words over $\{a,b\}$ that begins with an aa, ends with a bb, but never has an interior occurrence of aa or bb. It can be represented by $aaFbb \cap aLT_2(ab \cup ba)b$. Note that, for any

α, $LT_k(\alpha)$ is always k- testable in the strict sense, since it will always satisfy the definition of locally testable in the strict sense. There are some events, which are k- testable but not in the strict sense, that can be represented by means of the operator LT_k. For example, the set of all words that either have both the segments aa and bb or have neither is represented by the expression $LT_2(ab \cup ba) \cup [FaaF \cap FbbF]$.

By using the operator LT_k we can easily represent any event L that is k-testable in the strict sense. Let L be the set of words W such that $L_k(W) \in \alpha$, $I_k(W) \subseteq \beta$, $R_k(W) \in \gamma$. Then

$L = (\alpha \cap \sum^k)F \cap \sum LT_k(\beta) \sum \cap F(\gamma \cap \sum^k)$.

Using these ideas we can show the following theorem.

Theorem 3.2 LT ⊆ SF.

See exercise 1.

Figure 3.1 represents a noncounting event. Two regular expressions for it are $(0 \cup 1)^* 11(01 \cup 1)^*$ and $(F11 \neg (00)) \cap F1$, of which the latter shows the event to be in SF. It is not locally testable.

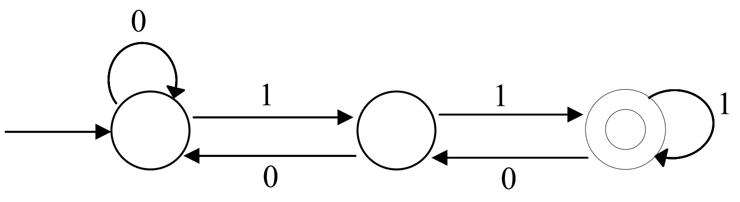

Figure 3.1. Automaton for $(0 \cup 1)^*$ 11(01 ∪ 1)*.

This is an example of a manually operated elevator that services three floors, with input at the operator's handle and “output” at the third floor. At any time the operator can turn the handle to the down position (input 0) or to the up position (input 1); the middle position of the handle is no input at all, in which case the elevator does not react. If the elevator reacts at the third floor and the operator turns the handle to “up,” then the elevator reacts by merely remaining where it is; and similarly it remains where it is if he turns the handle to "down" when the elevator is at its lowest floor. The output at the third floor is on when the elevator is there and off otherwise. This is a discrete interpretation of a device that is actually continuous, and ignores what happens in case the operator reverses directions between floors. Nevertheless, we are fond of referring to Figure 3.1 as "the elevator automaton".

A code event is an event of the form $(W_1 \cup \ldots \cup W_n)^*$, where each W_i is a word. The code is the set $\{W_1, \ldots, W_n\}$. Any word in the event can be thought of as a message. Let us see, how this code can be locally testable. If $W \in$ L and L is a code event, then W can be parsed into phrases so that each phrase is one of the W_i's; a sufficient condition for L to be locally testable is that it be locally parsable.

To define this concept, we recall first a concept from coding theory: a code $\{W_1,\ldots, W_n\}$ is unambiguous if $W_{i1}, W_{i2}\ldots W_{is} = W_{j1}, W_{j2}\ldots W_{jt}$ implies that $s = t$ and, for each x, $i_x = j_x$; viz., every word in $(W_1 \cup \ldots \cup W_n)^*$ can be parsed in only one way. We can consider the operation of parsing physically: a word is parsed if slashes are written between certain letters of the word so that the portions between slashes, and also the portions to the left of the leftmost slash and to the right of the rightmost slash, are each one of the W_i's. {0, 10, 01} is an example of an ambiguous code, since 010 can be parsed in two ways: 0/10 and 01/0.

A code $\{W_1,\ldots, W_n\}$ is k-parsable if it is unambiguous and if, for any space between letters in a word in $(W_1 \cup \ldots \cup W_n)^*$, one can tell whether to place a parsing line there by looking only at the letters of the word that are within k letters of that space. An event L is k-parsable if there exists a k-parsable code $\{W_1,\ldots, W_\mathrm{n}\}$ such that L $= (W_1 \cup \ldots \cup W_n)^*$. And a code, or event, is locally parsable if it is k-parsable for some k.

For example, the event $(10101)^*$ is locally parsable, and in fact 1-parsable. For a space in any word in $(10101)^*$ gets a parsing line if and only if it has a 1 on both sides.

A parsing machine for a *k-parsable* event is capable of scanning $2k$ squares at once. It moves along a large tape containing a word from $(W_1 \cup \ldots \cup W_n)^*$ placing parsing lines in exactly the right places. The significance of local parsability is that this machine, although it would have to examine every space between two letters, could examine them in any order.

Local parsability implies local testability. In fact, if a code $\{W_1, \ldots, W_n\}$ is k-parsable and e is the length of the longest word among the W_i's, then the event $(W_1 \cup \ldots \cup W_n)^*$ is $(2e + 2k - 1)$-testable. The proof is an exercise.

Thus, $(10101)^*$ is locally testable. In fact,

$$(10101)^* = \lambda \cup [10101F \cap LT_5(10101 \cup 01011 \cup 10110 \cup 01101 \cup 11010) \cap F10101].$$

This event can also be described as a star-free expression by using the arrow. This is

$$(10101)^* = \lambda \cup [10101F \cap (1010 \rightarrow 1) \cap \neg(01010) \cap (1011 \rightarrow 0) \cap (0110 \rightarrow 1) \cap (1101 \rightarrow 0)].$$

An example of an unambiguous code that is not locally parsable is $\{01, 10\}$. For consider $V_k = (01)^{2k}$, for any k. if this code were k-parsable, then whether or not the center space of V_k has a parsing line would be determined for any U,W such that $UV_kW \in$ L. However, if $U = 1$, $W = 0$ there is no parsing line there, but if $U = W = 01$, there is one.

We can prove a little theorem about events of the form W^*, generalizing from the example $(10101)^*$ just considered.

A word V is a *cyclic transform* of W if these are words X_1 and X_2 such that $W = X_1X_2$ and $V = X_2X_1$. If neither X_1 nor X_2 is null, then V is a *proper* cyclic transform of W.

Lemma 1. If W is not null and is equal to one of its proper cyclic transform, then there is a V and a $k \geq 2$ such that $W = V^k$.

Proof: (1) Let V be the shortest non-null initial segment of W such that, for some X, $W = VX = XV$. Then, (2) let k be the largest integer such that, for some Y, $W = V^kY$. Note that $V^kY = YV^k$. The proof is complete if we can prove that $Y = \lambda$, so let us assume that Y has positive length.

Case I: The length of Y is less than the length of V. Then, since $W = V^kY = YV^k$, stipulation (1) is contradicted.

Case II: The length of Y is equal to or greater than the length of V. Then, since $YV^k = V^kY$, it follows that, for some Y_0, $Y = VY_0$, and thus $W = V^{k+1}Y_0 = Y_0\, V^{k+1}$, contradicting stipulation (2).

Theorem 3.3: For every non-null word W: if $W = V^k$ for $k \geq 2$, then $W^* \notin NC$; if $W \neq V^k$, for any V and $k \geq 2$, then W^* is locally testable.

Proof: If $W = V^k$, $k \geq 2$, then, for arbitrarily large x, $V^{kx} \in W^*$ but $V^{kx+1} \notin W^*$. Hence $W^* \notin NC$.

On the other hand, if such is not the case (for any V and $k \geq 2$), then W is not equal to any of its proper cyclic transforms by lemma 1. Then W^* is locally parsable: slash a space in any word of W^* if and only if it is flanked on both sides by a W.

Theorem 3.3 implies that, for every W, W^* is locally testable if it is in NC. For even slightly more complicated kinds of events, events of the form $(W_1 \cup W_2)^*$, this proposition is no longer true. An example is $(01 \cup 10)^*$, which is not locally testable but is in NC. A star free expression for this event is:

$\lambda \cup [(01 \cup 10)F \cap F(01 \cup 10) \cap \neg(000 \cup 111) \cap$

$\sim (0 \neg (11)00F) \cap \sim (1 \neg (00)11F) \cap \sim (F00 \neg (11)0) \cap$

$\sim (F11 \neg (00)1) \cap \sim (0 \neg (11)0 \cap \sim (1 \neg (00)1) \cap$

$\mathrm{Btw}(00, 11,00) \cap \mathrm{Btw}(11,00,11)]$.

The following are exercise solutions from the book.

1. Prove that $LTO \subseteq SF$ (which will include a proof of Theorem 3.2, that $LT \subseteq SF$).
   Ans: Text shows that locally testable in the strict sense events can always be represented by Star-free Expressions. Following is the star-free expression for the definition of the locally testable in the strict sense event.

   $L = (\alpha \cap \sum) (\sim\varphi) \cap \sum LT_k (\beta) \sum \cap (\sim\varphi) (\gamma \cap \sum^k)$

   $\therefore LTSS_k \subseteq SF$

   Now as we know, Locally testable languages are closed under Boolean operation of Locally testable in the strict sense.

   Since, *LT* languages are closed under Boolean operations.

   i.e closure (LT, $\cup$, $\cap$, ~)

   $\therefore LT_k \subseteq SF$

   We know that *LTO* contains all the locally testable events and is closed under Boolean operations and concatenation.

   As described in theorem 3.1, the events represented by letters of the alphabet, λ, and φ are in *LTO*.

   Now by the definition of star free expression, it is closed under Boolean operations and concatenation.

   $\therefore LTO \subseteq SF$

2. Express the following ideas in terms of the operators defined. The set of all words such that if any segment is in α then there is a segment in β. The set of all words in which, if there is an initial segment in α, then there is a terminal segment in β. The set of words in which no α segment in followed immediately by a β segment. (you may assume that $\lambda \notin \alpha$ and $\lambda \notin \beta$.)
   Ans:
   $L = \neg (F\alpha F) \cup [\neg(F\alpha F) \cap \neg(F\beta F)]$
   $L = \sim (\alpha F) \cup [\sim(\alpha F) \cap \sim(F\beta)]$
   $L = \neg (F\alpha\beta F)$

3. Each of the state graphs of Exercise 1 of Chapter 2 represents a noncounting event. Represent each of the five as a star-free expression. (Figures 2.4, 2.5, and 2.6 are more difficult than the others.)
   Ans: 2.1

   $(F\,000 \cup F111) - (F\,000\,F(0 \cup 1) \cup F\,111\,F\,(0 \cup 1))$

   2.2

   $F\,(001 \cup 110)$

   2.3

   $\lambda \cup (001F \cup 110F) \cap (001 \rightarrow \{110, 001\} \cap 110 \rightarrow \{110, 001\})$

2.7

$0F \cap 11 \cup 1F \cap 00$

4. Do exercise 1 of chapter 2 if you have not already completed it.
   Ans: Done.

5. Prove that if W^* is locally testable then it is e-testable, where e is the length of W.
   Ans: Let L = W^* and let $e = |W|$. Then if $W = a_1 \ldots a_e$, and $Z \in$ L, we have 3 cases:
   $|Z| = e$, $W = L_e(Z) = R_e(Z)$, $I_e(Z) = \varphi$.
   $|Z| = 2e$, $W = L_e(Z) = R_e(Z)$, $I_e(Z)$ = all cycles of W except W.
   $|Z| \geq 3e$, $W = L_e(Z) = R_e(Z)$, $I_e(Z) = CYCLE(W)$.
   We assume L is locally testable. We know W has e distinct cycles. *i.e* $|W| = e = |CYCLE(W)|$. So let $Z \in \sum^*$. If Z is in Case (1) then Z has some local attributes as W and trivially $|Z| = e$ and $Z = W$. if Z is in Case (2). It is easy to see $|Z| \geq 2e$ and if $|Z| = 2e$, clearly $Z = W^2$. For now skip the case that $|Z| > 2e$ and looks like (2).
   Now we assume Z is in Case (3) and we wish to show $Z \in$ L, it is easily seen $|Z| > 2e$ and $Z = WZ'$. So let $Z = a_1 \ldots a_e b \ldots$ we want to show $b = a_1$. Now $a_1 = a_2 \ldots a_e a_1 \in CYCLE(W)$ and by Case (3), $a_2 = a_2 \ldots a_e b \in CYCLE(W)$. But clearly $b(a_2) > b(a_1)$ if $b \neq a_1$, and this is impossible if a_1 and a_2 are both cycle of W. Hence $b = a_1$ similarly if $Z = a_1 \ldots a_e\, a_1 b \ldots$.We see $b = a_2$. By the obvious induction any prefix of Z has the form $W^j U$ where $U \leq W$. since $R_k(Z) = W$. We must have $Z \in$ L $= W^*$. Our omitted cases for where Z is in Case (2) can be handled similarly.

6. Show that $(10101)^*$ although 1-parsable is not 4-testable. Let m be an arbitrary positive integer: show that there exists a code event that is 1-parsable but not m-parsable.
   Ans: By using the fact that local parsability implies local testability. If a code $\{W_1, \ldots, W_n\}$ is k-parsable and e is the length of the longest word among the W_i's , then the event $(W_1 \cup \ldots \cup W_n)^*$ is $(2e + 2k - 1)$ – testable. It is clear that the value of $(2e + 2k - 1)$ – testable would be odd value. Therefore, there is no way it is 4-testable.
   Show that L is not 4-testable. For $W \in$ L, $L_4(W) = 1010$, $R_4(W) = 0101$ and $|W| = 5$ so, $I_4(W) = \varphi$ and if $|W| > 5$ then, $I_4(W) = \{0101, 1011, 0110, 1101, 1010\}$. Consider $W' \in$ ~L to be 101010101101010101 and $W \in$ L to be 1010110101. We observe $L_k(W) = L_k(W')$, $R_k(W) = R_k(W')$, and $I_k(W) = I_k(W')$ so it is clear that it is not 4-testable.
   Find for every m, W^* which id 1-parsable and not m testable for m even let $W = (10)^{m/2}\,1$ (e.g. $m = 6$ gives 1010101). Everything in (a) works analogously for m odd remember that L is m testable then L(m +1) testable.

CHAPTER 4

THE SYNTACTING MONOID OF A REGULAR EVENT

We will discuss some basic concepts of monoids and semigroups in finite automata. Let us start with the definition of semigroup. A *semigroup* is a set of elements S together with a binary product on these elements (the product of a and b being written ab) satisfying the following two stipulations:

1. $ab \in S$ for $a, b \in S$
2. $(ab)c = a(bc)$ for all $a, b, c \in S$.

If, in addition there is an identity (or neutral element) e such that

3. $ea = ae = a$ for all $a \in S$,

then S is a *monoid*. For example, $\{2n + 1 \mid n \geq 0\}$ is a semigroup only under addition, whereas 2n is a monoid. Using this concept, a group can be defined as a monoid whose every element a has an inverse a^{-1} such that $a^{-1}a = aa^{-1} = e$.

The set $\sum^*$ of all words over an alphabet $\sum$ is a free monoid whose identity is the null word λ, concatenation of words being the monoid product. If ψ is a homomorphism of $\sum^*$ to a monoid, then of necessity $\psi(\lambda)$ is the identity of this monoid. The null word λ is very important because its presence semisubgroup a submonoid. Let us discuss some notations used in this chapter. Let ψ be a mapping, such that, for any word $W \in \sum^*$, $\psi(W)$ is an element of a monoid M. To say that ψ is a homomorphism is to say that $\psi(WV) = \psi(W)\psi(V)$, for all $W,V \in \sum^*$. For any set α of words in $\sum^*$, $\psi(\alpha)$ is the set of all elements m in M such that for some word $W \in \alpha$, $\psi(W) = m$. Here $\psi^{-1}(W)$ is the set of words that map to m under ψ. And for any subset S of M, $\psi^{-1}(S)$ is the set of all words in $\sum^*$ that map to a member of S under ψ.

We can say that W_1 is *congruent* to W_2 modulo L, or $W_1 \equiv W_2$ (*mod* L), if, for all words V and X, $VW_1X \in$ L if and only if $VW_2X \in$ L. A *congruence class* of words modulo L is a nonempty set having, with any member, all and only all the words congruent to it modulo L. Clearly congruence is an equivalence relation.

There are some homomorphic mappings of $\sum^*$ for an event L, one of them is mapping to the *syntactic monoid* of L, or *Syn*(L). The elements of *Syn*(L) are the congruence classes modulo L. Now defining a mapping of $\sum^*$: each word is mapped to the congruence class modulo L of which that word is a member; it is a many-one mapping because every word over $\sum$ is a member of one and only one congruence class.

Now, to make *Syn*(L) a monoid we must bring in a product, establish that it is associative, and establish the existence of an identity. We note that, by exercises 4.1, if α and β are congruence classes, then for some congruence class γ, $\alpha\beta \subseteq \gamma$. That is to say, the congruence class γ contains all words of the form WW', where $W \in \alpha$, $W' \in \beta$. This examination provides the product operation: thus the product of two congruence classes α and β is defined as that congruence class γ such that $\alpha\beta \subseteq \gamma$. That this operation is associative follows from the fact that concatenation is associative. See exercise 4.2. Here, we consider the same concept for the right congruence classes induced by L, with restriction of suffix. We define $W_1\ R_L\ W_2$ if for every $X \in$ L, $W_1\ X \in$ L *iff* $W_2\ X \in$ L. I.*e*. Let $[u]_{RL}$ and $[v]_{RL}$ be right congruence classes. Then $[u]_{RL} \cdot [v]_{RL}$ not $\subseteq [uv]_{RL}$. For example let L = $\{ac, bc, cd, dd\}$. Now $[a]_{RL} = \{a, b\}$ [words such that only suffix $z = c$ goes to final state] and $[c]_{RL} = \{c, d\}$ [only the suffix d gets us to L]. And $[a]_{RL} \cdot [c]_{RL} = \{ac, ad, bc, bd\}$ but not $ac\ R_L\ ad$. Thus

$[a]_{RL} \cdot [c]_{RL}$ is not contained in any R_L congruence class. And in particular is not a subset of $[ac]_{RL}$. Also the congruence class containing λ is an identity of the semigroup of congruence classes. Hence a monoid formed by the congruence classes is called the "syntactic monoid" because its definition is in terms of the congruence of words modulo an event. The mapping is clearly a homomorphism, and we have that "congruence induced by L" can differ from "right congruence induced by L".

Theorem 4.1. For any event L over an alphabet ∑, *Syn*(L) as defined is a monoid. If ψ is the homomorphism of $\sum^*$ to *Syn*(L), then L is closed under $\psi^{-1}\psi$. And, for any words W and W' over ∑, $\psi(W) = \psi(W')$ if and only if $W \equiv W'$ (*mod* L).

Proof: The first sentence of the theorem is proved by the discussion immediately preceding the theorem. And to prove the second sentence, it suffices to prove that any two words W and W' mapping to the same element under ψ are either both in L or both outside of L. But if W and W' map to the same element they must be congruent modulo L; thus $W = \lambda W \lambda \in L$ if and only if $W' = \lambda W' \lambda \in L$. The last sentence is just a restatement of the definition of "syntactic monoid."

However, there are other such homomorphisms ψ of $\sum^*$ such that L is closed under $\psi^{-1}\psi$. The relationship between the image monoids of these and the syntactic monoid is important. The following is a generalization of Theorem 4.1.

Theorem 4.2. A necessary and sufficient condition that L be closed under $\psi^{-1}\psi$, where ψ is a homomorphism, is that, for all W and W', $\psi(W) = \psi(W')$ implies $W \equiv W'$ (*mod* L).

Proof of sufficiency: Assume the condition. For any $W \in \sum^*$, let $\alpha_w = \psi^{-1}\psi(W)$. Taking β = α_w, it follows that $\psi^{-1}\psi(\beta) = \beta$. We shall show that $\psi^{-1}\psi(L) = L$ by showing that L = β.

That L $\subseteq \beta$ follows immediately from the definition of β. Suppose now that W is an arbitrary word of β. There is a W' $\in$ L such that $\psi(W') = \psi(W)$ implying that $W \equiv W'$ (*mod* L). But then $W \in$ L also. ($W' = \lambda W' \lambda \in$ L, which implies by the definition of "congruence" that $\lambda W \lambda = W \in$ L).

Proof of necessity: Assume that $\psi^{-1}\psi(L) = L$. We must prove that, for every W and W', $\psi(W) = \psi(W')$ implies $W \equiv W'$ (*mod* L). Assume $\psi(W) = \psi(W')$, and let V and X be arbitrary words. $\psi(VWX) = \psi(V)\ \psi(W)\ \psi(X) = \psi(V)\ \psi(W')\ \psi(X) = \psi(VW'X)$. But this implies that $VWX \in \psi^{-1}(\psi(VW'X))$. But since $\psi^{-1}\psi(L) = L$, $VWX \in$ L if and only if $VW'X \in$ L. Hence, $W \equiv W'$ (*mod* L).

Theorem 4.3. If $\psi^{-1}\psi(L) = L$, where ψ is a homomorphism, then the monoid $M = \psi(\sum^{*})$ can be mapped homomorphically onto *Syn*(L).

Proof: By Theorem 4.2, each inverse image class of ψ is a subclass of some congruence class modulo L. Let π be the mapping of M onto *Syn*(L) such that, for each $s \in M$, $\pi(s)$ is the congruence class modulo L of which $\psi^{-1}(s)$ is a subclass. Clearly, the mapping is onto. We need prove only that π is a homomorphism; in other words, that, for each s_1 and $s_2 \in M$, $\pi(s_1 s_2) = \pi(s_1)\ \pi(s_2)$. But, since every element in M is the image under ψ of some word in $\sum^{*}$, it suffices to show that, for all words W_1 and W_2, $\pi(\psi(W_1)\ \psi(W_2)) = \pi(\psi(W_1))\ \pi(\psi(W_2))$. Since ψ is a homomorphism, $\psi(W_1)\ \psi(W_2) = \psi(W_1 W_2)$. Therefore, it suffices to prove $\pi(\psi(W_1 W_2)) = \pi(\psi(W_1))\ \pi(\psi(W_2))$, or, in other words, that $\pi\ \psi$ is a homomorphism.

Figure 4.1 is helpful in visualizing the proof of Theorem 4.3, and shows the similarity of the theorem to many other theorems in algebra for which the same figure has often been given.

$$\begin{tikzcd}
\sum^{*} \arrow[r, "\psi"] \arrow[dr, "\pi\,\psi"'] & M \arrow[d, "\pi"] \\
 & Syn\,(L)
\end{tikzcd}$$

Figure 4.1. Proof of Theorem 4.3.

Whatever we discussed till now applies whether the monoid $\psi(\sum^{*})$ is finite or infinite. We are interested in this monograph only in regular events, and the following theorems will tell us that we can confine our attention to those ψ for which $\psi(\sum^{*})$ is finite.

As we know, a given event regular event can be obtained by a state graph. Methods for obtaining a state graph from some other representation of the event are well known. If G is a state graph, then let $G(s,W)$ be the state reached by a path beginning at s and spelling out W. The state $G(s,W)$ is always unique in a state graph. Let $G(W)$ be $G(s_0, W)$ where s_0 is the initial state. Thus $G(\lambda)$ is the initial state itself.

A *reduced state graph G* for a regular event L is one in which (1) for every state s of G there is a word W such that $s = G(W)$, and (2) for every pair of states s and s' such that $s \neq s'$, there is a W such that either $G(s, W)$ is terminal state and $G(s', W)$ is not a terminal state or vice versa.

Theorem 4.4. Every regular event L has a reduced state graph which is unique up to isomorphism, and has fewer states than any other state graph for L, and is algorithmically determinable from any state graph for L.

For a proof see Moore 1956, or any of several books written since Moore's paper.

Theorem 4.5. For any words W and W' and any regular event L, $W \equiv W'$ (*mod* L) if and only if, for every state s in the reduced state graph G for L, $G(s,W) = G(s,W')$.

Proof: Suppose first that, for some state s in G, $s_1 = G(s, W) \neq G(s, W') = s_2$. There is a path from the initial state to s; suppose it spells out V. Then there is an X such that either $G(s_1, X)$ is a terminal state and $G(s_2, X)$ is not or vice versa. Thus, $VWX \in$ L but $VW'X \notin$ L, or vice versa, which implies that $W \neq W'$ (*mod* L).

Suppose now that, for every s in G, $G(s, W) = G(s, W')$. Then, for every V, $G(VW) = G(VW')$, and hence, for every X, $G(VWX) = G(VW'X)$. Hence, $VWX \in$ L if and only if $VW'X \in$ L; that is, $W \equiv W'$ (*mod* L).

Given a state graph G, we may associate each word W with that mapping on the state of G that, for every state s, sends s to $G(s, W)$. Theorem 4.5 says congruent words modulo L are those which induce the same map on the states of the reduced state graph G for L. It is well known that a monoid (indeed any semigroup) is a set of maps of a certain space closed under composition; the monoid identity is the identity map of the space. The syntactic monoid of a regular event is a set of distinct maps on the state space of the reduced state graph. As a corollary of Theorem 4.5, we see that the syntactic monoid induced by any regular event is finite.

Theorem 4.5 gives us a useful computational method for getting the syntactic monoid, which we shall present by example. Consider the state graph G of Figure 4.2, clearly reduced. In order to keep track of congruence classes of words, we must see what happens when each word is applied to each of the states. Construct a new state graph G' whose initial state is labeled qrs and such that, for each word W, $G'(W) = G(q, W)\ G(r, W)\ G(s,W)$. The state graph *G*’ can be constructed in a step-by-step fashion, noting that *G*’(0) = *qqr*, *G*’(1) = *rss*, *G*’(00) = *qqq*, *G*’(01) = *rrs*, etc. The result is Figure 4.3.

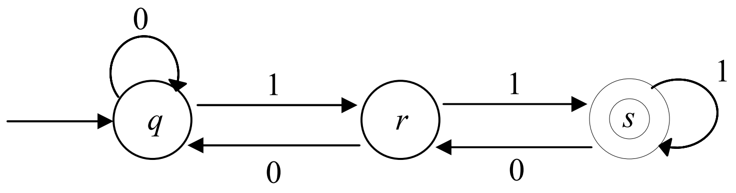

Figure 4.2. Automaton for (0 ∪ 1) 11*.

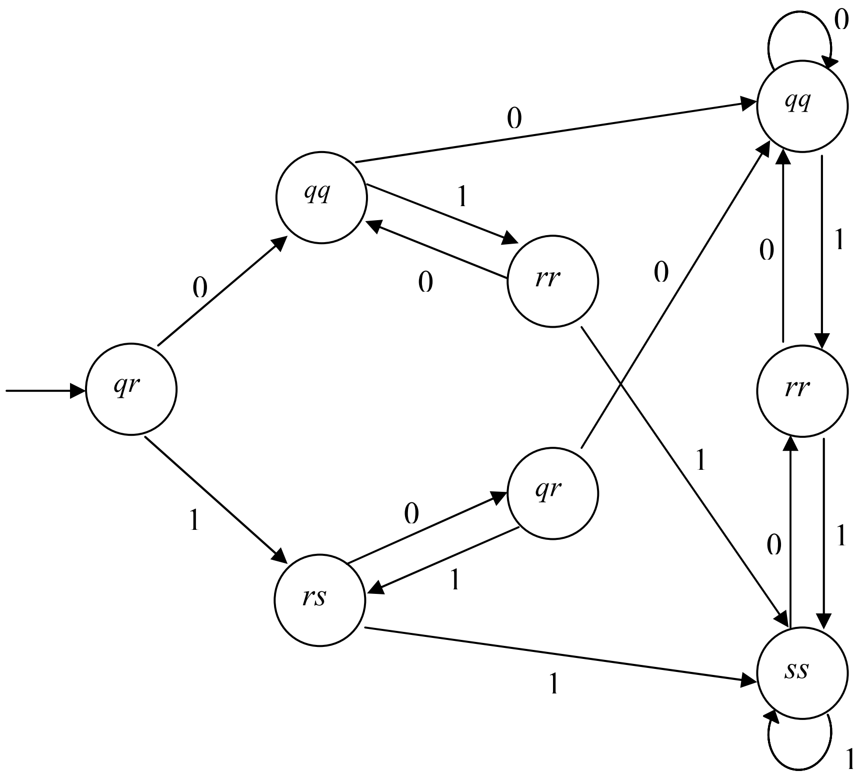

Figure 4.3. State graph for Figure 4.2.

Each state of *G*’ represents a congruence class of words modulo the event given by *G*. For any state *s*’, the congruence class is the set of all words *W* such that *G*’(*W*) = *s*’. Thus the congruence class corresponding to *qrs* is the unit set of [λ]; *qqr* is the 0(10)*, etc. By the construction method and Theorem 4.5, it should be clear that *G*'(*W*) = *G*'(*W*') if and only if *W* ≡ *W*' (*mod* L). Thus *G*' represents the syntactic monoid of *G*.

If we let representative words in each congruence set be names for the classes, except for using "*m*" for *rrr*, the multiplication table for this monoid is Figure 4.4. This table can be identified with the monoid, and any small semigroup can be practically presented in tabular form.

	λ	0	1	01	10	00	11	m
λ	λ	0	1	01	10	00	11	*m*
0	0	00	01	*m*	0	00	11	*m*
1	1	10	11	1	*m*	00	11	*m*
01	01	0	11	01	*m*	00	11	*m*
10	10	00	1	*m*	10	00	11	*m*
00	00	00	*m*	*m*	00	00	11	*m*
11	11	*m*	11	11	*m*	00	11	*m*
m	*m*	00	11	*m*	*m*	00	11	*m*

Figure 4.4. Multiplication table for Figure 4.3.

But note that the state graph of Figure 4.3 also presents the same information; the multiplication of any two elements can be obtained in a scan that is almost as easy as looking the multiplication up in the table. For example, if we wish to find the product of the element 10 with *m* we note that *m* = 110 and then trace out 10110 from the initial state and land at the circle marked *rrr*. We then recall that *rrr* represents 100 = *m*, and we have completed the multiplication. In order to facilitate this process, we relabel the state graph of Figure 4.3 to get Figure 4.5 and call it "the monoid graph." This will be the manner of presenting the syntactic monoid of a regular event.

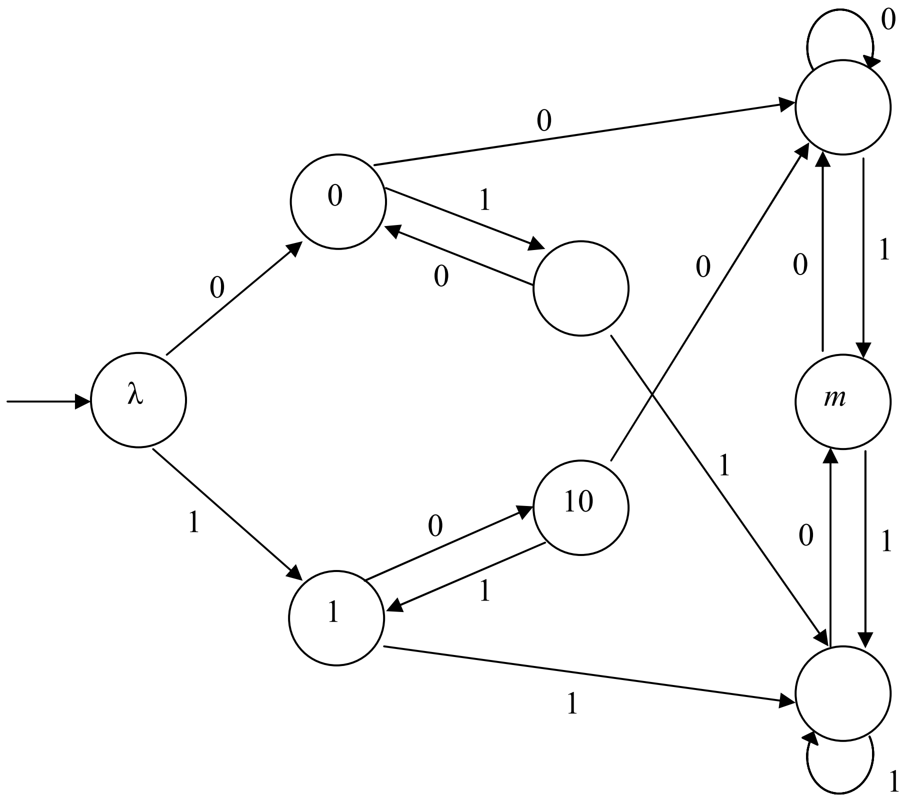

Figure 4.5. Syntactic monoid for Figure 4.2.

We give three more examples. Figure 4.6a is a reduced state graph for an event and Figure 4.6b is its syntactic monoid. A two-headed arrow labeled 1 (or 0) means that a 1(or 0) goes from each state to the other.

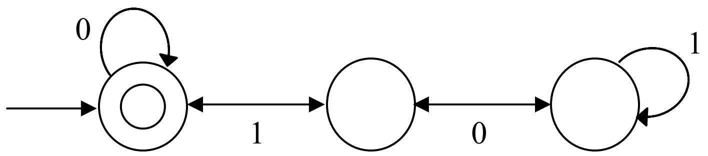

Figure 4.6a. State graph for an event (0 ∪ 1)*(1∪ 0).

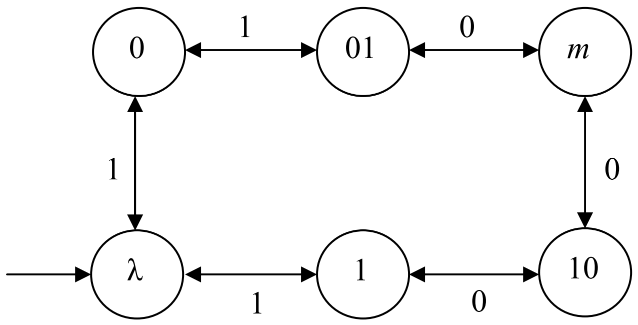

Figure 4.6b. Syntactic monoid for Figure 4.6a.

Figures 4.7 and 4.8 are reduced state graphs of two events. It turns out that the syntactic monoid of each of these events is isomorphic to the state graph, which is why separate graphs for the syntactic monoid are not given.

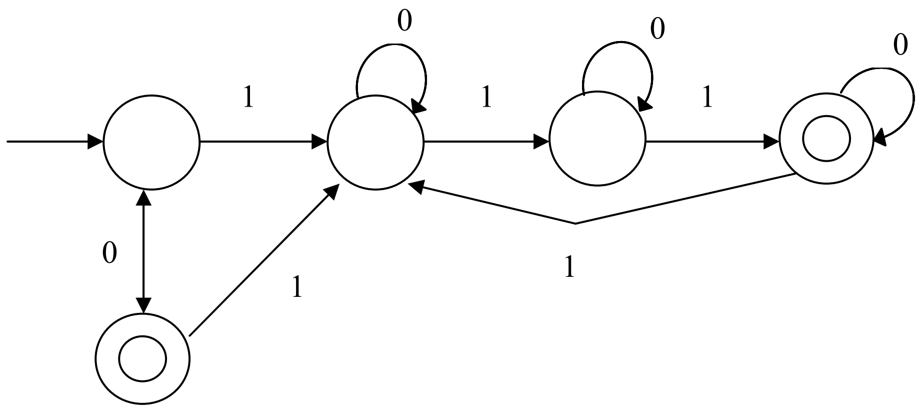

Figure 4.7. State graph for words ending in 0 or 11110*.

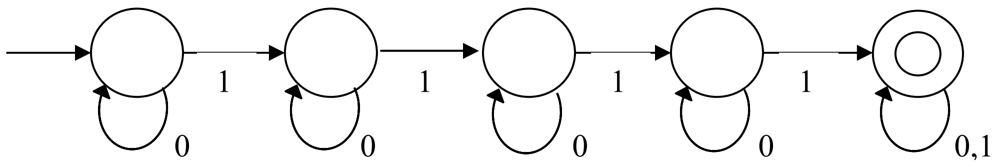

Figure 4.8. State graph for words ending in (0∪ 1) 1111* (01).

In the last few pages we have happened to know the solution of the problem whether a given graph is the graph of a monoid. The only method known is to treat such a graph as a reduced state graph for an event and construct the syntactic monoid graph for it. Then the original graph is a monoid graph if and only if it is isomorphic to the graph that results by this construction.

It turns out to be important in many applications to investigate the subgroups of the syntactic monoid. A *subgroup* of a monoid is defined to be a subset of the monoid that turns out to be a group under the product operation of the monoid. However, a strong word of caution on this point: the identity *i* of a subgroup need not be the identity of the original monoid. In fact, every idempotent of the monoid is the identity of some subgroup of the monoid, even though it may be just the *trivial group* (that is, the group of order 1) consisting of the idempotent all by itself.

Given a subgroup *G* whose identity is *i*, we shall say that *G* is a subgroup *around* the idempotent *i*. It turns out that every idempotent of a monoid has a maximal group around it, which will follow from Theorem 4.9.

Note that, in the monoid of Figure 4.5, all the elements are idempotent except 0 and 1. The monoid of Figure 4.7 has the two idempotents λ and 111, while the monoid of Figure 4.6b has just λ, the monoid identity, as an idempotent. As we shall presently, see the plethora of idempotents in Figure 4.5 reflects the fact that that monoid has only trivial subgroups. On the other hand, the monoid of Figure 4.6b is a group. Figure 4.7 is between these two extremes: it has nontrivial subgroups, but it is not a group itself. Figure 4.8 has only two idempotents, and yet has no nontrivial subgroups.

The remainder of this chapter will be concerned with a method for finding the idempotents and the maximal subgroups of a given finite monoid *M*. We begin with an arbitrary element a of *M*, and look at the power series a, a^2, a^3, . . . Since *M* is finite, there can be only finitely many distinct elements in this power series. Let $n(a)$, $q(a)$, and $m(a)$ be positive integers defined as follows : $n(a)$ is the smallest positive integer such that, for some $y < n(a)$, $a^{n(a)} = a^y$; $q(a) = n(a) - y$; $m(a)$ is that multiple of $q(a)$ such that $y \leq m(a) \leq n(a) - 1$. These three integers determine powers of a that play an important role in determining the maximal subgroups of a monoid. Let $C_a = \{a^{n(a) - q(a)}, a^{n(a) - q(a) + 1}, \ldots, a^{n(a) - 1}\}$.

Lemma. For any positive integer x, $a^{n(a) + x} = a^{n(a) - q(a) + x}$.

Proof: $a^{n(a) + x} = a^{n(a)} a^x = a^{n(a) - q(a)} a^x = a^{n(a) - q(a) + x}$.

Corollary. For any positive integer x, there is a y where $n(a) - q(a) \leq y \leq n(a) - 1$, such that $a^{n(a) + x} = a^y$.

Theorem 4.6. In a finite monoid, $a^{m(a)}$ is an idempotent and the only idempotent among the positive powers of a.

Proof: By the lemma and since $m(a)$ is a multiple of $q(a)$, $a^{m(a)} = a^{m(a) + q(a)} = a^{m(a) + 2q(a)} = \ldots = a^{2m(a)}$, which shows that $a^{m(a)}$ is an idempotent.

Suppose now that a^x is an idempotent; in other words, that $a^{2x} = a^x$. Then $a^x = (a^x)^{m(a)} = (a^{m(a)})^x = a^{m(a)}$.

Theorem 4.7. In a finite monoid, C_a is a cyclic subgroup of order $q(a)$ with identity $a^{m(a)}$ and generated by $a^{m(a) + 1}$.

Proof: Note that $a^{m(a) + 1} a^{m(a) + 1} = a^{m(a) + 2}$. Indeed, for any i, $(a^{m(a) + 1})^i = a^{m(a) + i}$; also, if $m(a) + i > n(a) - 1$, then $a^{m(a) + 1} = a^{m(a) + 1 - q(a)}$. Thus, the set generated by $a^{m(a) + 1}$ is identical to C_a. In particular, $(a^{m(a) + 1})^{q(a)} = a^{m(a)}$, and $(a^{m(a) + 1})^{q(a) + 1} = a^{m(a) + 1}$; and so, by elementary group theory, it is a cyclic group with identity $a^{m(a)}$. No two distinct powers in C_a can be identical since $a^{n(a)}$ is the first power of a equal to a smaller power. The order of C_a is thus $q(a)$.

Theorem 4.8. In a finite monoid, any power a^x of a is in a subgroup of the monoid if and only if $x \geq n(a) - q(a)$.

Proof: Theorem 4.7 gives us one-half of Theorem 4.8, and so let us assume that $x < n(a) - q(a)$. By elementary group theory, if a^x were in a finite group, then a^x would generate a cyclic subgroup of that group and, for some $y > 1$, $a^x = (a^x)^y = a^{xy}$. Since $a^{n(a)}$ is the smallest power of a equal to a smaller power, $xy \geq n(a)$. By the corollary to the lemma, there must be a z, $n(a) - q(a) \leq z \leq n(a) - 1$, such that $a^{xy} = a^z$. $a^x = a^z$, $x < z < n(a)$, and thus the defining property of $n(a)$ is contradicted.

The significance of Theorem 4.8 is that, once we have determined $n(a)$ and $q(a)$, we have determined how many distinct powers of a there are, and exactly which are in subgroups and which are not.

Theorem 4.9. The set-theoretic union of all cyclic subgroups around an idempotent of a monoid is a group and contains every subgroup around that idempotent as a subgroup.

Proof: Let G_u be the closure under the product operation of the set-theoretic union of all cyclic subgroups around the idempotent u. Thus G_u as defined is a submonoid; and for any element $a \in G_u$, $a = c_1c_2 \ldots c_n$ where the c's are elements of respective cyclic subgroups around u. Then $a^2 = c_1c_2 \ldots c_nc_1c_2 \ldots c_n$ is also in G_u, as are all powers of a, of which there are finitely many. For each I there is a $c_i^{-1} \in G_u$ such that $c_i^{-1}c_i = c_ic_i^{-1} = u$. Now $b = c_n^{-1} \ldots c_1^{-1}$ is in G_u, and all powers of b of which there are finitely many. Clearly $ab = ba = u$. The set consisting of u, all powers of a and all powers of b is thus a finite group with u as identity, since it clearly satisfies all the axioms; this group must be identical to its cyclic subgroup generated by a. And so we have proved that every element of G_u is in a cyclic subgroup with u as identity. Hence G_u is the set-theoretic union itself. That G_u is a group is easily verified.

Now consider an arbitrary subgroup G around u. For every $a \in G$, the cyclic subgroup of G generated by a has u as its identity and must be a subgroup also of G_u. But since G is the union of such cyclic subgroups, G is a subgroup of G_u.

Theorem 4.10. Subgroups around distinct idempotents of a monoid are non-overlapping.

We should note how absurd it would be for an element a of a finite monoid to be a member of both groups around distinct idempotents u_1 and u_2: u_1 and u_2 would both have to be powers of a, contradicting Theorem 4.6.

We now describe a computation procedure for obtaining all the maximal subgroups of a finite monoid, justified by Theorem 4.6-4.10. Start with an arbitrary member a of the monoid. Determine $n(a)$, $q(a)$, and $m(a)$; this yields the information that $a^{n(a) - q(a)}, \dots, a^{n(a) - 1}$ are distinct elements in the maximal subgroup around the idempotent $a^{m(a)}$, and that $a, a^2, \dots, a^{n(a) - q(a) - 1}$ are not in any subgroups at all.

Next, take an arbitrary element b not among the powers of a and do the same for b, finding a cyclic group around an idempotent among the powers of b. This idempotent may or may not be identical to the previously discovered idempotent, and if so it may be that some, all, or none of the other elements of the cyclic subgroup are among those of the previous cyclic subgroup.

We continue this way until all elements of the monoid are exhausted, keeping track of which maximal group, if any, each element belongs to. The result is the division of the monoid into its maximal subgroups, and the miscellaneous collection of elements not in any subgroup.

Applying this procedure to the monoid of Figure 4.7, taking $a = 0$, we get the power series $0, 0^2, 0^3 = 0$; thus, $n(0) = 3$, $q(0) = 2$; the idempotent is $0^2 = \lambda$, and the cyclic group is $\{0, \lambda\}$ around λ. Then taking $b = 1$, the power series is $1, 1^2, 1^3, 1^4 = 1$; and $n(1) = 4$, $q(1) = 3$. The cyclic group is {1, 11, 111) around the idempotent 111. This exhausts the monoid; it consists of two maximal subgroups, which are themselves cyclic groups, and no elements outside of these.

Note that if our procedure is applied to Figure 4.6*b* the same idempotent λ would turn up each time. There are four distinct cyclic groups: $\{\lambda, 0\}$, $\{\lambda, 1\}$, $\{\lambda, m\}$, and $\{\lambda, 01, 01\}$. No elements are outside the one maximal subgroup around λ, which means that the monoid as a whole is a group. Figure 4.6b is the symmetric permutation group on 3 objects.

In applying this procedure to the monoid of Figure 4.5, we might start with 0, getting $0, 0^2, 0^3 = 0^2$ with $n(0) = 3$, $q(0) = 1$. Thus 0 is not a member of any subgroup, and we have the trivial subgroup of the idempotent 0^2. Similarly, 1 is not part of ant subgroup, but it generates the trivial subgroup around 11. Continuing, we find that each of the remaining elements is an idempotent, and there are no nontrivial subgroups. The following theorem will be very useful.

Theorem 4.11. If a monoid M of order m has only trivial subgroup, then, for all $a \in M$, $a^m = a^{m+1}$.

Proof: As seen above the powers of each a contains an idempotent a^r, and a^{r+1} generates a cyclic subgroup with identity a^{r}. since there are only trivial subgroups, $a^r = a^{r+1}$, whence $a^{r+2} = a \cdot a^r = a^{r+1}$, etc. Since r must be at most m, we have $a^m = a^{m+1}$.

The following are exercise solutions from the book.

1. Prove that if α and β are congruence classes modulo L then $\alpha\beta \subseteq \gamma$, for some congruence class γ. Show, by counterexample, that equality need not hold.
   Ans: Suppose α and β are congruence classes modulo L.
   Therefore, α = {set of words congruent to some W_1 modulo L} ∈ L
   β = {set of words congruent to some W_2 modulo L} ∈ L
   Let γ be the set of all the words like W_1W_2
   ∴ let $\gamma = [W_1W_2]_\equiv$
   Remark: $\alpha\beta \subseteq \gamma$
   Proof: Let $W_1' \equiv W_1$, $W_2' \equiv W_2$, To show $W_1'W_2' \equiv W_1W_2$. Let $U, V \in \sum^*$.
   Let $U W_1' W_2' V \in$ L. Then $(U W_1') W_2' V \in$ L so $(U W_1') W_2 V \in$ L.
   Similarly, $U W_1' (W_2 V) \in$ L $\Leftrightarrow$ $U W_1 W_2 V \in$ L.
   Thus $U W_1'W_2' V \in$ L $\Leftrightarrow$ $U W_1 W_2 V \in$ L so $W_1'W_2' \equiv W_1W_2$.
   Then, $\alpha\beta \subseteq \gamma$.
   Since γ is a congruence class, then also γ ∈ L
   ∴ $\alpha\beta \subseteq \gamma \in$ L
   Counter example for equality:
   Let L = {*a*, *b*}
   Then congruence classes are {λ}, {*a*, *b*} and $\{W \mid |W| \geq 2\}$.
   Let α = {*a*, *b*} and β = {*a*, *b*}
   αβ = {*aa*, *ab*, *ba*, *bb*} which is not equal to $\{W \mid |W| \geq 2\}$.
   ∴ $\alpha\beta \subseteq \gamma$ but $\alpha\beta \neq \gamma$.

2. For any congruence classes α and β modulo some event, let [αβ] be that congruence class such that $\alpha\beta \subseteq [\alpha\beta]$. Prove that, for any congruence classes α, β, abd γ, $[[\alpha\beta]\gamma = [\alpha[\beta\gamma]]$.
   Ans: Let α and β are congruence classes modulo L, and [αβ] be that congruence class such that $\alpha\beta \subseteq [\alpha\beta]$.
   Let $\alpha = [x]$, $\beta = [y]$, $\gamma = [z]$.
   Then $(x \cdot y) \cdot z \in ([x][y]) \cdot [z]$
   But $(x \cdot y) \cdot z = x \cdot (y \cdot z) \in [x] \cdot ([y] \cdot [z])$.
   Since congruence classes are a partition and $([x] \cdot [y]) \cdot [z] \cap [x] \cdot ([y] \cdot [z]) \neq \varphi$, we have $([x] \cdot [y]) \cdot [z] = [x] \cdot ([y] \cdot [z])$.

3. Prove that the congruence class containing λ is the identity of the semigroup of congruence classes (and hence that this semigroup is a monoid).
   Ans: Generally, in the monoid [*u*] times [*v*] = [*uv*] in general, where we know [*uv*] contains the concatenation $[u] \cdot [v]$.
   But then $[\lambda]\,[v] = [\lambda \cdot v] = [v]$, so λ is a left identity, and similarly $[u] \cdot [\lambda] = [u\lambda] = [u]$.

4. 4.4 Let *G* be a state graph and let *G*' be the monoid graph constructed from *G* according to the algorithm of Section 4.3. Prove that *G* is the graph of a monoid if and only if *G* is isomorphic to *G*'.
   Ans: ➔ Trivial. Since *G*' is always a monoid graph (Note: by "monoid graph" we will just mean a graph that results from the monoid construction applied to a min finite automata graphs).

$\Leftarrow$ The monoid construction expands $\{[x]_{RL} : x \in \sum^*\}$ to $\{[x]_{\equiv L} : x \in \sum^*\}$. If this expansion is null (the sets are the same) then $\{[x]_{RL} : x \in \sum^*\}$ already has the requisite algebraic property that $[x]_{RL} \cdot [y]_{RL} = [xy]_{RL}$ and conversely.

5. 4.5 Prove that, in a finite monoid with only trivial subgroups, $ab = e$ only if $a = b = e$. (Thus e must be in any set of generators of such a monoid.)
   Ans: Since M is finite, the sequence $a, a^2, a^3, \ldots$ has a duplicate element. Let $a^i = a^j$ where $i < j$. Then $a^j b^i = a^{j-i} = a^i b^i = e$. But then $<a>$ is a non-trivial subgroup unless $a = e$. So $a = e$ and $b = e$.

6. Let $M = \{e, a, b, c\}$, where e is the monoid identity and a, b, and c are right zeros. Thus:
   $a^2 = ba = ca = a$,
   $ab = b^2 = cb = b$,
   $ac = bc = c^2 = c$.
   Prove that no event has a syntactic monoid isomorphic to M.
   Ans: Note: For any $Syn(L)$ and every $W \in \sum^*$, $[W] \subseteq L$ or $[W] \subseteq \sim L$ because if $W \equiv W'$ then $\lambda W \lambda \in L \Leftrightarrow \lambda W' \lambda \in L$.
   So suppose L is regular event and $Syn(L) = M$, and let $e = [\lambda]$, $a = [W_1]$, $b = [W_2]$, $c = [W_3]$ where these are the only congruence classes.
   Either two of $\{[W_1], [W_2], [W_3]\}$ belong to L, or two belong to ~L. Without loss of generality assume $[W_1] \subseteq L$, $[W_2] \subseteq L$. Now for every $u \in \sum^*$, $[u][W_1] = [W_1] \subseteq L$ and $[u][W_2] = [W_2] \subseteq L$. if $[z] = [\lambda]$ we have $[u][W_1][z] = [W_1] \subseteq L$ and $[u][W_2][z] = [W_2] \subseteq L$ whence $uW_1z \in L \Leftrightarrow uW_2z \in L$. Similarly if $[z] \neq [\lambda]$ we have $[u][W_1][z] = [z]$ and $[u][W_2][z] = [z]$ whence $uW_1z \in L \Leftrightarrow uW_2z \in L$. But then by def $W_1 \equiv W_2$, so $[W_1] = [W_2]$, contradiction. $Syn(L) \cong M$ is not possible.

7. Prove that, for every finite monoid M of order two or more, there is an event L, over some alphabet $\sum$, such that (1) $L \neq \sum^*$, (2) $L \neq \varphi$, and (3) M is homomorphic to $Syn(L)$.
   Ans: We explore finite monoid structure some more.
   Let M be any finite monoid. Let $M_e = \{a \in M \mid \exists\, b\, [ab = e]$ = all its with right inverses in 'e'. Let $a \neq e$. Consider $a, \ldots, a^{h-1}, a^h, \ldots, a^{h+t}$ where $a^h = a^{h+t}$, $t \neq 0$, t least. If $e \neq a \in M_e$ and $ab = e$ then $a^h b^h = e = a^{h+t} b^h = a^t$, so a has a 2-sided inverses. Thus we have shown:
   Theorem 1: Let $M_e = \{a \in M \mid a$ has a 2-sided inverse with respect to $e\}$. Then M_e is a subgroup (If a, b have right inverses, so does ab, so M_e is closed under $\cdot$). Moreover, all elements of M with a right (left) inverse in M with respect to e are actually in M_e.
   So finiteness of the monoid "cancels out" "sideness" of inverses.
   Definition: M_e is the *canonical* subgroup of M.
   Now let $a, b \in \sim M_e$ and suppose $ab = c$ and c has a right inverse d with respect to e. Then $cd = e = (ab)\, d = a\, (bd)$ so a has a right inverse and thus is in M_e by theorem above. Contradiction. Hence $ab \in \sim M_e$, giving us
   Theorem 2: $\sim M_e$ is a subsemigroup of M.
   So let $b \in M_e$, $d \in \sim M_e$. Now b^{-1} exists and is in M_e so if $bd \in M_e$ then $b^{-1}bd = d \in M_e$. Contradiction. And if $db \in M_e$ we again have $d \in M_e$, Thus
   Theorem 3: $M_e \times \sim M_e$ and $\sim M_e \times M_e \subseteq \sim M_e$.
   Solution of exetcise: [L non trivial means $Syn(L)$ non-trivial also.] From the above discussion we partition M into M_e, $\sim M_e$ and show that $h : M_e \Rightarrow e_2$, $\sim M_e \Rightarrow a_2$ where $M_2 =$

$\{e_2, a_2\}$ is really an epimorphism, where $a_2^2 = a_2$. Theorem's 2 and 3 show the homomorphism property. Shown in Figure 4.9.

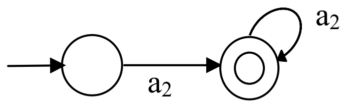

Figure 4.9. Automaton for one letter alphabet.

Since M_2 has a one-letter alphabet and it is a minimal finite automata. We above have trivially that M_2 is a syntactic monoid. This solution shows the alphabet can be kept to one letter.

8. Prove that the syntactic monoid of the event $a^* \cup b^* \cup c^*$ is not isomorphic to the syntactic monoid of any event over a binary alphabet.
   Ans: Let M be as shown in Figure 4.10.

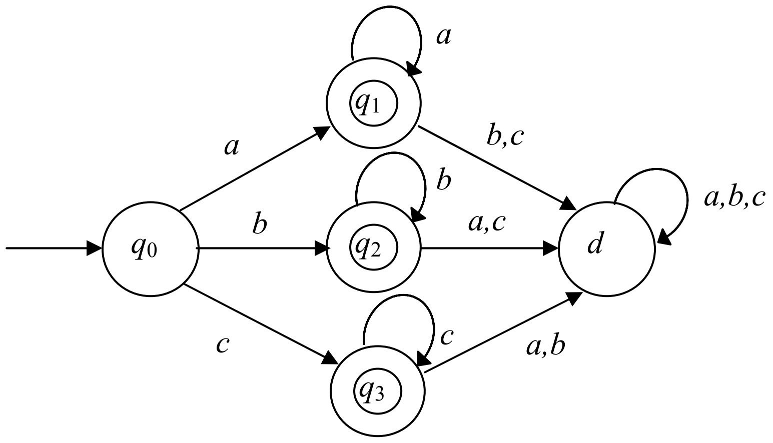

Figure 4.10. State graph for $a^* \cup b^* \cup c^*$.

This is clearly minimal. Now we have *Syn*(L) as shown in Figure 4.11.

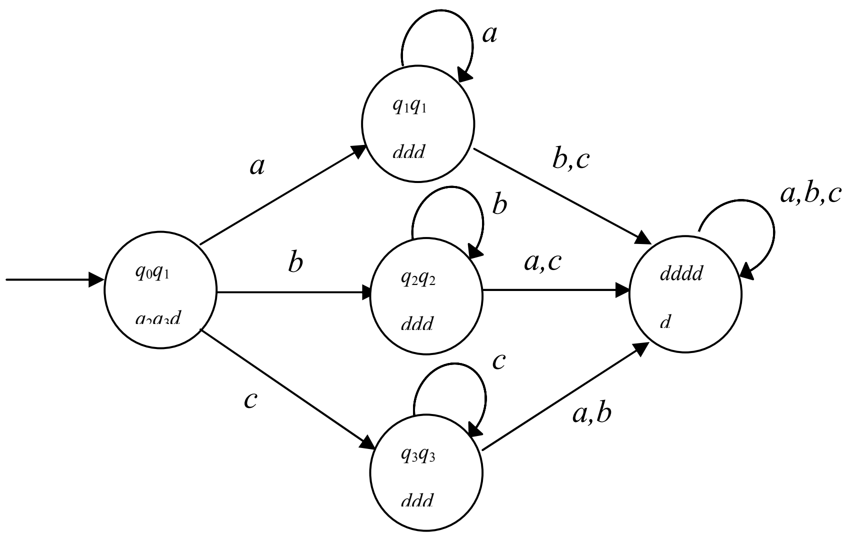

Figure 4.11. Syntactic monoid for *a** ∪ *b** ∪ *c**.

So the monoid graph is simply *M*' would be as shown in Figure 4.12.

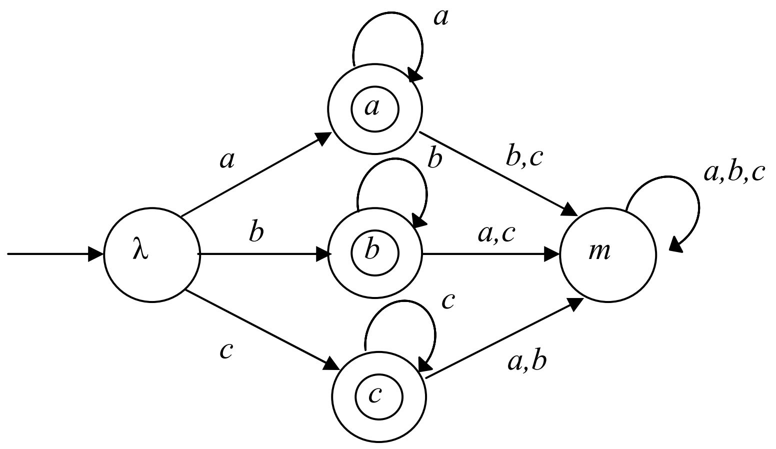

Figure 4.12. Monoid graph for *a** ∪ *b** ∪ *c**.

We observe that *M*' = monoid graph has 3 generators and cannot have two (try any pair). Now if L ⊆ $\{a,b\}^*$; clearly *Syn*(L) has 2 generators, namely [*a*] and [*b*].

9. An event L is *Abelian* if, for all words *V*, *W*, *X*, and *Y*, *VWXY* ∈ L if and only if *VXWY* ∈ L. Prove that an event is Abelian if and only if its syntactic monoid is Abelian.
Ans: ➔ Let L be *Abelian*. Then for all words *W*, *X* we have for all *V*, *Y* *VWXY* ∈ L ⇔ *VXWY* ∈ L, whence by definition of congruence classes we have [*X*][*W*], etc., denotes monoid multiplication.

🡐 If *Syn*(L) is *Abelian*. Then for all words W, X we have $[W][X] = [X]\ [W]$, whence $[WX] = [XW]$, whence for all $V, Y\ [VXWY \in \mathrm{L} \Leftrightarrow VWXY \in \mathrm{L}]$.

10. (Advanced and difficult; suggested for those who have read Chapter II of Clifford and Preston 1961, or the equivalent.) Prove that if a finite monoid M has a group G' as a homomorphic image then a subgroup of M has G' as a homomorphic image. (For the solution see Ginzburg 1968, p. 36.)
Ans: Let $\mathrm{E} = \{a \in M \mid a^2 = a\}$, which is a set of idempotents choose $e_0 \in \mathrm{E}$. such that $|Me_0|$ is minimal (because e_0 could be a right zero, $|Me_0|$ could be even 1).
Now let $G' = e_0me_0$. Then $e_0m_1e_0e_0m_2e_0 = e_0\ (m_1e_0m_2)\ e_0$. So G' is a semigroup and e_0 is a 2-sided identity trivially of G'. Suppose f is "another" idempotent in G', *i.e.* in $G' \cap \mathrm{E}$. Let $f = e_0m_1e_0$, and note that $Mf = me_0m_1e_0 \subseteq Me$. So by choice of e_0, $Mf = Me_0$. Then $e_0 \in Me_0 = me$ 🡒 $e_0 = m_2f$. Whence $e_0 = m_2f = m_2ff = e_0f = e_0e_0m_1e_0 = e_0m_1e_0 = f$. Hence G' has unique idempotent e_0.
By definition of Group, for every m, there exist n such that $(eme)^n$ is idempotent, *i.e.* $(eme)^n = e$ or $(eme)(eme)^{n-1} = e$ establishing an inverse, so G' is a group. Finally $h(G') = h(e_0Me_0) = e_G\ h(m)\ e_G = h(m) = G$.

11. (Elementary) Show that finiteness is necessary in Exercise 18 (in book) by exhibiting an infinite monoid homomorphic to a group G, but such that no subgroup of M can be mapped homomorphically into G.
(Slightly more advanced) Show that every semigroup is the homomorphic image of a semigroup without idempotents.
Ans:
Let S_2 be arbitrary and let M be any idempotent-free semigroup, e.g. $M = \{2, 4, 6, 8, \ldots\}$ under +. Let $S_1 = M \times S_2$ with the "usual operation". Define h *log* $h((m, s_2)) = s_2$. Then H is clearly epimorphic homomorphism S_1 to S_2, and S_1 has no idempotents since M does not.
Let S_1, M, S_2 be as in (*b*) above, and let S_2 be similarly $\{e, a\}$ where $a^2 = e$ so S_2 is a group. Let M' be $\{e\} \cup S_1$ whereas before $S_1 = M \times \{e, a\}$. Make M'_a monoid via $(e')^2 = e'$ and $e's_1 = s_1\ e'$ for all $s_1 \in S_1$. Now define $h' : M' \rightarrow S_2$ by $e' \rightarrow e$ and $(m, s_2) \rightarrow s_2$. Then h' is an epimorphism. But M' has a unique subgroup $\{e'\}$ since this is the only idempotent and the only invertible element. And $\{e'\}$ cannot map onto $\{e, a\}$.
Note: It is clear homomorphism the $a, a^2, a^3, \ldots$ technique in this chapter that finite semigroup has an idempotent(though it might not have an identity). Thus in 19 (b) problem (in book), M must be infinite.

12. (Elementary) Let $M = \{e, a, f, b, c\}$ be the monoid with identity e in which $a^2 = e$, $ab = ba = b$, $b^2 = c$, $b^3 = f$, $b^4 = b$. Prove that M has a subgroup onto which M cannot be mapped homomorphically (thus spoiling the hopes for a converse to exercise 18).
Ans: The multiplication table is shown in Figure 4.13.

	e	*a*	*f*	*b*	*c*
e	*e*	*a*	*f*	*b*	*c*
a	*a*	*e*	*f*	*b*	*c*
f	*f*	*f*	*f*	*b*	*c*
b	*b*	*b*	*b*	*c*	*f*
c	*c*	*c*	*c*	*f*	*b*

Figure 4.13. Multiplication table.

We assume associativity.

The two idempotents are $\{e, f\}$ and any subgroup must contain precisely one of them. The choices are $\{a, e\}$, and $\{b, b^2, b^3\} = \{b, c, f\}$. The map $a \rightarrow f$, $e \rightarrow f$, $f \rightarrow f$, $b \rightarrow b$, $c \rightarrow c$ seems to be an epimorphism onto $\{b, c, f\}$. So we try $\{e, a\}$. If $h : M \rightarrow \{a, e\}$ is epimorphism, we must have $h(f) = h(e) = e$ since h preserves idempotents. For "order" reasons we need $h(a) = a$. But then $e = h(f) = h(af) = h(a)\ h(f) = ae = a$, so h is not homomorphic.

Note: a sledge hammer approach to this issue is to let $M = A_5$, a simple group. Take any non-trivial subgroup H of A_5. If $h : M \rightarrow H$ is epimorphism and homomorphism then the kernel of h is a non-trivial normal subgroup, a contradiction. So such an h cannot exist.

Problems 21 through 26 involve left, right, and 2-sided zeros, which we now define. The problems explore the relationships between such zeros, and "inescapable" states.

Definition: Let M be any monoid. Then $z \in M$ is a left (right) zero. If for all x, $[zx = z]$ (For all x, $[xz = z]$).

And z is a zero if it is both a left and right zeros.

Remark: A monoid has at most one zero.

Proof: If Z and Z' are zeros then $Z = ZZ'$, $ZZ' = Z'$ where Z is left zero and Z' is right zero.

Note: If M has a zero, it has no other left zero or right zero.

We know what a dead state is we extended by

Definition: A state q of a finite automata is immortal if $q \in F$ and $\delta\ (q, a) = q$ for all $a \in \sum$. A state is inescapable (we would say "sink") if it is either dead or immortal.

We recall Theorem 4.5 Let L be fad. Then $W \equiv_L W' \Leftrightarrow$ for all q in L is minimal finite automata $\delta\ (q, W) = \delta(q, W')$.

13. A dead state in a state graph is a nonterminal state such that there is no path from that state to any *other* state. A left zero in a monoid M is an element z such that, for all $x \in M$, $zx = z$. Where ψ is the homomorphism to *Syn*(L), prove that if *Syn*(L) has a left zero z such that $\psi^{-1}(z) \cap \mathrm{L} = \varphi$ then the reduced state graph of L has a dead state.

Ans: Let $a \in \sum$. Since $[z][a] = [z]$ for all a, i.e $\psi(za) = \psi(z)$ for all a, by theorem 4.5 $\delta(q_0, z) = \delta(q_0, za)$ for all a. Let $\delta(q_0, z) = p$. Then $\delta(p, a) = p$ for all a, so p is inescapable and $z \in \sim$L so p is dead.

We now consider an example

E.g. 1

Let M be as shown in Figure 4.14.

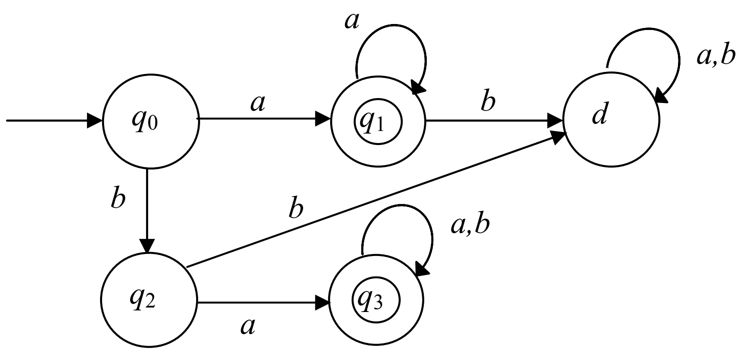

Figure 4.14. State graph for M.

Now M has a dead state and an immortal state. And L(M) = $a^+ + ba\sum^*$, and M seems minimal. Consider $[ab]_{\equiv L}$ and any $x \in \sum^*$. Then $\delta(q_0, ab) = d = \delta(q_0, abx)$, $\delta(q_1, ab) = d = \delta(q_1, abx)$, $\delta(q_2, ab) = q_3 = \delta(q_3, abx)$ and same for d and q_3. Let $z = ab$. Hence $[z]_{\equiv L}$ $[x]_{\equiv L} = [z]_{\equiv L}$ by theorem 4.5 so $[ab]$ is a left zero. Consider $[ba]_{\equiv L}$ and any $x \in \sum^*$. We see similarly that for all q and $x \in \sum^*$ $\delta(q, ba) = \delta(q, bax)$ so $[ba]$ is a left zero also and $[ab] \neq [ba]$ since $ab \in \sim$L and $ba \in$ L. Now $\delta(q_0, ab) \in \sim F$ and $\delta(q_0, ab) \in F$ so $[ab]$ is not a right zero and $\delta(q_0, aba) \in \sim F$ so ba is not a right zero. Indeed it should be verifiable that Syn(L) has no zeros.

14. Let ψ, L, M be as in 21. Let $[z]_{\equiv L}$ be left zero of Syn(L) where $[z] \subseteq$ L. Then M has an immortal state.

Ans: This is just another version of 21 and is proven the same way.

Corollary: If Syn(L) has a left zero $[z]$, then the minimal finite automata has an inescapable state.

E.g 2. Let M be as shown in Figure 4.15.

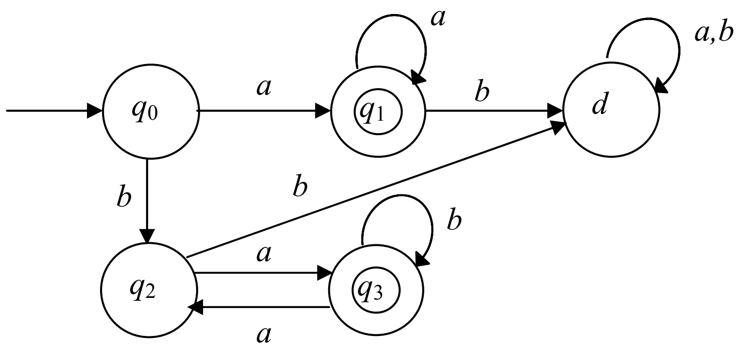

Figure 4.15. State graph for M.

This variant of E.g 1 has a dead state, no immortal state and no left zero in *Syn*(L) as can be verified. This solves Exercises 4.22.

15. Show, by counterexample, that the converse of Exercise 21 (in book) is false.
Ans: See above example.

16. Call a terminal state an "immortal state" if there is no path from it to any other state. State a necessary and sufficient condition (using Exercise 21) on the reduced state graph for L that *Syn*(L) have a left zero. (Note that a reduced state graph can have at most one dead state and at most one immortal state.)
Theorem: Let L be fad, M minimal finite automata for L. Then, *Syn*(L) has a left zero ⇔ (1) M has an inescapable state and (2) there exists z, for all q $\delta(q, z)$ is inescapable (Clearly (2) ➔ (1)).
Ans: All parts are now easy from the preceding discussion and Theorem 4.5.

17. Let G be a state graph (not necessarily reduced) for an event L. Prove that if from every state of G' there is a path to a dead state then *Syn*(L) has a (two-sided) zero z. That is, prove that $xz = zx = z$ for all $x \in$ *Syn*(L).
Now for converse. Let L be fad.
Theorem: If *Syn*(L) has a zero then for every $p \in$ Q such that p is inescapable and is accessible from every state.
Ans: Let z be a zero of *Syn*(L). Then $zx = z = yz$ for all x, y whence for all x, y $[xzy] = [x][z][y] = [z]$. Let M = minimal finite automata for L. We know for all $p \in Q$ such that p is inescapable. Without loss of generality let $[z] \subseteq$ ~L so by Exercise 4.21 $\delta(q_0, ab) = p$ is dead. Now $xzy \equiv_L z$ ➔ $xzy\ R_L\ z$ and so $\delta(q_0, xzy) = p$ for all x,y. so any word containing z goes from q_0 to p. Let q be any state let $\delta(q_0, u) = q$. Then $\delta(q, zy) = \delta(q_0, uzy) = p$ so p is accessible from any q.

CHAPTER 5

THE ALGEBRAIC CHARACTERIZATION OF NONCOUNTING EVENTS

By the definition from chapter 1, a regular event L is in *NC* if and only if, for some positive integer *n* and for all words *V*, *W*, and *X*, $VW^nX \in$ L if and only if $VW^{n+x}X \in$ L, for all positive integers *x*. In this chapter, we will use a more convenient form of this definition: a regular event L is in *NC* if and only if, for some positive integer *n* and for all words *V*, *W*, and *X*, $VW^nX \in$ L if and only if $VW^{n+1}X \in$ L.

We make precise the definition of *GF*: an event L is in *GF* if and only if *Syn*(L) has only trivial subgroups. A third definition is that of *PF* (permutation free): an event L is in *PF* if and only if in the reduced state graph for L there is no word *W* that makes a nontrivial permutation of any subset of the set of states (so we consider only subsets of 2 or more states). A *trivial permutation* is simply the identity permutation. The abbreviations *GF* and *PF* are meant to suggest "group free" and "permutation free," respectively.

As an example of an event disqualified for membership in *PF*, consider the event given be the reduced state graph of Figure 5.1. The word 01 induces a nontrivial permutation on the subset $\{p, r\}$, and so this event is not in *PF*. It is to be noted that in this example, no shorter word (*i.e.*, none of the words λ, 1, or 0) induces such a permutation; furthermore, there is no word that nontrivially permutes the entire set of states. Taking $V = X = \lambda$ and $W =$ 01, for any *n*, one, but not both, of $(01)^n$ and $(01)^{n+1}$ is in the event, which shows the event is also not in *NC*. The homomorphic images of 01 and 0101 in the syntactic monoid constitute a

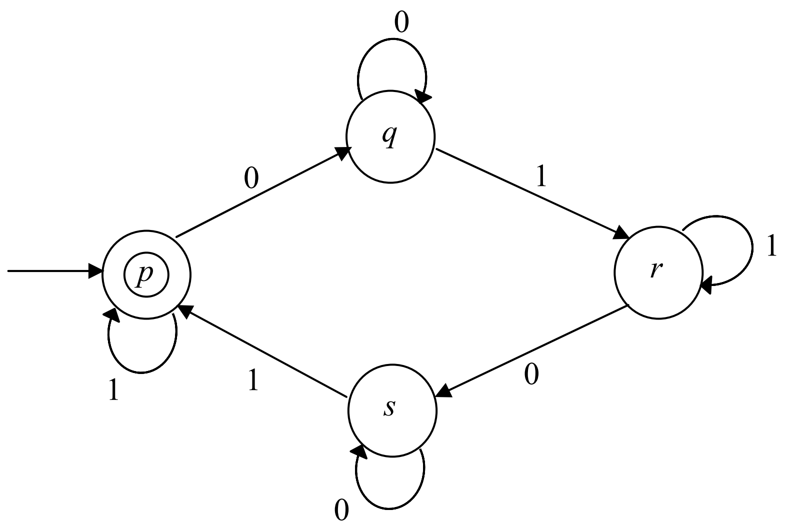

Figure 5.1. State graph for $(01)^{2n}$.

nontrivial subgroup. Thus, the word 01 can be used to disqualify the event from *GF*, from *NC*, and from *GF*. These three concepts are closely interrelated, as the proof of the following theorem shows.

Theorem 5.1 (Main theorem). NC = GF= PF.

The proof is complete with the proof of three lemmas.

Lemma 1. NC $\subseteq$ PF.

Proof: Assume L is regular but L $\notin$ *PF*. Then for some set $\{s_1, \ldots, s_m\}$, $m \geq 2$, of states of the graph G for L , there is a word W such that $G(s_i, W) = s_{i+1}$, for $1 \leq i \leq m - 1$, and $G(s_m, W) = s_1$. Let V_0 be a word such that $G(V_0) = s_1$. Then by definition of "reduced state graph" there is a word X such that one, but not both, of $G(s_1, X)$ and $G(s_2, X)$ is a terminal state. Let $n \geq 0$ be arbitrary. Select j so that $n + j$ is a multiple of m. Then $G(V_0W^jW^n) = s_1$ and $G(V_0W^jW^{n+1}) = s_2$. Therefore, talking $V = V_0W^j$, one, but not both, of VW^nX and $VW^{n+1}X$ is in L. This argument shows that, for each n. such V, W and X exist; hence L $\notin$ *NC*.

Lemma 2. *PF* $\subseteq$ *GF*.

Proof: Suppose L is regular but L $\notin$ *GF*. Then there is an element a in the syntactic monoid such that $\{a, a^2, \ldots, a^q\}$ is a cyclic subgroup of order q around the idempotent a^q, where $q \geq 2$. Let W be a word that maps to a under the homomorphism to the syntactic monoid. Then $W^q \equiv W^{2q}$ (*mod* L), by Theorem 4.1. But $W^q \neq W^{q+1}$ (*mod* L) and so, by Theorem 4.5, there must be a state s in the reduced state graph G such that $s_1 = G(s, W^q) \neq G(s, W^{q+1})$. Then $G(s_1, W) = G(s, W^{q+1}) \neq s_1$, but $G(s_1, W^q) = G(s, W^{2q}) = G(s, W^q) = s_1$. The word W permutes nontrivially the set $\{s_1, G(s_1, W), \ldots, G(s_1, W^{q-1})\}$, and so L $\notin$ *PF.*

Lemma 3. *GF* $\subseteq$ *NC*.

Proof: Suppose that L is regular but L $\notin$ *NC*, and let ψ be the homomorphism to *Syn*(L). Let n be the least common multiple of all the positive integer $m(a)$, where a ranges over all elements of *Syn*(L), and $M(a)$ is a power of a that is an idempotent as in chapter 4. Then, by Theorem 4.6, for every $a \in$ *Syn*(L), a^n is an idempotent.

Since L $\notin$ *NC*, there exists words V, W, and X such that one but not both of VW^nX if and $VW^{n+1}X$ is in L. Let $\psi(W) = a$. Then $a^{n+1} \neq a^n$, by Theorem 4.1, since $W^{n+1} \neq W^n$ (*mod* L). The element a^{n+1} generates the cyclic subgroup $\{ a^{n+1}, a^{n+2}, \ldots, a^{2n}\}$ around the idempotent $a^{2n} = a^n$, since $(a^{n+i})\, a^{n+1} = a^{n+(i+1)}$. Since $a^{n+1} \neq a^{2n}$, this subgroup is not trivial and so L $\notin$ *GF.*

The statements of the three lemmas all involve the idea of being free of nontrivial groups, nontrivial permutation and nontrivial counting operations. But the proofs themselves consists of proving the logical contrapositives, by establishing the existence of nontrivial groups, permutations, or counting contexts. Although this thesis (and the McNaughton and Pappert monograph) is concerned with events from which these things are absent, the reader with broader interests should note that these proofs offer constructive ways of obtaining cyclic subgroups, counting contexts for the first lemma shows how to find an integer n and words V, W, and X such that one, but not both of VW^nX if and $VW^{n+1}X$ is in L, from a reduced state graph in which the word W is a nontrivial permutation of a subset of the states. The proof of lemma 2 shows how to find the nontrivial permutation from the nontrivial subgroup. And the proof of the Lemma 3 shows how to find the nontrivial subgroup from a counting context.

These proofs also show that, for an event L not in *NC*, *GF*, or *PF*, these exists a single word W that disqualifies it from all three: W provides a counting context, it permutes some set of states of the reduced state graph nontrivially, and a power of the image of W in the syntactic monoid is an element of a nontrivial subgroup. Thus we have further justification for saying that *NC*, *GF*, and *PF* are intimately related as concepts.

No short proof that $NC \subseteq SF$ was known by McNaughton and Papert, so the complete equivalence of our various language sets, is postponed, as in their monograph.

To test that a regular event L is *NC*, we can find a minimal state machine and see if some word permutes some subset of states non-trivially (PF test).

Definition: A *pseudovariety* of *monoids* is any family m of monoids satisfying (1) m is closed under submonoids, (2) m is closed under homomorphic image, (3) m is closed under direct product, i.e $M_1, M_2 \in m \rightarrow M_1 \times M_2 \in m$.

The following are exercise solutions from the book.

1. Prove that the following are equivalent conditions on a regular event L with alphabet $\sum$:
   The syntactic monoid of L is a group.
   Every word over $\sum$ effects a permutation of the entire reduced state graph for L.
   There is a k such that, for all words U, V, W over $\sum$, $UV^kW \in$ L if and only if $UW \in$ L.
   Ans:
   Syn(L) a group $\rightarrow$ for all k and for all U, V, W $\;UV^kW \in$ L $\Leftrightarrow$ $UW \in$ L.

Proof: If *Syn*(L) is a group then by Lagrange's Theorem the order of $a \in Syn(\mathrm{L})$ divides the order n_0 of the group *Syn*(L). Hence for all $[V]_\equiv \in Syn(\mathrm{L})$ $[V]_\equiv^{n0} = [\lambda]_\equiv$, the unique idempotent of the group.

By the definition of ≡ for all U, W $UV^{n0}W \in \mathrm{L} \Leftrightarrow U\lambda W = UW \in \mathrm{L}$ which is precisely what we are to show . For easiness we let $k = n_0$.

For all k and for all U, V, W [$UV^kW \in \mathrm{L} \Leftrightarrow UW \in \mathrm{L}$] $\rightarrow$ *Syn*(L) is a group.

Proof: Let k be such a magic number. As usual let $V \neq \lambda$. Then for all $V \neq \lambda$, $[V^k]_\equiv = [\lambda]_\equiv$.

But then $[V][V^{k-1}] = [\lambda]$ so $[V]$ has an inverse and is an arbitrary member of *Syn*(L).

Hence *Syn*(L) is a group.

Note: $\delta(q, \lambda) = q$ for all q, so $V = \lambda$ is trivial in both (2) and (3).

(2) $\rightarrow$ (3) Assume each $W \in \sum^*$ "permutes Q" trivially or otherwise. Let $|Q| = n$, Let π_W be the permutation induced by W. Then any cycle in π_W has length $\leq n$. So $n!$ is divisible by every cycle length of every π_W. Hence for all q, W $\delta(q, W^{n!}) = q$. We let $k = n!$. Let U, V, $W \in \sum^*$ and let $\delta(q_0, U) = q$. As observed $\delta(q_0, UV^k) = q$ so $\delta(q_0, UV^kW) = \delta(q_0, UW)$ whence $UV^kW \in \mathrm{L} \Leftrightarrow UW \in \mathrm{L}$.

(3) $\rightarrow$ (2) Let k be magic number, *i.e* $UV^kW \in \mathrm{L} \Leftrightarrow UW \in \mathrm{L}$, for all U, V, W. Let $q \in Q$ be arbitrary and $\delta(q, U) = q$ since q is accessible in the minimal machine. Let V be any non-null word. Then using Nerode Theorem proof, $UV^k R_\mathrm{L} U$ since for all W $UV^kW \in \mathrm{L} \Leftrightarrow UW \in \mathrm{L}$. By proof of Nerode we can think of $[UV^k] = [U]$ as a state, *i.e.* $\delta(q_0, U) = \delta(q_0, UV^k)$. (Or use the "definition" of "reduced finite automata" in the text). So $\delta(q, V^k) = q$ and hence V has a cyclic affect on q. Since q and V are arbitrary, we are done.

2. Prove that the class of finite monoids with only trivial subgroups is a pseudovariety of monoids.

Ans: Let m be class of all finite monoids with only trivial subgroups.

(1) Let $M \in m$, M' a submonoid of M. If M has non-trivial subgroup G, then clearly so does M. So $M' \in m$.

(2) Let $\theta : M \rightarrow M'$ onto, where $M \in m$.

By theorem 4.11 $a^m = a^{m+1}$ for all $a \in M$ where $m = |M|$.

Thus $\theta(a)^m = \theta(a^m) = \theta(a^{m+1}) = \theta(a)^{m+1}$ and this equation is thus true for each $b \in M'$, *i.e.* for all $b \in M'$, $b^m = b^{m+1}$. Now if M' had a nontrivial subgroups $<b>$, *i.e.* $b^k = e$ where e is an idempotent of M', k least such, $k \geq 2$, $(b \neq e)$. But then $b^{k+1} \neq b^k$ for all k, a trivial cyclic group property. Contradiction. So, $M' \in m$.

(3) Let $M_1, M_2 \in m$. Suppose $M_1 \times M_2$ has non-trivial subgroup G with idempotents (identity) (e_1, e_2) (Note: e_1, e_2 need not be identities of M_1, M_2). Then $(e_1, e_2)^2 = (e_1, e_2)$ so e_1, e_2 are idempotents of M_1, M_2.

Let $(a_1, a_2) \in G$, $(a_1, a_2) \neq (e_1, e_2)$. Then for all k such that k minimal and $(a_1, a_2)^k = (e_1, e_2) = (a_1^k, a_2^k)$. Without loss of generality assume $a_1 \neq e_1$. Then $<a_1>$ is a non-trivial subgroup of M_1. Contradiction. So, $M_1 \times M_2 \in m$.

Note: $NC \subseteq \{\mathrm{L} \mid \mathrm{L}$ regular$\}$.

Remark: The main theorem of this chapter says that for L regular, $\mathrm{L} \in NC \Leftrightarrow \mathrm{L} \in GF$ (*i.e.* *Syn*(L) has only trivial subgroups). Thus NC is "determined" by this pseudovariety of monoids.

3. Prove that the class of all finite groups is a psuedovariety of monoids. Show, on the other hand, that the class of all groups (finite and infinite) is not a pseudovariety of monoids.
Ans:
Let G_F be the class of all finite groups. We show G_F is a pseudovariety of monoids.
Let M be a submonoid of finite group $(G, \cdot, e)$ since G has but one idempotent and M has an identity, $e \in M$. Moreover $a \in M$ means all powers are in M, and some power of a is an inverse *i.e.* $aa^{m-1} = e$ for some m, so M has inverse and we are done.
The homomorphic image of a finite group is well known to be a finite group, e.g. $e' = h(e) = h(a)h(\mathrm{L})$ so each $h(a)$ has an inverse in the image.
(3) The direct product of 2 finite groups is also well known to be a finite group.
Let G be class of all groups. Then G is not a pseudovariety of groups.
Proof: Let $G = (Q - \{0\}, \cdot, 1)$, a group. A submonoid is $(z - \{0\}, \cdot, 1)$, which is not a group. The other 2 conditions are satisfied.

4. Let $\mathrm{L}_1 \in NC$, L is arbitrary, and let $\mathrm{L}_2 = \{W \mid \text{for all } X \in \mathrm{L} \text{ such that } XW \in \mathrm{L}_1\}$, i.e. L_2 is suffix as of L, where the prefix is in L, the dual of L_1/L. Show L_2 is NC.
Ans: Let n_0 be the magic number to show L_1 is in NC. Suppose $Z = V\,W^{n0}\,Y \in \mathrm{L}_2$. And let $x \in N^{+}$. Let $U \in \mathrm{L}$ such that $UZ = (UV)\,W^{n0}\,Y \in \mathrm{L}_1$. Then $(UV)\,W^{n0+x}\,Y \in \mathrm{L}$, whence $VW^{n0+x}\,Y \in \mathrm{L}_2$, and "conversely". So L_2 is in NC.
Note: This short proof uses nothing in Chapter 5. Note that fad's are also closed under this operation.
Corollary: Define languages quotient in the usual way. Then the "dual" of the proof above shows that Theorem, The NC languages, like the fad languages, are closed under quotient with an arbitrary language, and thus closed under PREFIX, SUFFIX and SUBSTRING.

5. NC is not closed under ½.
Ans: Let $\mathrm{L} = \{a^n e^2 (bc)^m \mid n, m \geq 1\}$. It is clear $\mathrm{L} \in NC$. Now $NC = SF$ and is closed under intersection, so if L is closed under "half", then $(\tfrac{1}{2}\,\mathrm{L}) \cap a^*e = \{a^n e \mid |a^n e| = |e\,(bc)^m|\}$ is NC. But $|a^n\,e| = |e\,(bc)^m|$ means $n = 2m$ so we really have $\{a^{2n}\,e \mid n \geq 1\}$ which is clearly not in NC. Hence NC is not closed under "half."

CHAPTER 6

NERVE NETS

Nerve nets provide another characterization of noncounting events and also they provide another essential link in the proof that all the characterizations are equivalent.

In Figure 6.1, the triangle is a *neuron* and the integer is the *threshold* of the neuron. The lines leaving the right side are the neuron's *exit axons*; the lines ending at the dots are *incident* axons. In our nets, a neuron has any fixed number of incidents axons, and may have as parts any number of axons.

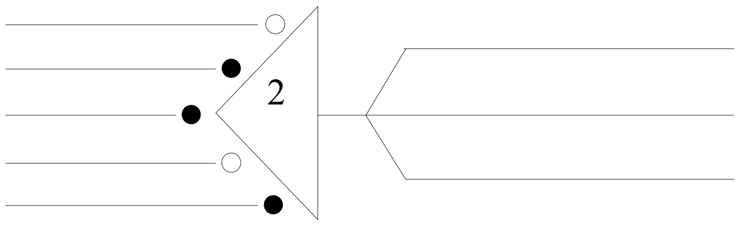

Figure 6.1. Neuron and axons.

There are two types of incident axons; *inhibitory* and *excitory*. In the figures an excitory incidence is indicated by a solid dot, an inhibitory incidence is indicated be a small circle. So, there are three excitory and two inhibitory incident axons in Figure 6.1.

For the sake of conceptual ease, our nerve nets are synchronous. We disregard the continuity of time and assume that it is made up of moments: time 1, time 2, time 3, etc. At any time a neuron either fires or fails to fire, a pulse is transmitted along each of its axons. Whether or not it fires depends on the presence of pulses at each of its incident axons. We stipulate that the neuron fires at any moment if and only if the number of pulses on the excitory incident axons minus the number of pulses on the inhibitory incident axons at that moment equals or exceeds the threshold of the neuron. Thus a neuron computes what is known in switching theory as a threshold function. For example, the neuron of Figure 6.1 will fire if any two of the excitory and none of the inhibitory incident axons have pulses; and it will fire if all three excitory and no more that one of the inhibitory incident axons have pulses; but it will not fire otherwise. Note that neuron with threshold zero does not fire if there are more inhibitory pulses then excitory pulses.

The pulse produced by a neuron travels along every one of its output axons. Each axon of a neuron has associated with it a constant nonnegative integer, its delay time, which is the time it takes a signal to travel along the axon. We assume no computation time for a neuron. Thus it is possible for information to go from one neuron to another in zero time, namely, in cases where the axon has zero delay time. This feature of our nets may seem strange but will prove to be convenient.

Our nets will have in addition to neurons certain axons that are not parts of any neurons. These will be called "input axons," and will serve as net input. Net outputs will be the end points of axons that are not incident to any neurons; these will be called "output axons," and the end points themselves will be called "outputs."

A *nerve net* is defined as a finite set of input axons and neurons: each neuron has as part of it one or more axons; each axon is incident to at most one neuron. Not all nerve nets so defined are useful models of computation. Accordingly, we proceed to introduce an important restriction on the delay times of certain axons.

It is useful to stipulate that there be no instantaneous feedback. A *loop* is a sequence $N_0A_0N_1A_1 \ldots N_{p-1}A_{p-1}N_p$, where each A_i is an axon of the neuron N_i and incident to the neuron

N_{i+1}, and where $N_0 = N_p$ but otherwise $N_i \neq N_j$ *for* $i \neq j$. The stipulation is then that in every such loop one of the axons has at least one delay. A net satisfying this stipulation is a *well-formed net*. The word "net" will henceforth be used to mean "well-formed nerve net". It is to be noted that, although some nets that are not well formed behave in an understandable manner, the class of well-formed nets as a whole is adequate in the following sense: for any net that behaves intelligibly, there is a well-formed net that behaves in exactly the same way. The purpose of the concept of "well-formed net" is that it provides us with a class that is adequate in this sense but has no net whose behavior is unintelligible.

We stipulate that, at time 1 (the initial moment of time) and times following, either a pulse is placed or no pulse is placed at each input axon. As in the usual treatment, the placing of such pulses is arbitrary and independent of the functioning of the net. But this input history and the structure of the net together determine the history of the incidence point of each axon of the net, that is, whether or not that point has a pulse at each moment of time. The incidence point or end point of an axon with delay d has no pulse at times 1 through $d - 1$; the reason is that no input has a pulse before time 1 and no neuron can fire before time 1.

In our figures, a small vertical line in an axon denotes a delay of one. A delay of *d* is denoted by *d* small vertical lines along the axon.

The word "junction" will now be introduced to means a point in the net where a pulse may or may not appear. One might say that there are infinitely many points along an axon all of which may experience a pulse travelling along. However, we shall say that the axon has only finitely many junctions, one at each end of the axon, and one between each successive pair of delays if the axon has two or more delays. Thus an axon with delay zero or delay one has two junctions, whereas an axon of delay two has three junctions, etc. All the axons that are part of a neuron have one junction in common, namely, the initial point of all those axons. This junction will be called the *neural junction*. A net input is a junction; there may be several input axons having that junction as its initial point.

Thus a net has finitely many junctions; at any time each of these may or may not have a pulse. The pulses on the net inputs are arbitrary; but, given them, the pulses on the remaining junctions are mathematically determined.

Each junction of a net represents an event, and we say that a net with one output represents the event represented by that output. If the net has n inputs, the event is over an alphabet of 2^n letters, each letter representing a combination of the presence and absence of pulses on the inputs. This alphabet is called the input alphabet, or $\sum$. If the net has no inputs, then the alphabet $\sum$ is unspecified. At each moment of time the input junctions are, as it were, pulsed with a letter of the input alphabet. Thus, for any time *t*, the net has received an input word from time 1 through time *t*.

We are now in a position to define the event of a junction of a net. To say that a junction represents an event L over the input alphabet is to say that, for all times *t*, the junction has a pulse at time *t* if and only if the input word from time 1 through time *t* is a member of the set L. A consequence of this stipulation is that no junction represents an event containing the null word.

Figure 6.2 is an example of an inputless neuron with threshold zero, whose one axon has a delay of one. Thus junction *A* has no pulse at time 1 but will have a pulse at every succeeding moment of time. If we wish to describe the behavior of the junction *A* as an event, we can say that it represents the event $\sum\sum\sum^*$, where $\sum$ is unspecified.

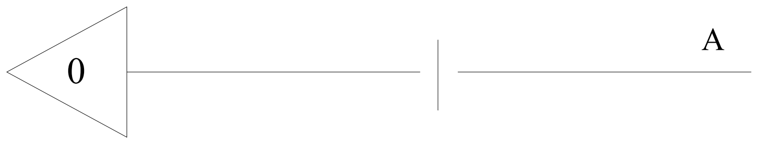

Figure 6.2. Inputless neuron.

In Figure 6.3, *D* receives a pulse at any time *t* if and only if *C* has a pulse at time *t*, *B* has no pulse at time *t*, but B has had a pulse at some time before time *t*. The behavior of *D* can be described as an event over a four-letter alphabet {0, 1, 2, 3} where 0 represents no pulse at either input *B* or *C*, 1 represents a pulse at *B* but none at *C*, 2 represents a pulse at *C* but none at *B*, and 3 represents a pulse at both *B* and *C*. The event of *D* is then $\sum^*(1 \cup 3)$ $\sum^*2$. If we have $|\sum| = 3$ then we need B = 1 and C = 1, which cannot occur. This event is not restricted by length of word.

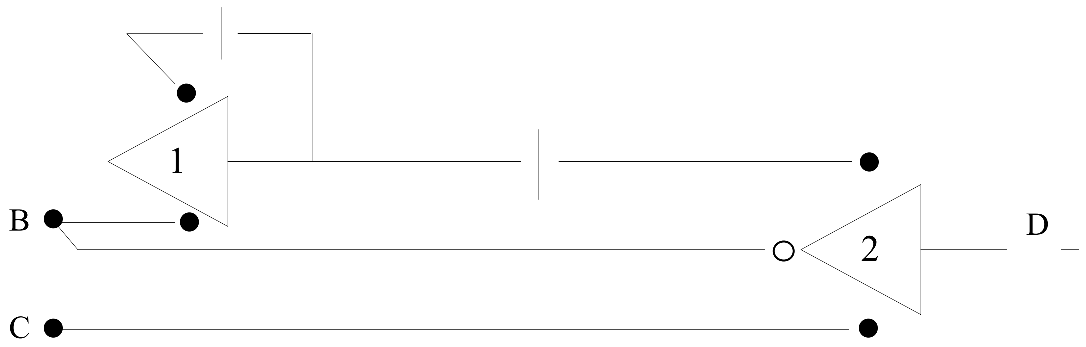

Figure 6.3. Nerve net for $\sum\sum\sum^*$.

Now we can see that all regular events that do not contain λ can be realized by our nets. We will use the state graph to prove this fact rather than the regular expression. Figure 6.4 shows a state graph with initial state, state *A* and state *B*, and Figure 6.5 is a net that realizes Figure 6.4, as will be explained.

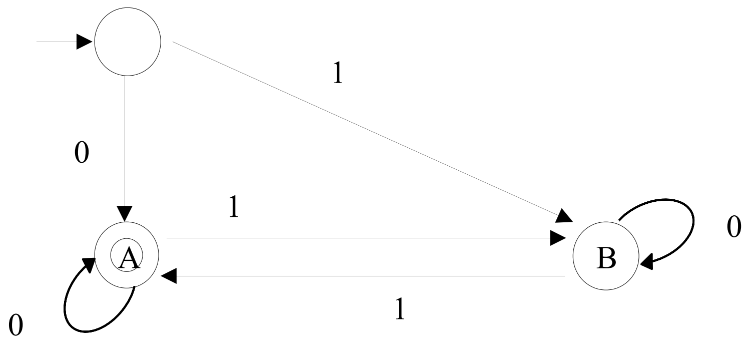

Figure 6.4. Example of an event without λ.

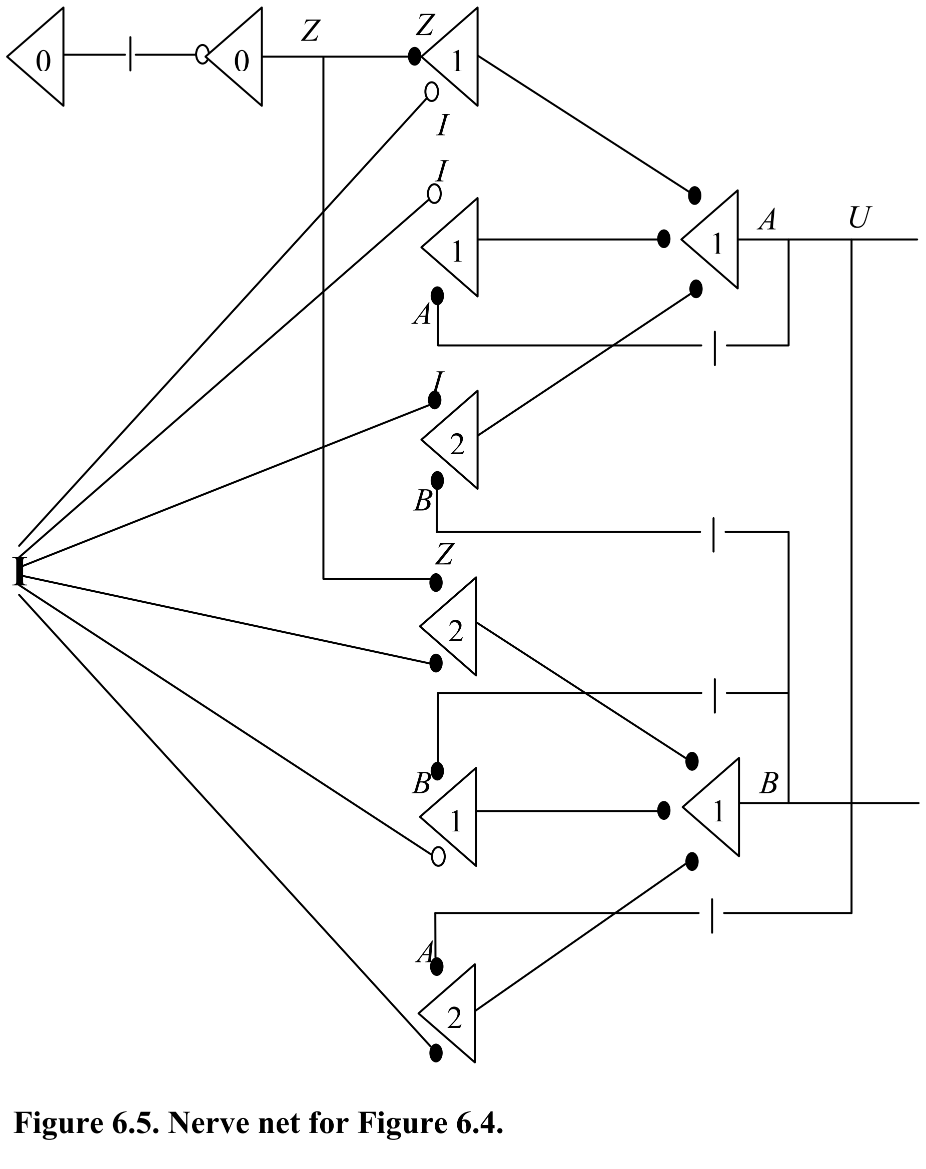

Figure 6.5. Nerve net for Figure 6.4.

It is shown in Figure 6.5 that there is an axon with zero delay from neural junction labeled *Z* and from the input *I* to each incidence point with the same label. Also there is an axon with delay one from the neural junctions labeled *A*' and *B*', respectively.

To see how the net of Figure 6.5 realizes the state graph of Figure 6.4, note how an input word traces out a path through the graph. Let us think of this path as occurring in time; let us say that it is in the initial state before the beginning of time. At time 1, it is at state *A* or *B* depending on the whether the first letter in the word is 0 or 1. At any time $t + 1$, the state the path is in is determined by the state it was in at time *t* and the input at time $t + 1$ (i.e., the ($t + 1$)th letter in the input word). By the nature of the state graph, at no time *t* is the path at the initial state. The correspondence between the state graph and net is affected by stipulating that an input of 1 in the state graph represents a pulse at the input *I* in the net, and 0 represents the absence of a pulse *I*. For any input word, and for any time *t*, one and only one of the two neural junctions *A* and *B* will have a pulse at time *t*; the pulse will be at *A* or *B* according to whether the path is at state *A* or state *B*. There is an axon connecting *A* and *U*, so that if there is a pulse at *A* there will also be a pulse at *U*, which is justified by the fact that *A* is a terminal state of the graph.

Note that the junction *Z* has a pulse at time 1 and none thereafter, thus determining, with the input at time 1, which of *A* or *B* has a pulse at time 1. Thereafter, whether or not *A* or *B* has a pulse is in accord with the transition rule of the state graph.

We proceed now to describe a class of events realizing noncounting events. A net is *buzzer-free* if there exists no loop with an axon that has an inhibitory incidence. The significance of this concept is exemplified in the buzzer net of Figure 6.6, constructed according to a well-known principle.

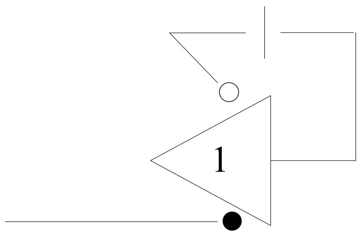

Figure 6.6. Buzzer free net.

As long as the input axon is pulsed, the buzzer is on and the output will have an intermittent pulse. When the input axon has no pulse, the buzzer is off and the output does not receive any pulses. The event represented by this net is $(\lambda \cup F0)\ (11)^{*}1$, which is not a noncounting event. The role of the inhibitor in the loop is what gives the effect of a buzzer, and motivates our term "buzzer-free" for nets without inhibiting feedback axons. (We do not mean imply that all nets that have loops with inhibitors act like buzzers.)

The property of being buzzer-free is not significant in itself. The construction method of Section 6.3 always results in a buzzer-free net, which a glance at Figure 6.5 will verify. Thus, there is a buzzer-free net for every event of *NN*. To get a real distinction we must conjoin this property with another property.

A net is *almost loop-free* if it has no loop that has more than one neuron, or an axon with more than one delay. In other words, $p = 1$ for all loops (if it has any loops at all) and A_0 has a single delay. Figure 6.7 and 6.8 show the significance of this concept. The feedback axon of Figure 6.7 has a delay of 2, and thus the output U is able to count modulo 2. Note that the net has no input. The neuron A has a pulse at time 1 and no pulse thereafter. Since the feedback delay on the rightmost neuron is 0, U will have a pulse at time 1 and at every other moment thereafter. The event represented by this net is $(\sum\sum)^{*}\sum$, which is not a noncounting event.

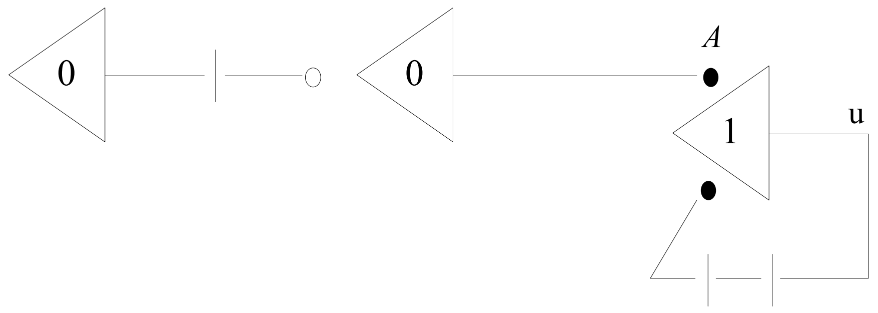

Figure 6.7. Nerve net counts modulo 2.

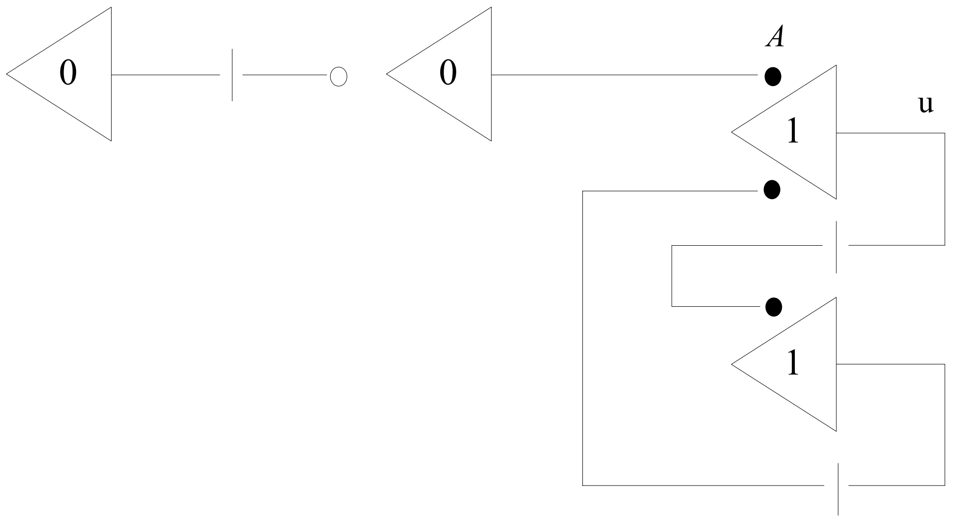

Figure 6.8. Almost loop-free net.

If loops could involve more than one neuron, events outside of *NC* could be represented. An example is Figure 6.8, which is equivalent to Figure 6.7. In an almost loop-free net all feedback occurs by means of axons that are incident to their own neurons. Such will be called *feedback axons*.

The class of nets that we are looking for is the class of nets that are both buzzer-free and almost loop-free. Note that such nets are made up in a series-parallel fashion from neurons of two types: neurons without feedback axons and neurons that have a limited type of feedback. The limitation is that feedback axons are all excitory, have delay 1, and are incident to their own neurons. Neurons with such feedback play the role of what are sometimes known as *RS* flip-flops or set-reset flip-flops: that is, the neuron is either off (not firing) or on (firing). The incident axons, other than the feedback axons are, in effect, the inputs to this *RS* flip-flop. But the feedback axons are regarded as internal to the flip-flop, and thus neither inputs nor outputs. There are three classes of input condition to an *RS* flip-flop: the set, which turns it on if it is off and leaves it on if it is on; the reset, which turns it off if it is on and leaves it off if it is off; and a third that leaves it off if it is off and leaves it on if it is on. There is no fourth type of input condition that both turns it on when it is off and turns it off when it is on (which would permit it to buzz). Thus we can say that, when an *RS* flip-flop changes state, what state it changes to is determined by its input and is independent of its previous state. Figure 6.9 is a state graph for an *RS* flip-flop.

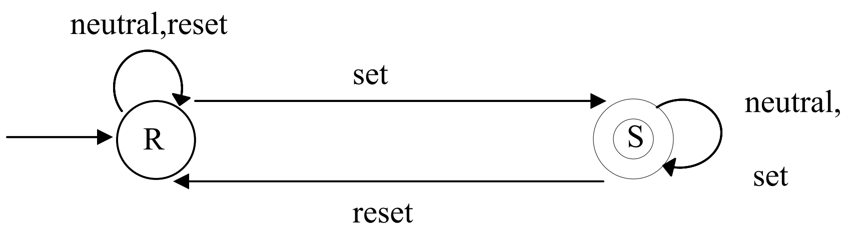

Figure 6.9. State graph for RS flip-flop.

Note that, as a general concept of flip-flops, Figure 6.6 (the buzzer) is a trigger flip-flop or *T* flip-flop, and a delay element is a *D* flip-flop.

A neuron without feedback at all is often called a *combinational* element, since whether or not its axons (which is its output) receives a pulse at any time depends only upon the combination of pulses on the various incidence points (its input) of the neuron at that same moment of time.

We now define an *RS net* to be a net that is buzzer free and almost loop-free. In other terms, it is a net made up in a series-parallel fashion from combinational elements, delays, and *RS* flip-flops. And we define *LF* to be the set of all events represented by *RS* nets. The letters "*LF*" are meant to suggest "loop free," just as *GF* and *SF* suggest "group-free" and "star-free."

A familiar train of thought, made explicit by Kleene, establishes that a net without any loops at all represents a definite event. (Consider the set of all paths in such a net from an input to the output U. Count the number of delays along each such path, and let k be the maximum such number. Clearly then what happens at the inputs at any time cannot have any influence on what happens at U more than k moments later. Hence U realizes a k-definite event.) Such nets can be characterized more broadly by saying that they are made up in a series-parallel fashion from delay elements and combinational switching elements. It is interesting to see that if *RS* flip-flops are also allowed, the nets can realize all noncounting events. This insight (with "group-free" replacing "noncounting") must be credited to Krohn and Rhodes, since it is a corollary of their main theorem.

In the next chapter we shall shows that $LF \subseteq SF$. In preparation for this result we shall conclude the present chapter by proving something about the function performed by neurons functioning as *RS* flip-flops.

We define a *square operator* by $\alpha \square \beta$, for events α and β, as the set of all words having some initial segment in α, with no longer initial segment in β. Thus $\alpha \square \beta$ equals the set of all words W such that, for some V_1 and V_2, $W = V_1V_2$, $V_1 \in \alpha$, and for no V_3 and V_4 is $V_2 = V_3V_4$, $V_3 \neq \lambda$, and $V_1V_3 \in \beta$. In other words, $\alpha \square \beta$ is the set of words W with at least one initial segment in $\alpha \cup \beta$ such that if V_1 is the longest such segment (possibly W itself) the $V_1 \in \alpha$. (if V_1 is also in β, that does not spoil it.) This operation plays a crucial role in describing the behavior of *RS* nets, as the following theorem shows.

Theorem 6.1. Let $\gamma_1, \ldots, \gamma_n$ be the events realized by the incidence points other than those of the feedback axons of a neuron with feedback in an *RS* net. Then there exist Boolean functions β_1 and β_2 such that the neural junction represents the event $\beta_1(\gamma_1, \ldots, \gamma_n) \square \beta_2(\gamma_1, \ldots, \gamma_n)$.

Proof: Suppose the neuron has threshold h and f feedback axons. Let $f(t)$ be the number of pulses at time t at the excitory incidence points of the neuron, other than those of the feedback axons, minus the number of pulses at time t at the inhibitory axons. If the neural junction has a pulse at time t, then it will also have a pulse at time $t + 1$ if and only if $f(t + 1) \geq h - f$. But, if it has no pulse at time t, then it has a pulse at time $t + 1$ if and only if $f(t +1) \geq h$. From this we conclude that, for any time t, there is a pulse at the neural junction at time t if and only if there is a time $t_1 \leq t$ such that $f(t_1) \geq h$, but there is no time t_2, $t_1 + 1 \leq t_2 \leq t$, such that $f(t_1) < h - f$.

Let β_1 and β_2 be, respectively, the Boolean condition on the n incidence points at time t equivalent to $f(t) \geq h$ and $f(t) < h - f$. by definition of the $\square$ operator, the neural junction realizes the events $\beta_1(\gamma_1, \ldots, \gamma_n) \square \beta_2(\gamma_1, \ldots, \gamma_n)$.

The following exercises are not from the text.

1. Show Closure (*LF*, ∩, ∪, ~) = *LF*.
   Ans: Recall: $LF \subseteq NN$
   LF is *RS* i.e. *LF* is loop free and buzzer free.
   Let L_1 and L_2 are loop free and buzzer free languages.
   $\therefore L_1, L_2 \in LF$
   $\therefore \lambda \notin L_1, \lambda \notin L_2$
   So, $\lambda \notin L_1 \cup L_2$. We now do the proof.

$L_1 \cup L_2$: Figure 6.10 represents union.

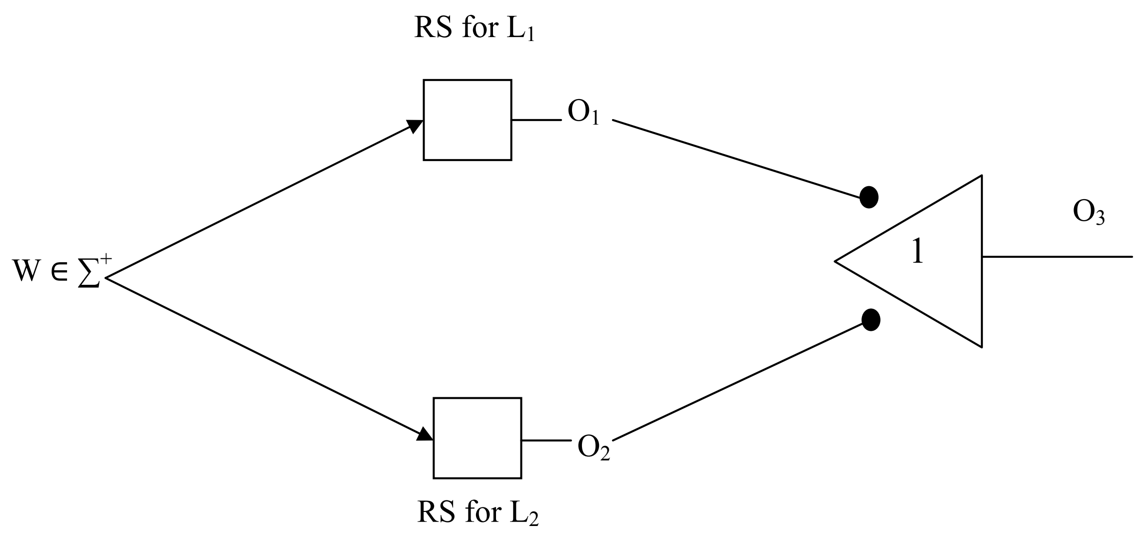

Figure 6.10. Union of L_1 and L_2.

$L_1 \cap L_2$: Figure 6.11 represents intersection.

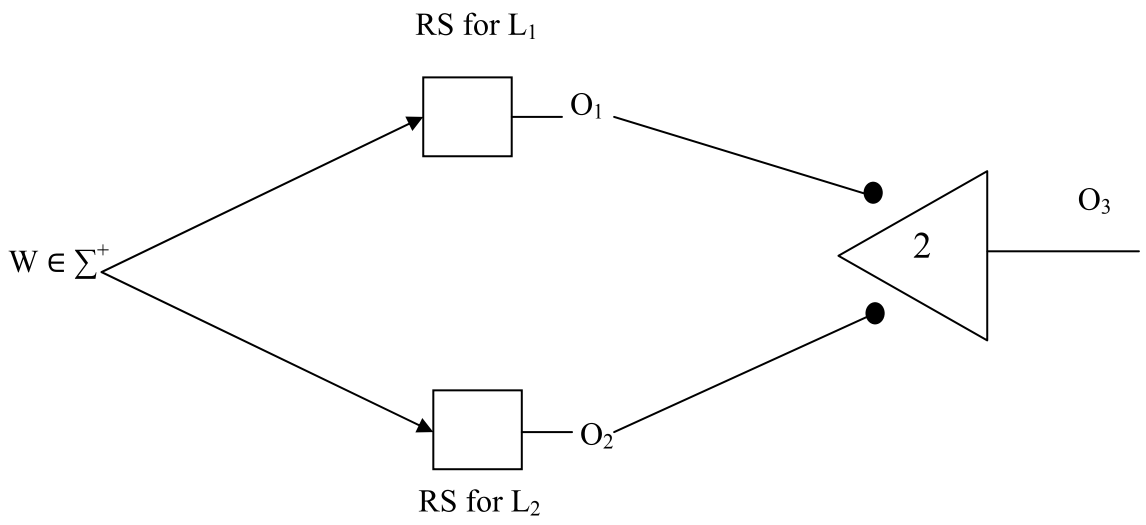

Figure 6.11. Intersection of L_1 and L_2.

$\sim L_1$: Figure 6.12 represents complement.

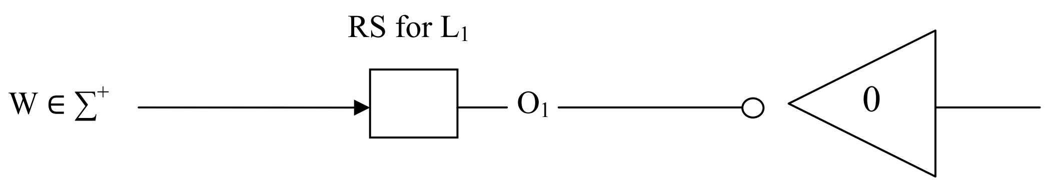

Figure 6.12. Complement of L_1.

2. Find RS nets for the following.
   Ans: $\sum = \{a, b, c\}$ L= $\{b\}$, Inputs are shown in Figure 6.13 and event L is represented by Figure 6.14.

$\sum$	i_1	i_2
a	0	0
b	0	1
c	1	0
bc	1	1

Figure 6.13. Inputs of net.

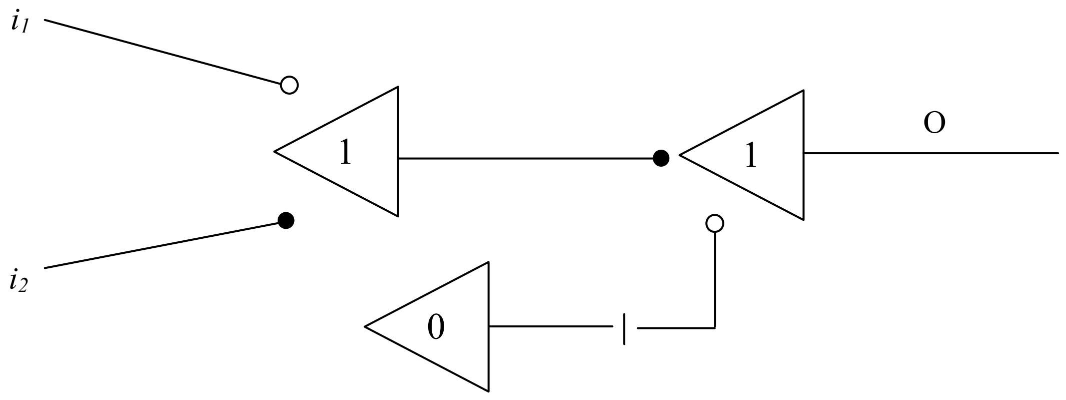

Figure 6.14. Represents output b.

L = $\{bc\}$, Figure 6.15 represents event L.

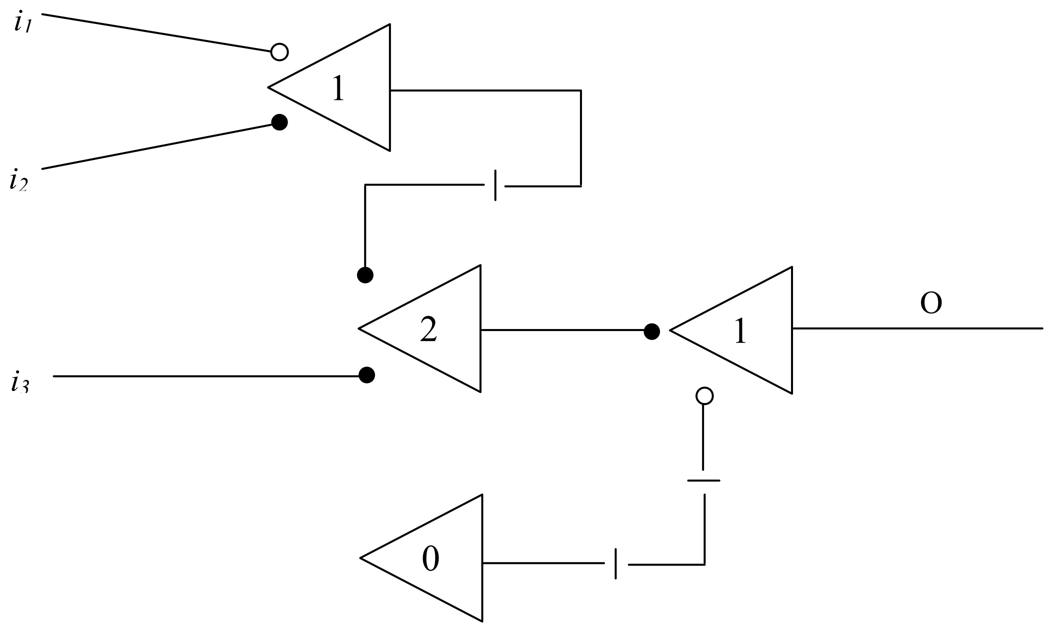

Figure 6.15. Represents output bc.

CHAPTER 7

THE BEHAVIORAL ANALYSIS OF RS NETS

In this chapter $GF \subseteq LF$ as theorem 7.4 is proven. What precedes this theorem establishes a technique of state graph analysis of nets in general, and shows how to modify nets so as to change the initial state and output states, meanwhile keeping the structure of the net essentially the same. It turns out that this what is needed to be able to analyze an *RS* net in terms of star-free expressions.

In order to apply state-graph analysis to nets, a concept of the "state" of a "net state" is introduced. The state of a net at time x is the same as the state at time y if, for every junction J, there is a pulse at J at time x if and only if there is a pulse at time y.

The *initial state* of the net is assumed before time 1, which thereafter the net never assumes.

Theorem 7.1. Let S and S' be two net states such that, for every D-junction D, D has a pulse in S if and only if D has a pulse in S'. Then $S = S'$

Proof: See the book.

In order to prove the main theorem, the concept of net depth of an *RS* net is introduced.

A neuron M directly drives a neuron M' if $M \neq M'$ and an axon of M is incident to M'. A neuron that is not directly driven by any neuron has depth 0 if it has no feedback and has depth 1 if it has feedback. Otherwise, the depth of a neuron M is defined in terms of the maximum depth d of all neurons directly driving M; the depth of M is d if M has no feedback but is $d + 1$ if M has feedback. The depth of an *RS* net is the depth of the neuron of maximal depth in the net.

Theorem 7.2. Let *N* be an *RS* net of depth *n* with a single output and let *S* be a realizable state of *N*. Then there is a net N_S of depth n realizing the event given by the state graph obtained from the structural state graph of *N* by making *S* the initial state.

Proof: See the book.

Theorem 7.3. Let *N* be an *RS* net of depth *n* and let *S* be a realizable state of *N*. Then there is a net N^S of depth *n*, which is like *N* with the same state behavior but which has in addition an output U_S that has a pulse when and only when N^S is in state *S*.

Proof: See the book.

Theorem 7.4. (Main Theorem). LF ⊆ SF.

Proof: See the book.

CHAPTER 8

DECOMPOSITION THEORY FOR NONCOUNTING EVENTS

In this chapter we shall prove that $GF \cup NN \subseteq LF$. As was explained in chapter 1, NN is the class of all regular events L such that $\lambda \notin$ L. Recall from chapter 6 that no net can represent an event containing λ, by virtue of the very stipulation by which nets represent events. But, although we cannot prove $GF \subseteq LF$, it will turn out that proving $GF \cup NN \subseteq LF$ will serve our purpose just as well. The main proofs in this chapter are fairly elaborate and unintuitive, unless you have experience with Krohn-Rhodes. But the presentation in "the text" is very complete, and presented in nearly identical form here.

Our task is thus to show how to build an *RS* net to represent any event without the null word whose syntactic monoid has only trivial subgroups.

We extend in an obvious fashion the concept of "syntactic monoid." By "finite automaton" we mean anything whose input-output behavior is given by a finite state graph. By "the syntactic monoid of a finite automaton" we mean the monoid obtained from the state graph of the automaton by means of the algorithm of Section 4.3. Thus we can talk about the syntactic monoid of a net, since we have defined in Chapter 6 "the state graph of a net."

A finite automaton has been said in the literature to be *decomposable* if there are several finite automata, each simpler than the given automaton, such that it is possible to build an equivalent machine in a series-parallel fashion from the smaller machines. "Series-parallel" means that there are no loops in the constructed machine except those that occur within the building blocks themselves.

The relevance of this problem for the task that now faces us should be apparent. There are several approaches to the theory of decomposition, of which the semigroup approach of Krohn and Rhodes (1965) is obviously appropriate. Indeed what we want to prove is in essence implied by the main theorem of that paper, which states (1) that any finite automaton can be built up in a series-parallel fashion out of finite automaton whose syntactic monoids are simple groups and some rather small finite automata whose syntactic monoids are small and group-free; and (2) a simple-group machine is needed if and only if that simple group is a homomorphic image of a maximal subgroup of the syntactic monoid. Thus it is implied that if the syntactic monoid of an event has only trivial subgroups then the event can be realized by a series-parallel network of small machines, which gives us our result in essence.

It turns out that the proof of the proposition of this section is much shorter than a proof of the Krohn-Rhodes theorem, even though the proof used here is based on part of the proof by Krohn and Rhodes (1965). There are two respects in which this chapter is a miniaturization of the Krohn -Rhodes theory. First, since we are concerned only with events in *GF*, we are able to avoid all the complication connected with group decomposability. Second, this chapter does not prove $LF \subseteq GF$.

We remind the reader that a net is an *RS* net if and only if it is buzzer-free and almost loop-free.

We shall accomplish our objective by constructing nets to represent monoids rather than by constructing nets to represent events directly. Thus suppose $M = \{e, m_1, \ldots, m_{r-1}\}$ is a finite monoid with identity *e*. The net of Figure 8.1 will represent *M* if the net performs the monoid multiplication: more precisely if, for any history during which one and only one of the inputs is 1 at any time and for each time *t*, the output at time *t* is the product $m_{i1} \cdot \cdot \cdot m_{it}$, where, for each $x \leq t$, m_{ix} is the input at time *x*.

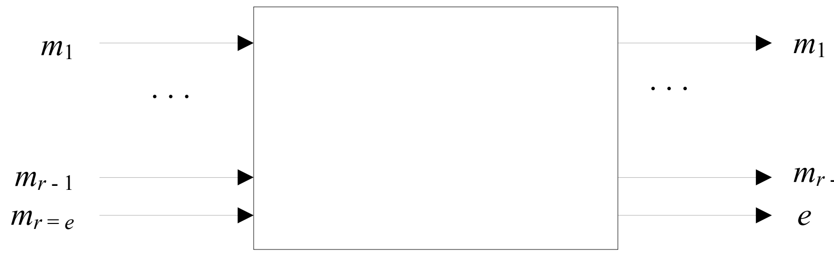

Figure 8.1. Monoid multiplication.

To justify this approach, we must show that, for any event L ∈ *NN*, having *M* as its syntactic monoid, a net representing L can be constructed from this net, which is almost-loop-free and buzzer-free if this net is. But this construction is simple; all that is required are decoders at each end. Thus let ∑ be the alphabet of L. If we suppose that the elements of ∑ are binary vectors, and if ψ is the homomorphism to the syntactic monoid *M*, then an input decoder computes ψ(∑). We shall omit this construction because it is a well-known technique to construct such a combinational net out of elements without feedback. Similarly an output decoder is needed: in fact, if $\psi(L) = \{m_{j1}, . . ., m_{js}\}$, then all we need is an “or” gate (i.e., a threshold gate whose threshold is 1) with *s* inputs: the input axons to this neuron will be from $m_{j1}, . . ., m_{js}$, and this neuron’s one axon will be the output of the net. From our discussion of the syntactic monoid it should be clear that a word at the input from time 1 through time *t* will get an output of 1 if and only if it is in L.

As an example, consider the event of Figure 8.2. One should compare this with Figure 4.2, which is like it except for the terminal state. Thus Figure 4.3 is the syntactic

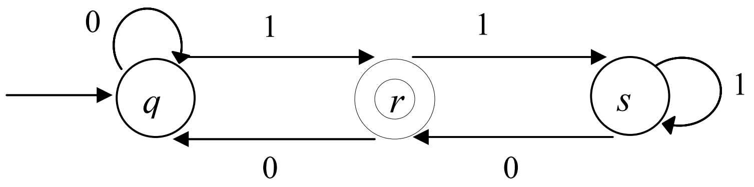

Figure 8.2. State graph for given example.

monoid for this event, which has order 8. Let us suppose *N*’, an eight-input eight-output net, represents this monoid. Then from *N*’ a net *N* to represent the event can be constructed as in Figure 8.3. Note that many of the inputs and outputs of *N*’ are not used.

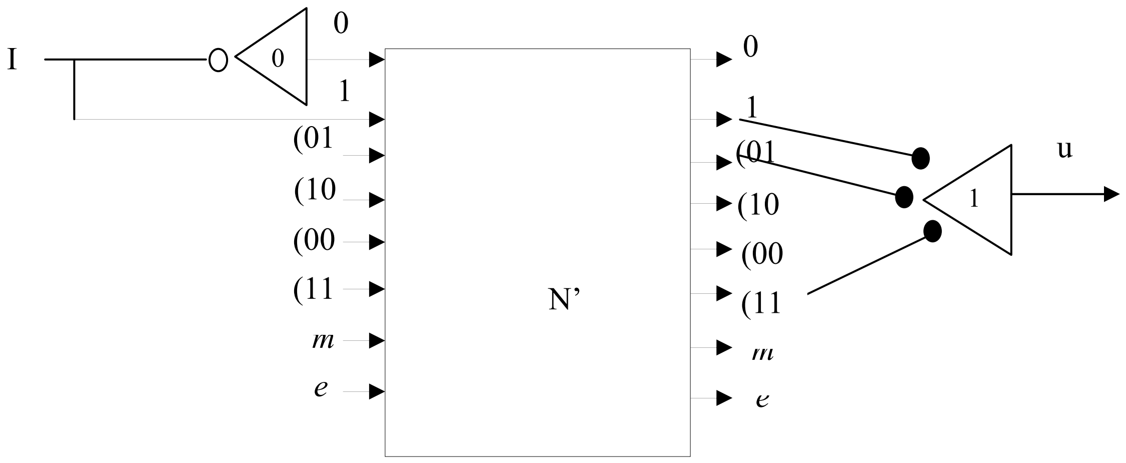

Figure 8.3. Monoid for Figure 8.2.

Thus the problem reduces to constructing a buzzer- free and almost loop-free net with the outline of Figure 8.1, for every finite monoid with only trivial subgroups. We shall say a monoid is realizable if it is represented by a buzzer free and almost loop-free net, all of whose axons have delay zero or one. This last property is a technical convenience. An element *a* in a finite monoid is a right identity if $xa = x$, for all $x \neq e$. note that *e* is a right identity. And the set of all right identities is a submonoid, since if *a*, *b* are right identities and $x \in M$, we have $x(ab) = (xa)b = xb = x$.

Theorem 8.1. A finite monoid consisting of only right identities is realizable.

Proof: Let $M = \{e, m_1, \ldots, m_{r-1}\}$, where $m_i m_j = m_i$, $1 \leq i, j \leq r - 1$. A buzzer-free and almost loop free net for M is given in Figure 8.4, in which r is taken as 4. It is left to the reader to see that Figure 8.4 could be generalized for any value of r.

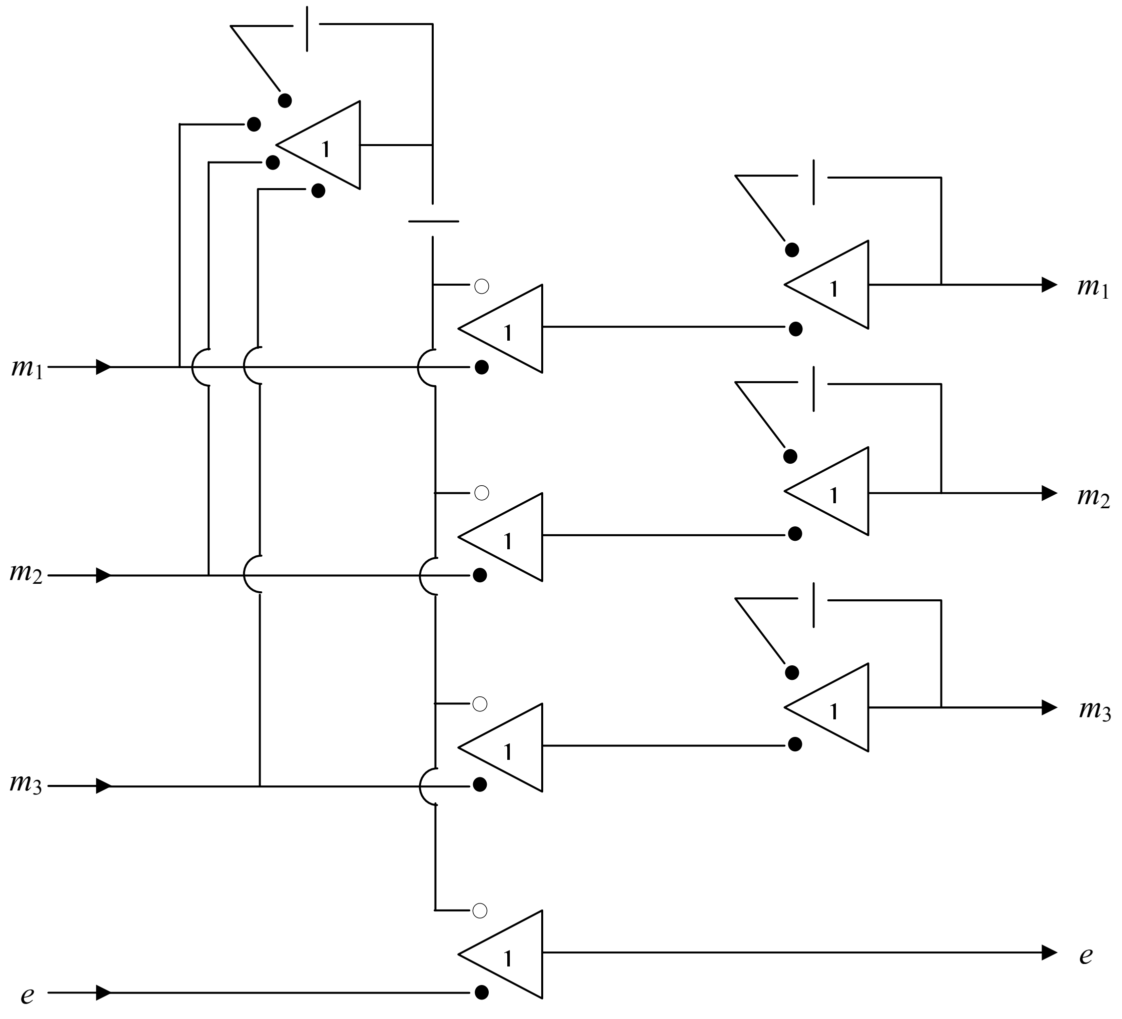

Figure 8.4. Buzzer-free and almost loop-free net.

A monoid that consists of the identity an all the positive powers of one of its elements is a *cyclic monoid*. A similar concept in the semigroup literature is a *cyclic semigroup*, which consists of all the positive powers of one of its elements. Thus a cyclic monoid is a cyclic semigroup with an identity added if it does not already have one. If we stipulate that the identity $e = a^0$, for every element a in the monoid, then we can say briefly that a cyclic monoid consist of all the nonnegative powers of one of its elements, which can then be called its “generator.” A cyclic monoid of finite order m has only trivial subgroups if and only if $a^{m-1} = a^m$, where a is its generator. Within isomorphism, there is only one such monoid of order m.

Theorem 8.2. A finite cyclic monoid with only trivial subgroups is realizable.

Proof: Let C_m be the cyclic monoid of order m with only trivial subgroups. The net realizing C_m (e.g., Figure 8.5 for C_4) has the inputs $I_0, \dots, I_{m-1}$ and outputs $U_0, \dots, U_{m-1}$. The input I_0 and the output U_0 represents e and, for each $i \geq 1$, I_i and U_i represent a^i. It also has junctions $J_1, \dots, J_{m-1}$ such that, for each i, the junction J_i is 1 at time t, if and only if either (1) J_i is 1 at time $t-1$, or (2) I_x is 1 at time t, for some $x \geq i$, or (3) for some x and y, $x + y \geq i$, and $x < i$, J_x is 1 at time $t-1$ and I_y is 1 at time t. The output U_0 is 1 at time t if and only if J_1 is 0 at time t. For $1 \leq i \leq m-2$, the corresponding U_i is 1 at time t if and only if J_i is 1 and J_{i+1} is 0 at time t. The output U_{m-1} is 1 at time t if and only if J_{m-1} is 1 at time t. From this description it should be clear that the buzzer-free and almost loop-free net can be constructed.

As the net for C_4 is shown in Figure 8.5, where axons with the same label are to be thought of as connected, and where U_3 and J_3 are identified. The junction J_0 has no function and does not appear. Note that a cyclic monoid other than a trivial cyclic monoid has a nontrivial subgroup and hence is not realizable.

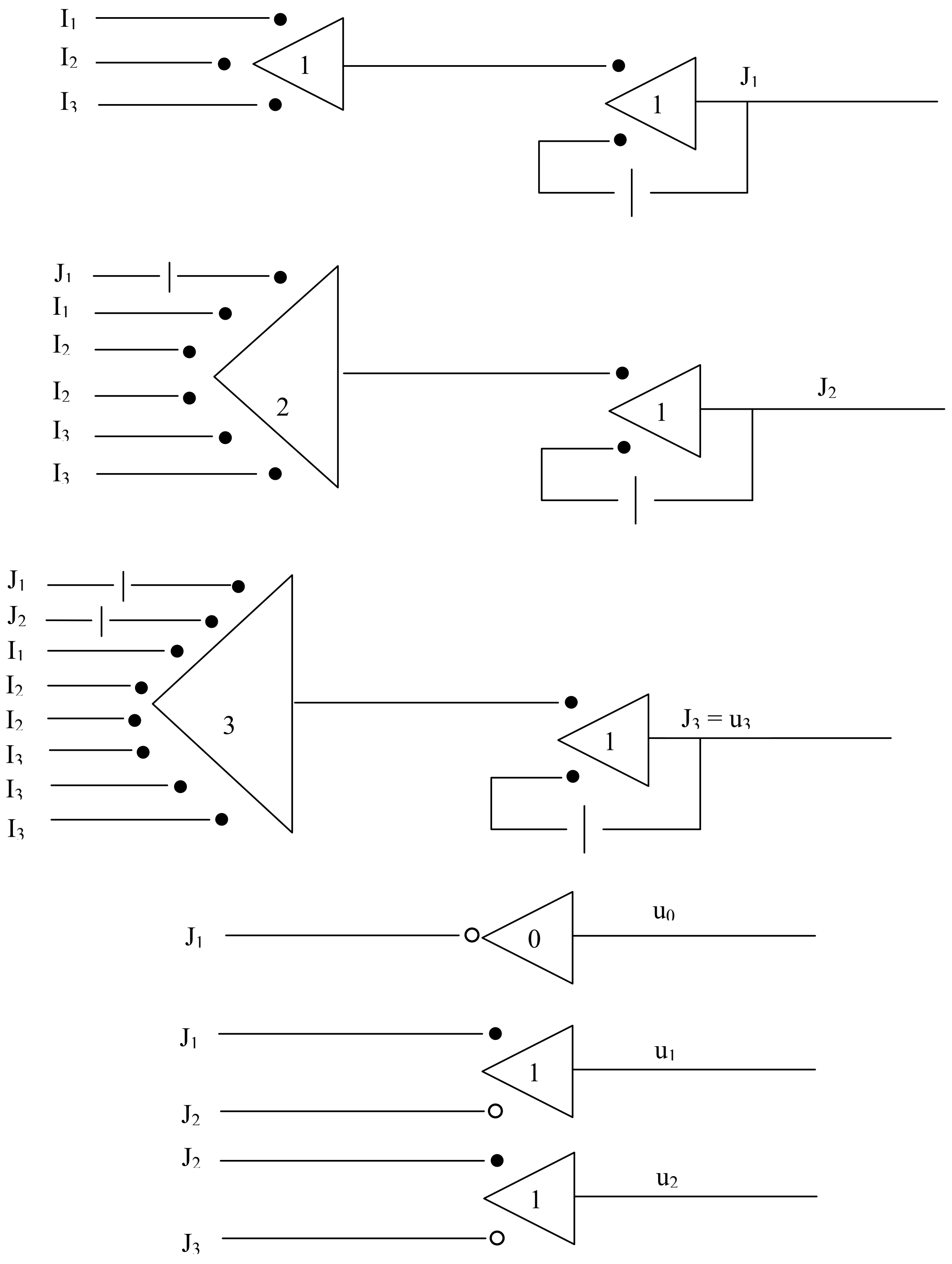

Figure 8.5. Net realizing cyclic monoid for order 4.

For the special purpose of this chapter we shall say that monoids L and S constitute a decomposition of the finite monoid M if (1) $e \in L$ and $L - \{e\}$ is a left ideal of M, (2) $e \in S$ and S is a submonoid of M, (3) $M = L \cup S$, (4) $|L| < |M|$, and (5) $|S| < |M|$. (The quantity $|A|$ is the cardinality, or order, of the set A, and $A \subseteq M$ is a left ideal of M if $MA = A$.)

Note that if monoids L and S constitute a decomposition of M and if M has only trivial subgroups, then so do L and S. The monoid M is decomposable if there exist L and S that constitute a decomposition of M. The concept of decomposition used in this section is narrower than most such concepts in the algebraic literature; our restricted concept is not theoretically interesting; its only virtue is that it works for the purpose at hand.

As an example, $L = \{e, 1, 01, 11, m\}$ and $S = \{e, 0, 00, 10\}$ constitute a decomposition of the monoid of Figures 4.4 and 4.5.

Theorem 8.3. If L and S constitute a decomposition of a finite monoid M, and if L and S are realizable, then so is M.

Lemma. Let N be a net all of whose axons have delay 0 or 1. Then there is a net N' like N except for an extra input I that clears N': N' behaves like N as long as I is 0; if I is 1 at time t, then no neuron fires at time t, whereupon if t' is the first time after t at which I is 0 the behavior of N' from t' on is like the behavior of N from time 1 on. If N is buzzer-free and loop-free, so is N'. Also N' has no axons with delay greater than 1.

Proof: Given N, N' is constructed as follows: enough inhibitors from I are added to each neuron so that, no matter what else happens, if I is 1 the neuron fails to fire; the threshold remains the same, so that when I is 0 the neurons will fire or not fire as before. For example, if the neuron had a threshold of r and j excitory incident axons, $j - r + 1$ inhibitory axons are added coming from *I*. The net *N*' is the result of so modifying each of *N*'s neurons. That *N*' will behave as specified follows from the fact that *N*'s axons all have delay 0 or delay 1.

Consider Figure 8.6 for the proof of Theorem 8.3.

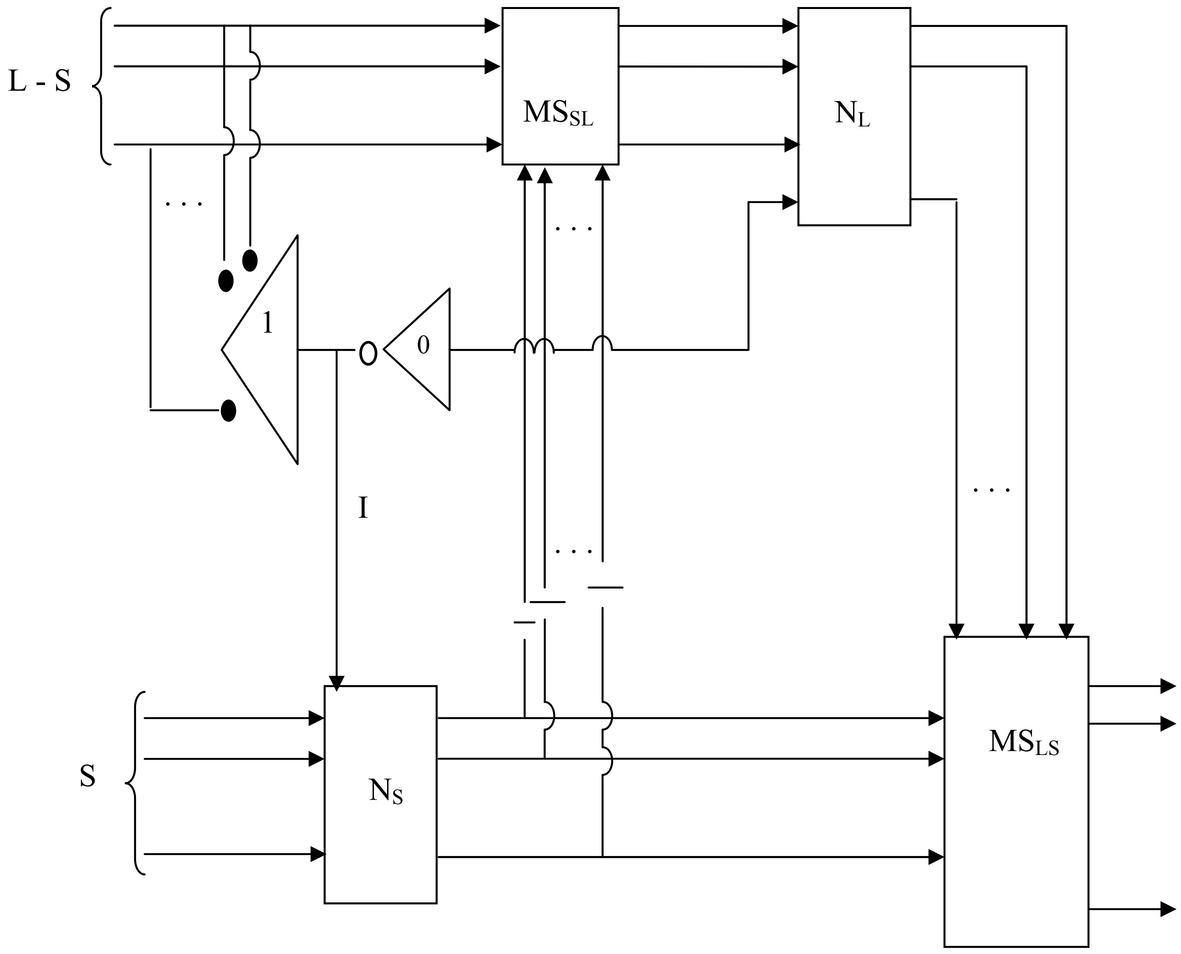

Figure 8.6. Net to represent monoid M.

Proof of theorem: The net to represent *M* is constructed out of the nets N_L, the realizing *L*, and N'_S, the net realizing *S* with clearing input *I*, as shown in Figure 8.6. The elements of *M*, as inputs, are divided into two sets: those in *S* and those in $L - S = M - S$. As long as inputs are from *S*, *I* is 0 and N'_S multiplies the inputs according to the monoid *S*. But when an input in $L - S$ appears, *I* becomes 1, clearing *N*'. Now, MS_{SL} is a combinational net (*i.e.*, a loopless net all of whose axon delays are 0) for taking products xy where $x \in S$ and $y \in L - S$. ("*MS*" stands for "multiplicative switch.") Note that since $L - \{e\}$ is a left ideal of M, $S(L - \{e\}) \supseteq L - \{e\}$; a fortiori, $S(L - S) \subseteq L - \{e\}$. The net MS_{SL} is so constructed that if none of $L - S$ is 1 then none of the outputs of MS_{SL} is 1; in this case the input to N_L will be e. On the other hand, if none of the bottom row of input is 1 (which happens at the initial moment of time, and at any moment immediately after a moment with an input from $L - S$), then MS_{SL} interprets this as an e from N'_S. Thus if the inputs at time $t - 1$ and t are both in $L - S$, then the output at time t from MS_{SL} will be exactly the same as the input at time t. The construction of MS_{SL} with the required properties is straightforward, given knowledge of the principle of combinational nets.

The net MS_{LS} is another combinational net (in fact, another multiplicative switch) that serves to multiply $x \in L$ and $y \in S$ to obtain xy: x is the top input and y is the left input. Note that since $L - \{e\}$ is not known to be a right ideal; nothing can be said about this product except that it is in M. Note also that, always, one of the top inputs to MS_{LS} is 1. In case none of the left inputs in 1, MS_{LS} interprets this situation as an input of e.

To summarize the action of various parts of the net, suppose $x_1, \ldots, x_t$ is the input history of the net. Then the output of N_L at time t is the product $x_1 \cdots x_{t'}$, where $x_{t'}$ is the last input in $L - S$, or e if none of $x_1, \ldots, x_t$ is in $L - S$. The output of N'_S is the product $x_{t'+1} \cdots x_t$; if $t = t'$, then N' has no output at time t (because of the clearing input I); and, if none of $x_1, \ldots, x_t$ is in $L - S$, the output of N'_S equals the product $x_1 x_2 \cdots x_t$. In any case the output of MS_{LS} at time t is $x_1 \cdots x_t$, showing that the net as a whole does represent the monoid M.

The proof of Theorem 8.3 is concluded except for two sample net histories to help clarify the construction. First, consider the input history x_1, x_2, x_3, x_4 (times 1 through 4 inclusive), where $x_1 \in L - S$, $x_2 \in L - S$, $x_3 \in S$, and $x_4 \in S$. Then the bottom inputs of MS_{SL} are all 0 up to and including time 3. The N_L will put out the product x_1x_2 at time 2. At time 3, it will multiply by e and the output of N_L will again be x_1x_2. The net N'_S has no output at time 1 or time 2, but at time 3 it will have x_3 as output. At time 3, MS_{LS} will multiply x_1x_2 by x_3 and output $x_1x_2x_3$. At time 4, N_L still outputs x_1x_2 (an e having come at time 4); but now N'_S outputs x_3x_4, and so MS_{LS} outputs $x_1x_2x_3x_4$.

Now consider the input history x_1, x_2, x_3 (that is, at times 1 through 3 inclusive), where $x_1 \in S$, $x_2 \in L - S$, and $x_3 \in S$. At time 1, N'_S outputs x_1 and N_L outputs e, and so MS_{LS} outputs x_1. At time 2, the bottom input to MS_{SL} is x_1 and the left input is x_2; thus x_1x_2 is the input to N_L and also the output. But, at time 2, I clears N'_S and thus there is no output from N'_S and no left input to MS_{LS}. Thus, at time 2, the output from MS_{LS} is x_1x_2. At time 3, the input to N_L is e and so its output is still x_1x_2. But the input to N'_S is x_3, and the output of N'_S is x_3 also. Thus, at time 3, MS_{LS} has x_1x_2 as top input and x_3 as left input, and gives $x_1x_2x_3$ as output.

The net of Figure 8.6, we feel, is important enough to warrant explanation from another point of view. Suppose we wish to write a program for a digital computer to multiply out a long word $m_1 \cdots m_\mathrm{t}$, each $m_i \in M$. Suppose that we have (1) a subroutine for element-by-element multiplication in M, (2) a quick subroutine for multiplying out long words $x_1 \cdots x_t$, each $x_i \in L$, and (3) a quick subroutine for multiplying out long words $x_1 \cdots x_t$, each $x_i \in S$. Now we could do the program in terms of the first subroutine only, but it would take a lot of time. Using the second and third subroutine, we can write a faster program. The program first parses the given word as follows:

$m_1 \cdots m_{i1}/\ m_{i1+1} \cdots m_{i2}/\ m_{i2+1} \cdots m_{i3}/\ldots$

Here m_{i1+1} is the first element in the word in $L – S$; m_{i2+1} is the first element after m_{i1+1} in S; etc. The product $m_1 \cdot\cdot\cdot m_{i1}$ is then multiplied out by the third subroutine yielding a_1. The result a_1 and m_{i1+1} are multiplied by the first subroutine yielding m'_{i1+1}. Then $m'_{i1+1}\ m_{i1+2} \ldots m_{i2}$ is multiplied out by the second subroutine, and the result $b_2 = a_2$, representing the product $m_1 \cdot\cdot\cdot m_{i1}$, is stored. In like manner $m_{i2+1} \cdot\cdot\cdot m_{i4}$ is multiplied out yielding a_4. Then $b_4 = b_2a_4$, representing the product $m_1 \cdot\cdot\cdot m_{i4}$, is stored. The process continues; at each stage b_{2n} represents $m_1 \cdot\cdot\cdot m_{i2n}$, and $b_{2n+2} = b_{2n}a_{2n+2}$, where a_{2n+2} represents the product of the word $m_{i2n+1} \cdot\cdot\cdot m_{i2n+2}$.

Now there are two possibilities. Either the last phrase is a word of S, or it is $m_{i(2n+1)+1} \cdot\cdot\cdot m_{i2n+2}$ of $L – S$. in the latter case, the answer for the program b_{2n+2}. If , on the other hand, the last phrase is a word of S, $m_{i(2n+1)+1} \cdot\cdot\cdot m_{i2n+1}$, whose product is a_{2n+1}, then the answer must be obtained by the first subroutine, which multiplies b_{2n} by a_{2n+1}, yielding the answer $b_{2n}a_{2n+1} = m_1 \cdot\cdot\cdot m_{i2n+1}$.

Theorem 8.3 tells us that in order to realize a monoid we may try to decompose it, hoping for two smaller monoids that are realizable.

Theorem 8.4, which follows, shows that any non cyclic monoid with only trivial subgroups that is not made up exclusively of right identities is always decomposable. This theorem is the strategic algebraic step in this section.

Theorem 8.4. If M is a finite noncyclic monoid with only trivial subgroups, and if M has at least one element that is not a right identity, then M is decomposable.

Proof: Let m be the order of M. Then for every $x \in M$, $x^{m+1} = x^m$, by Theorem 4.11.

Lemma 1. For $a,b \in M$, if $ab = e$, then $a = b = e$.

Proof: Since $a^mb^m = e$, $a = a(a^mb^m) = a^{m+1}b^m = a^mb^m = e$. Similarly, $b = e$.

Note that this proof depends only on the fact that M is a finite monoid with only trivial subgroups.

Lemma 2. If a is not a right identity of M, then $(M - \{e\})a$ is a proper subset of $M - \{e\}$.

Proof: By Lemma 1, it is a subset. Assume that it is not a proper subset. Then, since $M - \{e\}$ is finite, for all $x, y \in M - \{e\}$, $xa = ya$ implies $x = y$. Also $xa^r = ya^r$ implies $x = y$. Let c be an arbitrary member of $M - \{e\}$. Since $a^{m+1} = a^m$, $ca^{m+1} = ca^m$, from which we obtain $ca = c$. thus a is a right identity, which is a contradiction.

The proof of Theorem 8.4 can now be divided into three cases. In each case we must define L and S and verify conditions (1)-(5) of the definition of "decomposition"

Case I: For some $a \in M$, a is not a right identity and $Ma \cup \{e\} = M$. Take $L = Ma^2 \cup \{e\}$ and $S = \{e, a\}^m$. Then $M(Ma^2 - \{e\}) \subseteq M^2a^2 \subseteq Ma^2$. But $e \notin M(Ma^2 - \{e\})$, by Lemma 1. Hence $M(Ma^2 - \{e\}) \subseteq M^2a - \{e\}$, and so $L - \{e\}$ is a left ideal. Since $e \in L$, condition (1) is verified.

That S is the cyclic submonoid generated by a is immediate from Theorem 4.11, and since $e \in S$, condition (2) is verified.

To prove condition (3), $M = S \cup L$, note that since $Ma \cup \{e\} = M$ and $a \neq e$, $Ma = M - \{e\}$, by Lemma 1. But $Ma = (M - \{e\})a \cup \{a\}$. Now $(M - \{e\}a$ is a proper subset of $Ma = M - \{e\}$, by Lemma 2. Thus a is the one and only member of Ma not in $(M - \{e\})a = Ma^2$, and thus the one and only member of M not in $Ma^2 \cup \{e\}$. Thus $M = S \cup L$.

Condition (4), $|L| < |M|$, follows from the fact that $a \neq L$. Condition (5), $|S| < |M|$, follows from the fact that S is a cyclic monoid, and M is not.

Case II: $MQ \cup \{e\} \neq M$, where Q is the set of all elements of M that are not right identities. Then, take $L = MQ \cup \{e\}$ and $S = M - Q$, that is the set of all right identities of M. Then the verification of condition (1) is as it was in Case I. The verification of conditions (2), (3), (4), and (5) are straightforward, condition (4) following from the hypothesis of Case II, and condition (5) from the hypothesis of the Theorem.

Case III: The hypothesis of neither Case I nor Case II is true. Then there exists an $a \in Q$ and a nonempty subset Q' of Q such that $MQ' \cup \{e\} \neq M$ but $M(Q' \cup \{a\}) \cup \{e\} = M$. Take $L = MQ' \cup \{e\}$ and $S = Ma \cup \{e\}$. Conditions (1) and (2) are again easily verified. We have $L \cup S = MQ' \cup Ma \cup \{e\} = M(Q' \cup \{a\}) \cup \{e\} = M$, establishing condition (3). Condition (4) follows from the assertion introducing Q' and condition (5) follows from the falsity of Case I.

Theorem 8.5 (Main Theorem). GF $\cap$ NN $\subseteq$ LF.

Proof: Let L $\in$ *GF* $\cap$ *NN*. We know that if the syntactic monoid of L is realizable then there is a loop-free net representing L. If Theorem 8.5 is false, therefore, there is a smallest r such that some monoid M of order r with only trivial subgroups is not realizable. By Theorem 8.1 and 8.2, M cannot consist exclusively of right identities, nor can M be a trivial cyclic monoid, implying that it cannot be cyclic. Thus by Theorem 8.4, M is decomposable into monoids S and L, each of order less than r. By hypothesis, S and L are each realizable. But then, by Theorem 8.3, so is M, a contradiction.

In spite of the reduction ad absurdum form of the proof of Theorem 8.5, the proofs of Theorem 8.1 through 8.4 five us a method of constructing a buzzer-free and almost loop-free net for an event in *GF* $\cap$ *NN*.

The following are exercise solutions from the book.

1. Prove that, if M is a finite monoid with only trivial subgroups, there are $m_1, m_2 \in M$ such that $|m_1 M m_2| = 1$. (Such a monoid is in this respect the antithesis of a group G, since, for any $g_1, g_2 \in G$, $|g_1 G g_2| = |G|$.) (Hint: use Theorem 8.4)
   Ans: Induction on the size of M. By Theorem 8.4 there exists a submonoid and left ideal L such that L < M etc. By proof details in all cases L contains a and Ma for a non right identity 'a'. So consider L' = $Ma \cup \{e\}$ instead of L. Now L' has no nontrivial subgroups, else M does.
   Case 1: $b \in Ma$ is not a right identity on L'. By induction there exits $c, d \in$ L such that $|c$ L'$d| = 1$ or $|c\ m_a\ d| = 1$.
   Case 2: Each $b \in Ma$ is a right identity on L'. Then $\forall\ x \in M$, let $xa = b \in Ma$. Then $a(xa) = a$, whence $|a\ M\ a| = 1$.

CHAPTER 9

COMPLETION OF THE MAIN ARGUMENT

In the book (Counter-Free Automata by McNaughton and Papert) Chapter 10 corresponds to this chapter.

We are now in a position to prove that all the characterization of noncounting events are virtually equivalent. Our procedure is first to use chapter 1-8 to link together all the characterization. After that, two theorems are proved that enable us to link in the symbolic-logic characterization. It is to be noted that chapter 7 (in which $LF \subseteq SF$ is proved) could have been omitted, with the work on symbolic logic providing the link between *LF* and *SF*.

Theorem 9.1. $NC \cap NN = LTO \cap NN = SF \cap NN = GF \cap NN = PF \cap NN = LF$.

Proof: This follows from (1) $GF \cap NN \subseteq LF$, (2) $LF \subseteq SF \cap NN$, (3) $SF \cap NN \subseteq LTO \cap NN$, (4) $LTO \cap NN \subseteq NC \cap NN$, (5) $NC \cap NN = PF \cap NN = GF \cap NN$. Statement (1) is Theorem 8.5; (2) follows from Theorem 7.4 and the fact that $LF \subseteq NN$. Statement (3) follows from Theorem 3.1, (4) follows from Theorem 2.1, and (5) follows from Theorem 5.1.

Theorem 9.2. NC = LTO = SF = GF = PF.

Proof: With Theorem 3.1, 2.1, and 5.1, all we need establish is that $NC \subseteq SF$.

Lemma. If $L \in NC$, then $L - \{\lambda\} \in NC$.

Proof: If $L \in NC$, then, for some k and for all V, W, X, $y \geq k$, $VW^yX \in L$ if and only if $VW^{y+1}X \in L$. But if $VW^yX = \lambda$ if and only if $VW^{y+1}X = \lambda$. So $VW^yX \in L - \{\lambda\}$ if and only if $VW^{y+1}X \in L - \{\lambda\}$, showing that $L - \{\lambda\} \in NC$.

To prove that $NC \subseteq LF$, let L $\in NC$. Then L – {λ} $\in NC$ by the Lemma and hence L – {λ} $\in NC \cap NN$. By Theorem 10.1 L – {λ} $\in SF \cap NN$. Hence there is a star-free expression ψ for L – {λ}. But then ψ or $\psi \cup \lambda$ is a star free expression for L, and hence L $\in SF$.

REFERENCES

WORKS CONSULTED

J. A. BRZOZOWSKI, *Derivatives of regular expressions*, J. Assoc. Computing Machinary, 11 (1984), pp. 481-494.

A. W. BURKS, R. MCNAUGHTON, C. H. POLLMAR, D. W. WAREEN, AND J. B. WRIGHT, *Complete decoding nets: General theory and minimality*, J. Soc. Industrial and Applied Math, 2 (1954), pp. 201-243.

A. W. BURKS AND H. WANG, *The logic of automata*, J. Assoc. Computing Machinary, 4 (1957), pp. 193-218, 279-297.

A. W. BURKS AND J. B. WRIGHT, *Theory of logical nets*, Proc. IRE, 41 (1953), pp. 1357-1365.

N. CHOMSKY AND M. P. SCHUTZENBERGER, *The algebraic theory of context-free language*, in Computer Programming and Formal Systems, P. Braffort and D. Hirschberg, eds., North Holland Publishing Co., Amsterdam, 1963, pp. 118-161.

A. CHURCH, *Proceedings of the International Congress of Mathematicians of August 1962*, Institute Mittogleffler, Djursholm, Sweden, 1963.

A. GINZBURG, *Algebraic Theory of Automata*, Academic Press, New York, 1968.

S. C. KLEENE, R*epresentation of events in nerve nets and finite automata*, in Automata Studies, C. E. Shannon and J. MaCarthy, eds., Princeton University Press, Princeton, 1956, pp. 3-42.

N. E. KOBRINSKII AND B. A. TRAKHTENBROT, *Introduction to the Theory of Finite Automata*, North Holland Publishing Co., Amsterdam, 1965.

K. KROHN AND J. RHODES, *Algebraic Theory of Machines. I. Prime Decomposition Theorem for Finite Semigroups and Machines*, Trans. Am. Math. Soc., 116 (1965), pp. 450-464.

R. MCNAUGHTON, *Symbolic logic for automata*, Wright Air Development Decision Tech, Note No. 60-244, Cincinnati, Ohio, 1960.

R. MCNAUGHTON, *Techniques for manipulating regular expressions*, in Systems and Computer Science, J. F. Hart and S. Takasu, eds., University of Toronto Press, Toronto, 1967, pp. 27-41.

R. MCNAUGHTON AND S. PAPERT, *Counter-free automata*, M.I.T research monograph no 65, The MIT Press, 1971.

R. MCNAUGHTON AND H. YAMADA, *Regular expressions and state graphs for automata*, Trans. IRE, EC-9 (1964), pp. 39-47.

T. MEDVEDEV, *On the class of events representable in a finite automaton*, in Sequential Machines, Selected Papers, E. F. Moore, ed., Addison-Wesley, Reading, Massachusetts, 1964, pp. 215-227.

A. R. MEYER, *A note on star-free events*, J. Assoc. Computing Machinery, 16 (1969), pp. 220-225.

M. L. MINSKY, *Computation: Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, N.J., 1967.

M. PERLES, M. O. RABIN, AND E. SHAMIR, *The theory of definite automata*, Trans. IEEE, EC-12 (1963), pp. 233-243.

M. PHISTER, *Logical Design of Digital Computers*, Wiley, New York, 1958.

M. P. SCHUTZENBERGER, *On finite monoids having only trivial subgroups*, Information and Control, 8 (1965), pp. 190-194.

M. P. SCHUTZENBERGER, *Sur certaines varietes de monoides finis*, in Automata Theory, E. R. Caianiello, ed., Academic Press, New York, 1966, pp. 314-319.

M. P. SCHUTZENBERGER, *On a family of sets related to McNaughton's L-language*, in Automata Theory, E. R. Caianiello, ed., Academic Press, New York, 1966, pp. 320-324.

R. E. STEARNS AND J. HARTMANIS, *Regularity preserving modifications of regular expressions*, Information and Control, 6 (1963), pp. 55-69.

Y. ZALCSTEIN, *On star-free events*, in Conference Record of the Eleventh Annual Symposium on Switching and Automata Theory, IEEE, New York, 1970, pp. 76-80.