STRUCTURAL SYNTHESIS AND ANALYSIS OF KINEMATIC CHAINS USING PATH MATRICES

By

FREDERICK FENN QUIST, JR.

Bachelor of Science

United States Military Academy

West Point, New York

1964

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1971

Thesis
1971
Q8s
cop. 2

STRUCTURAL SYNTHESIS AND ANALYSIS OF KINEMATIC CHAINS USING PATH MATRICES

Thesis Approved:

Atmaram H. Soni
Thesis Adviser

C E Price

[illegible]
Dean of the Graduate College

788734

PREFACE

This thesis presents a method of describing kinematic chains and mechanisms using matrices which represent paths or circuits in a chain. The path matrices developed permit the use of the digital computer in developing mechanisms from kinematic chains, synthesizing modified chains based on basic kinematic chains, and comparing kinematic chains. Computer programs using the path matrices were applied to the 230 ten-link chains and resulted in tremendous amounts of data which are summarized in the body of the thesis.

I would like to acknowledge my indebtedness to my adviser, Dr. A. H. Soni, whose inspiration and support were instrumental in the development of this thesis.

Indebtedness is also acknowledged to the National Science Foundation which furnished part of the finances for this research under NSF Grant GU-3160.

Finally, I wish to thank my wife and daughter for their patience, understanding, and support.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

During the past several years, the problems of structural synthesis and analysis of kinematic chains have been approached in several ways. Davies and Crossley (1)* utilized Franke's condensed notation, a method of representing kinematic chains as molecules, to obtain a census of seven, nine, ten, and eleven-bar kinematic chains. Hain (2,3,4,5) has done a great deal of work in this area of kinematics, working mainly with the six and eight-link chains. He has obtained censuses not only for the basic chains, but also for these chains with modifications such as the addition of cam pairs, prism pairs, and multiple joints. In addition, he has obtained the mechanisms derived from these chains and presented them schematically thus furnishing the designer with a large choice of solutions to a given problem. This work has been done mainly by visual inspection of the structural drawing of each chain.

The results obtained by the above individuals have relied heavily on the researcher's ability to visualize the various configurations a chain may assume and to recognize isomorphisms which may occur. As more complex chains are examined, this ability is heavily taxed and the tremendous number of chains and mechanisms makes a manual approach

*Numbers in parentheses refer to numbered references in the Bibliography.

impractical. Therefore, the need arises for an automated method of performing the structural analysis of kinematic chains.

Woo (6) has used graph theory in enumerating the plane kinematic chains with 13 revolute pairs and 10 links, thus verifying some of the results obtained by Davies and Crossley above. An incidence matrix was used to develop the chains and also served as the structural descriminant to detect isomorphisms. Similarly, Dobrjanskyj and Freudenstein (7) have developed an algorithm which applies graph theory to the comparison of kinematic chains and mechanisms. Buchsbaum (8) has further developed the use of graph theory for the structural classification and type synthesis of chains with multiple elements.

All of the preceding approaches which utilize graph theory have the advantage of being computerized and have contributed significantly to the analysis of kinematic chains.

The purpose of this thesis is to present an alternate approach to the problem of type synthesis and analysis of kinematic chains which does not rely on graph theory but utilizes a method of comparing paths or circuits in a chain. A Chain Path Matrix (CPM) is developed and from it, the Mechanism Path Matrix (MPM) is formed. The MPM serves as the structural discriminant in comparing mechanisms and chains. The CPM is useful in converting basic kinematic chains into chains with special elements such as pulleys and belts, cams, double joints, etc. The method is easily computerized and, consequently, fills the need for automated methods of processing chains with more than eight links. Furthermore, it has the advantage of requiring no mathematics in its derivation or application.

The path matrix approach has been applied to the analysis of the

230 ten-link chains with one degree of freedom (Appendix A). The amount of data generated by this anaylsis is enormous, and, consequently, has not been included in this thesis. The data has, however, been included in reference (9), a copy of which is available in the Oklahoma State University Library and summaries of this data are included at the end of each appropriate chapter herein.

In the chapters which follow, the development of the CPM and MPM will be described. The application of these matrices in synthesizing mechanisms, developing chains with special elements (cams, springs, etc.) and converting kinematic chains to gear trains will then be discussed.

CHAPTER II

DEFINITIONS OF NOTATION AND TERMS

In order to develop a basis for the following chapters, a brief discussion of notation and terms will be presented.

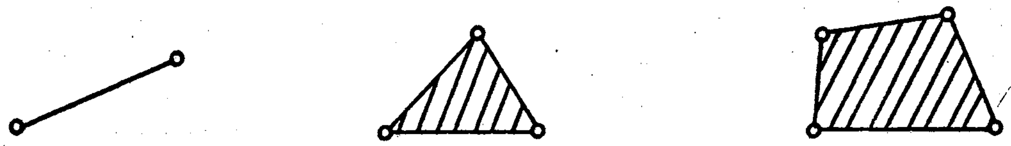
A kinematic link is a rigid, massless member and is classified according to the number of joints which it can accept. A binary link is a link which can accept two joints; a ternary link, three joints; and a k-nary link, k joints. The standard kinematic notation used to represent various links is illustrated in Figure 1.

A kinematic pair consists of two contacting elements and is used to connect two kinematic links. Three types of kinematic pairs will be considered here:

1.	Revolute Pair	- Permits only relative rotation between two connected links. Also called a turning pair or pin joint.
2.	Prism Pair	- Permits only relative translation between two connected links. Also called a sliding pair.
3.	Cam Pair	- Permits both rotary and translatory motion between two connected links.

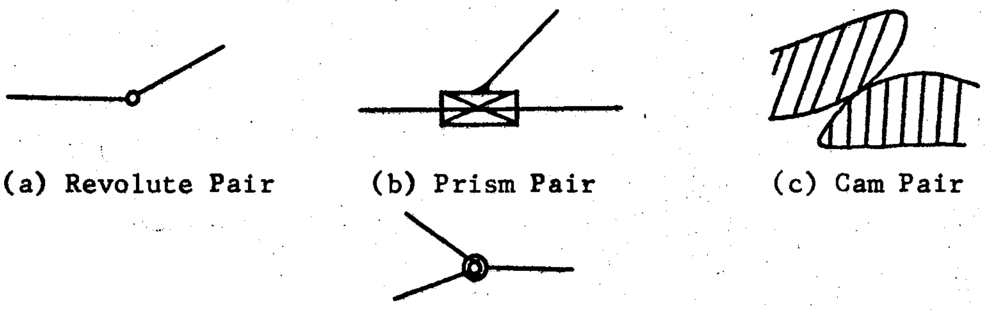
In addition to the three pairs mentioned above, the double joint will also be used as a means of connecting kinematic links. The double joint is simply the merging of two revolute pairs to form a joint which permits rotational motion among three links.

The kinematic notation used to represent the various joints described above is illustrated in Figure 2.



(a) Binary Link (b) Ternary Link (c) Quarternary Link

Figure 1. Representation of Kinematic Links



(a) Revolute Pair (b) Prism Pair (c) Cam Pair

(d) Double Joint

Figure 2. Representation of Kinematic Joints

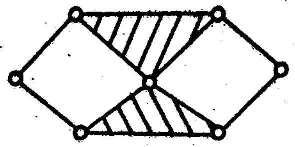


Figure 3. Six-Link Basic Kinematic Chain

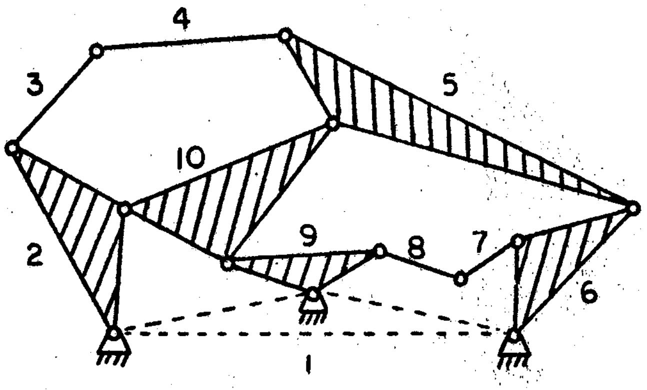


Figure 4. Ten-Link Mechanism

A kinematic chain is constructed using kinematic links and pairs to form one or more closed loops. Each link must be free to move relative to the links connected to it (e.g., a structure such as three binary links joined by three revolute pairs to form a triangle is not a kinematic chain). If a kinematic chain is constructed so that the motion of all links takes place in one plane or in parallel planes, the chain is called a planar kinematic chain. A basic kinematic chain is a planar kinematic chain with all links connected by revolute pairs (Figure 3).

A kinematic mechanism is defined as a kinematic chain which has one link fixed with respect to ground and which has a single degree of freedom. The degrees of freedom of a basic kinematic chain (and the mechanisms derived from it) can be determined from the formula $F = 6(N-1)-5P_1$ where F is the degrees of freedom, N is the number of links, and P_1 is the number of revolute pairs. Basically, a mechanism (or chain) with one degree of freedom is one in which the motion of all links is constrained, that is, no link is free to move independently of the other links. Figure 4 illustrates a ten-link mechanism.

A path, as used here, will be defined as a sequence of connected elements encountered in traversing a chain (or mechanism) beginning at a given element and returning to that element without repeating any element. For example, Figure 5 illustrates a kinematic chain with four loops. The links have been numbered as a means of identification. One path beginning at link 1 could be described by the sequence of link numbers 1,2,10,9. This path represents the links encountered in traversing loop 1 in a clockwise direction. A second path can be represented by the sequence 1,2,10,5,6,7,8 and can be thought of as the path

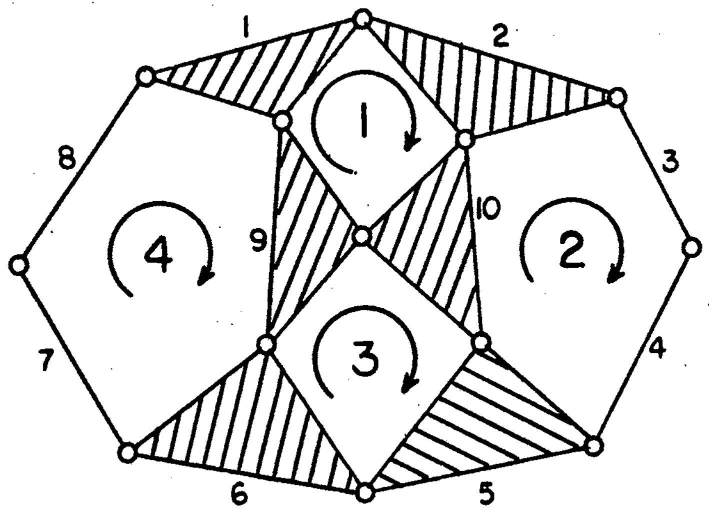


Figure 5. Four Loop Kinematic Chain

generated by combining loops 1, 3, and 4. Several other paths may also be found which originate at link 1. Each one of these paths can be thought of as a combination of either loop 1 or loop 4 (loops which contain link 1) and one or more of the other loops of the chain. This point will be further illustrated in the next chapter.

The foregoing provides sufficient background for an understanding of the path comparison method. The following chapters will describe the development of the path matrices and their applications.

CHAPTER III

DEVELOPMENT OF THE CHAIN PATH MATRIX

In the previous chapter, a path was defined as the sequence of elements encountered in traversing a loop or combination of loops in a kinematic chain. In this chapter, the development of the Chain Path Matrix (CPM), which represents all possible paths in a chain, will be discussed and a computer program for generating the CPM will be described.

In order to derive any meaning from the numbers representing the links in a chain, it is desirable to be able to designate both the link type (binary, ternary, quaternary, etc.) and the link number. To do this, a three or four digit integer will be used in which the ones and tens places represent the link number and the hundreds and thousands places represent the link type. For example, link number 1 of chain 189 (Figure 6) is a ternary link and this fact will be reflected by placing a three in the hundreds place of its identifying number. Since it is also link number 1, the digits 01 will be placed in the tens and ones positions. The identifying number for link 1 will then be 301. Similarly, the identifying number for link 2 is 402 and for link 3 is 203. The thousands place of the identifying number will be used in later chapters to designate special elements inserted in the basic kinematic chain.

The development of the CPM is based on a matrix called the loop

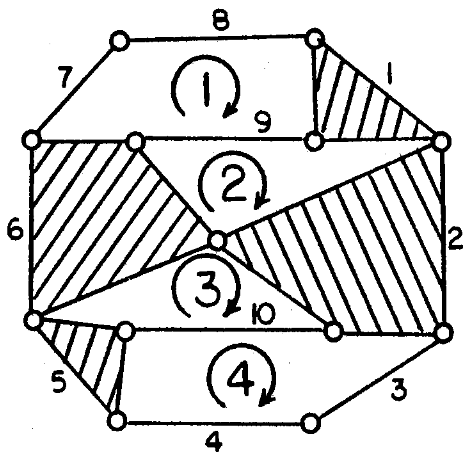


Figure 6. Ten-Link Chain Number 189

matrix which represents the path around each loop of the chain. The path is developed in a clockwise direction as used here although the direction is immaterial as long as all paths are taken in the same direction. Using the identifying numbers described above, the loop matrix for chain 189 would appear as shown in Table I. The starting element for each row may be chosen at random since the beginning and end elements of each row are considered to be adjacent.

The CPM will be composed of all possible combinations of the basic loops described by the loop matrix. The CPM will then be formed as shown in Table II and will identify every path or circuit which can be formed in the kinematic chain. In order for two loops to combine, they must have at least two elements in common, otherwise, an element in the combined path would be repeated. Because of this restriction, some of the combinations shown in Table II cannot be realized. When this case occurs, the row of the CPM representing that combination will be filled with zeroes to indicate that the path represented by that row does not exist. Based on the preceding, the CPM for chain 189 would be as shown in Table III.

Technically, neither the loop matrix nor the CPM as shown are true matrices since they are not rectangular arrays. However, the arrays as used in the computer program are true matrices as shown below.

To develop the CPM by computer, the loop matrix is required as the initial input. In most cases, developing the loop matrix is a simple matter but, for a certain class of chains such as Group 64D as shown in Appendix A, problems arise. These chains have paths which cross one another and, consequently, basic loops are not available. The loop matrices actually used for this class of chains (Appendix B) were

TABLE I

LOOP MATRIX FOR CHAIN 189

ROW 1	301	209	406	207	208
ROW 2	301	402	406	209	
ROW 3	402	210	305	406	
ROW 4	402	203	204	305	210

TABLE II

COMBINATIONS OF LOOPS USED IN FORMING THE CPM

ROW #	Loop Combinations
1	Loop 1
2	Loop 2
3	Loop 3
4	Loop 4
5	Loops 1 and 2
6	Loops 1 and 3
7	Loops 1 and 4
8	Loops 2 and 3
9	Loops 2 and 4
10	Loops 3 and 4
11	Loops 1 and 2 and 3
12	Loops 1 and 2 and 4
13	Loops 1 and 3 and 4
14	Loops 2 and 3 and 4
15	Loops 1 and 2 and 3 and 4

TABLE III

CPM FOR CHAIN 189

ROW #	Sequence
1	301 209 406 207 208
2	301 402 406 209
3	402 210 305 406
4	402 203 204 305 210
5	301 402 406 207 208
6	0 0 0 0 0
7	0 0 0 0 0
8	301 402 210 305 406 209
9	0 0 0 0 0
10	402 203 204 305 406
11	301 402 210 305 406 207 208
12	0 0 0 0 0
13	0 0 0 0 0
14	301 402 203 204 305 406 209
15	301 402 203 204 305 406 207 208

TABLE IV

COMPUTER LOOP MATRIX FOR CHAIN 189

301	209	406	207	208	0	0	0	0	0	0	5
301	402	406	209	0	0	0	0	0	0	0	4
402	210	305	406	0	0	0	0	0	0	0	4
402	203	204	305	210	0	0	0	0	0	0	5

arrived at by trial and error and no rules governing their development have become apparent.

The loop matrix is read into the computer in the form of a matrix and forms the first four (in the case of the ten-link chains) rows of the CPM. Sufficient columns are available in the matrix to contain the longest row which may occur (10 elements in the case of the ten-link chains) plus two. The two extra columns are required for use in certain operations performed in comparing rows. Any elements of the matrix which are not defined are filled with zeroes. As the loop matrix is read in, the number of non-zero elements in each row is counted and this number is stored in the last column of that row. Thus, the loop matrix (and the first four rows of the CPM for chain 189) would appear as shown in Table IV.

To form the remaining rows of the CPM, combinations of the first rows must be taken. In combining rows, the following rules are observed:

1. If an element is common to both rows and the elements on both sides of it are also common to both rows, that element will not appear in the combined row. For example, if rows 1 and 2 of Table IV are being combined, link 209 will not appear in the combined row since it is flanked by links 301 and 406 which are also common to both rows. Note that in row 2, 301 is considered to be adjacent to 209.

2. If row A is to be combined with row B, the path sequence will begin with A_1 (the first element in row A) and will continue to element A_m, the first element encountered which is common to both rows. If rule 1 does not apply and if A_{m+1} is the same as B_{n-1} (where B_n is the element in row B which is equivalent to A_m), the sequence will transfer

to row B at element B_n and will continue in row B until similar conditions permit a transfer back to row A to complete the path.

3. If no transfer can be made between rows, the resulting combined row will be filled with zeroes.

To illustrate the above procedure, the combining of rows 2 and 3 of the loop matrix for chain 189 (Table IV) will be performed step by step.

1. The first element of row 2 (301) is not common to both rows and will thus become the first element of the combined row.

2. The second element of row 2 (402) is the same as the first element of row 3. Rule 2 is applied to determine if a transfer can be made. Since the third element in row 2 (406) is the same as the last element in row 3 (which is considered as preceding the first element in row 3) the transfer can be made. The second element in the combined row thus becomes 402.

3. The sequence continues with elements 2 (210) and 3 (305) of row 3 since they are not common to both rows. At this point, the combined row is:

301 402 210 305.

4. The next element in row 3 (406) is common to both rows. Rule 2 is applied and, since the element following 406 in row 3 (402) is the same as the element preceding 406 in row 2, the sequence transfers back to row 2 and 406 becomes the fifth element of the combined row.

5. The sequence then continues in row 2 until the starting element is encountered. (When the sequence reaches the last element of a row, it starts over at the first element. Therefore, the first element of the row will be encountered after the last element of the row.)

6. The remainder of the combined row is filled with zeroes and the number of non-zero elements stored in the last column. The resulting row is:

```
301 402 210 305 406 209    0    0    0    0    0    6.
```

This row agrees with row 8 of the CPM as shown in Table III.

Rows 5 through 10 of the CPM are similarly derived. Row 11 of the CPM represents the combination of rows 1, 2, and 3 of the loop matrix. This combination can most easily be obtained by combining row 1 of the CPM with row 8 of the CPM (the combination of rows 2 and 3) or by combining row 3 with row 5 (the combination of rows 1 and 2). Since one or more of the combined rows mentioned above may be zero, the computer tests the rows to be combined and uses the first pair which have non-zero entries in their first columns. The possible combinations to be tested are provided as input data for the program. Rows 11 through 15 of the CPM are generated in this manner. The row combinations used for the ten-link chains appear in Table V.

TABLE V

ROW COMBINATIONS USED IN FORMING THE CPM FOR TEN-LINK CHAINS

ROW #	Combined Rows
5	1 and 2
6	1 and 3
7	1 and 4
8	2 and 3
9	2 and 4
10	3 and 4
11	1 and 8 or 5 and 3
12	4 and 5 or 7 and 2
13	10 and 1 or 3 and 7
14	8 and 4 or 2 and 10
15	5 and 10 or 6 and 9 or 7 and 8 or 12 and 3 or 13 and 2 or 14 and 1

CHAPTER IV

DEVELOPMENT AND USE OF THE MECHANISM PATH MATRIX

Theory

Once the CPM has been formed for a given chain, it represents the sequence of links encountered in traversing each possible path in the chain. The CPM could be used directly in comparing two chains but the procedure would involve checking each row in one matrix against each row in the other matrix and cycling one of the rows several times for each comparison. A method which will eliminate a great deal of the comparisons made in the above procedure is based on the premise that if a mechanism formed from one chain is identical to a mechanism formed from a second chain, the two chains are equivalent. Conversely, if a mechanism formed from one chain is not equivalent to any mechanism formed from the second chain, the chains are not equivalent. The problem thus becomes one of representing mechanisms and then, comparing two mechanisms.

As defined previously, a mechanism is created when one link of a kinematic chain is fixed with respect to ground. The configuration can be completely defined by defining all paths in the chain which include the fixed link. It will prove to be advantageous to use the fixed link as the starting element for each of the paths. A Mechanism Path Matrix (MPM) which describes the mechanism formed by fixing a specific link ~~is~~ developed by using only those rows in the CPM which contain the fixed link and by cycling each row so that the fixed link is the first element in the row. Table VI shows the CPM formed for chain 189 and Table VII, the MPM representing the mechanism formed by fixing link 1 of chain 189.

When comparing two paths, the sequence of link types encountered in traversing the paths is the factor which determines whether or not the paths are equivalent. Consequently, once the MPM is formed, the link numbers may be dropped leaving only the matrix of link types. Table VIII shows the MPM of Table VII reduced to a matrix of link types. The term Mechanism Path Matrix (MPM) will mean the matrix of link types in all following discussion.

To proceed further, the definition of equivalent paths must be established. Two paths are equivalent if and only if the sequence of elements encountered in traversing the first path in a given direction is the same as the sequence of elements encountered in traversing the second path in the same or opposite direction. That is, when comparing rows of two MPM's, the path represented by the sequence 2 3 2 3 4 is equivalent to the path represented by the sequence 2 4 3 2 3.

The equivalency of two MPM's, and therefore two mechanisms, will be established if each row in one path matrix has an equivalent row in the other path matrix, each row being used only once. Obviously, then, for two MPM's to be equivalent they must have the same number of rows and equivalent rows must contain the same number of non-zero elements. Only after these two conditions are satisfied is it necessary to compare sequences within two rows.

To illustrate the comparison of two MPM's, mechanisms 1 and 5 of chain 189 will be compared. The MPM for mechanism 5 is shown in Table

TABLE VI

COMPUTER CPM FOR CHAIN 189

301	209	406	207	208	0	0	0	0	0	0	5
301	402	406	209	0	0	0	0	0	0	0	4
402	210	305	406	0	0	0	0	0	0	0	4
402	203	204	305	210	0	0	0	0	0	0	5
301	402	406	207	208	0	0	0	0	0	0	5
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
301	402	210	305	406	209	0	0	0	0	0	6
0	0	0	0	0	0	0	0	0	0	0	0
402	203	204	305	406	0	0	0	0	0	0	5
301	402	210	305	406	207	208	0	0	0	0	7
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
301	402	203	204	305	406	209	0	0	0	0	7
301	402	203	204	305	406	207	208	0	0	0	8

TABLE VII

MPM FOR MECHANISM 1 OF CHAIN 189

301	209	406	207	208	0	0	0	0	0	0	5
301	402	406	209	0	0	0	0	0	0	0	4
301	402	406	207	208	0	0	0	0	0	0	5
301	402	210	305	406	209	0	0	0	0	0	6
301	402	210	305	406	207	208	0	0	0	0	7
301	402	203	204	305	406	209	0	0	0	0	7
301	402	203	204	305	406	207	208	0	0	0	8

TABLE VIII

MPM OF TABLE VII REDUCED TO MATRIX OF LINK TYPES

3	2	4	2	2	0	0	0	0	0	0	5
3	4	4	2	0	0	0	0	0	0	0	4
3	4	4	2	2	0	0	0	0	0	0	5
3	4	2	3	4	2	0	0	0	0	0	6
3	4	2	3	4	2	2	0	0	0	0	7
3	4	2	2	3	4	2	0	0	0	0	7
3	4	2	2	3	4	2	2	0	0	0	8

(1) Equal P
(2)

IX and will be compared with the MPM in Table VIII.

First, the two matrices have the same number of rows thus justifying further comparisons. Row 1 of Table VIII contains 5 elements and, therefore, may be equivalent to either row 2 or row 4 of Table IX. It is found to be equivalent to row 2 of Table IX. Similarly, each row of Table VIII is found to be equivalent to a row in Table IX and vice versa. Therefore, the two mechanisms are equivalent.

As mentioned previously, two chains can be compared by comparing mechanisms formed from them. However, other initial comparisons will determine whether or not comparison of mechanisms is warranted. Obviously, the two chains must have the same number of links and the same number of each type of link. Also, the CPM's must have the same number of rows thus indicating that the two chains have the same number of paths. Once these conditions have been met, mechanisms from each chain should be chosen for comparison. The choice of mechanisms will often reduce the number of comparisons required.

(1) Sa
l.
(2) S
each
lin
(3) C
n

For two mechanisms to be equivalent they must have the same type of fixed link. Thus, choosing the link type which appears least frequently in the chain will limit the number of mechanisms compared. For example, if two chains are to be compared and each chain has only one quarternary link, the only mechanisms which have to be compared are those formed from fixing the quarternary link.

Computer Development of the MPM

Subroutine MECH (Appendix D) uses the CPM to develop the MPM for each mechanism derived from a given chain. To do this, it takes each link (in numerical order) and forms the MPM from rows in the CPM which

TABLE IX

MPM FOR MECHANISM 5 OF CHAIN 189

3	4	4	2	0	0	0	0	0	0	0	4
3	2	4	2	2	0	0	0	0	0	0	5
3	4	2	3	4	2	0	0	0	0	0	6
3	4	4	2	2	0	0	0	0	0	0	5
3	4	2	2	3	4	2	0	0	0	0	7
3	4	2	3	4	2	2	0	0	0	0	7
3	4	2	2	3	4	2	2	0	0	0	8

contain that link. As the MPM is formed, each row is cycled so that the fixed link appears as the first element. In addition, the number of rows in each MPM is determined and this number is stored in the (N-1)st column of the first row of the MPM. This permits a rapid determination of the number of rows when the MPM's are being compared.

Computer Comparison of MPM's

Subroutine MECCOM (Appendix E) compares MPM's using the comparison procedure described earlier in this chapter as may be seen from the flow chart in Appendix E. An array of MPM's such as that developed by Subroutine MECH serves as the input to MECCOM. Each MPM is compared with all other MPM's and the results stored in array IEQUIV.

Two features are incorporated in Subroutine MECCOM which are not used in comparing the MPM's as they have been described thus far.

1. Provisions are made for cycling one of two rows being compared. This feature is used in comparing modified MPM's described in following chapters.

2. Provisions are made for zeroing certain rows of array JSTORE and then compacting the non-zero rows. This feature is used in the development of the modified MPM's described in following chapters.

Computer Program for Determining the Ten-Link Mechanisms

The main program and subroutines (other than LOOP, MECH, and MECCOM) used in determining the ten link mechanisms are shown in Appendix F. One page of the output obtained for chains 1 through 25 is shown in Table X. In Table X, the columns represent link numbers one through ten and the rows represent the chain being described. An entry

TABLE X

SAMPLE OUTPUT FOR TEN-LINK MECHANISMS

	LINK NUMBER										
CHAIN	1	2	3	4	5	6	7	8	9	10	UNIQUE MECHANISMS
1		1		3	1	1	3	3		9	3
2		1					3	4	6	5	5
3											10
4				3	2	1					7
5											10
6											10
7											10
8			2	1			6	5		9	5
9			2	1		5		5	5	7	4
10			2	1				5	6	7	5
11											10
12											10
13											10
14											10
15											10
16				3	2	1			7		6
17				2	1					9	7
18											10
19											10
20											10
21			2	1						9	7
22											10
23		1			4	3	3	4	4	3	3
24				1	2	3			8	7	5
25			1			5	4			8	6

in any column indicates that the link designated by that column is equivalent to the link number entered in the column. For example, in row 1 representing chain 1, links 2, 5, and 6 are equivalent to link 1, links 4, 7, and 8 are equivalent to link 3, and link 10 is equivalent to link 9. A blank entry indicates that the mechanism formed by fixing the link corresponding to the column number is a unique mechanism. For chain 1 there are three blank columns indicating that fixing links 1, 3, or 9 will give three unique mechanisms. Fixing any other link will result in a mechanism equivalent to one of the above three. The total number of unique mechanisms for each chain is reflected in the last column.

The entire program for developing the ten-link mechanisms took 6 minutes and 50 seconds to run on an IBM 360 Model 50 computer. The total number of unique mechanisms obtained was 1836.

CHAPTER V

CHAINS WITH SPRINGS

In Chapter I it was indicated that the CPM is useful in converting basic kinematic chains into chains with various elements such as springs and cams. In this chapter, the development of chains with springs will be covered.

Theory

Two binary links connected by a revolute pair may be replaced by a spring as shown in Figure 7. In the CPM, this configuration would appear as two adjacent binary links in one or more rows. If a spring is designated by a type number, for example 11, then the two numbers representing the binary links can be replaced by the single number representing the spring.

From Figure 6 it can be seen that links 3 and 4 and links 7 and 8 of chain 189 can be replaced by springs. This information is also apparent in Table VI which shows the CPM for chain 189. In the CPM, binary links 3 and 4 are seen to be adjacent in rows 4, 10, 14, and 15 and binary links 7 and 8 are seen to be adjacent in rows 1, 5, 11, and 15. Thus, from the CPM it would appear that chain 189 can yield two chains with one spring, one with links 3 and 4 replaced by a spring and one with links 7 and 8 replaced by a spring. It is evident from Figure 6 that these two chains are equivalent due to the symmetry of chain

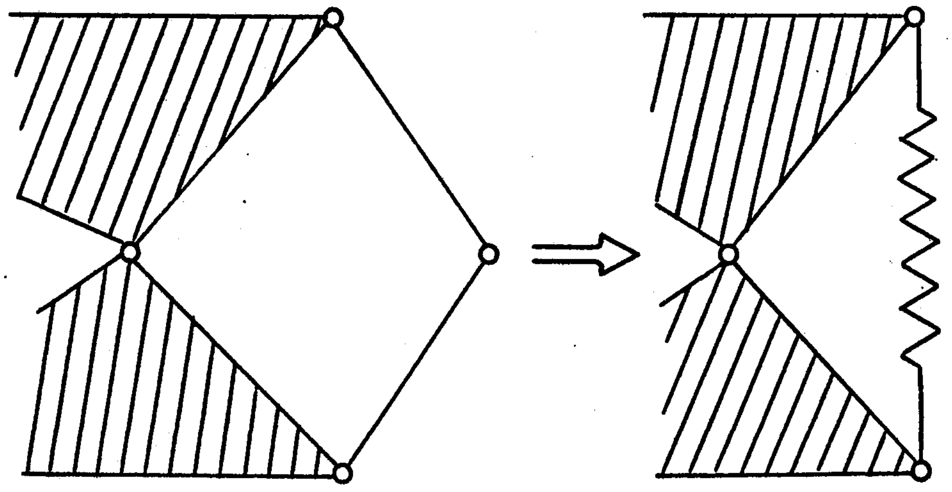


Figure 7. Two Binary Links Replaced by a Spring

189. However, a procedure must be devised to determine symmetry from the CPM.

If links 3 and 4 are replaced by a spring and an MPM formed as if the spring were a fixed link, the MPM would then represent all paths originating at the spring. Similarly, an MPM can be formed for the chain with links 7 and 8 replaced by a spring. The two MPM's would appear as shown in Tables XI and XII. The two MPM's are identical thus indicating that the two chains are equivalent.

To obtain a chain with two springs, both pairs of binary links are replaced with springs in the CPM. To compare the resulting chain with another chain containing two springs, the MPM's representing each spring could be compared. However, the same result may be obtained by using a modified MPM consisting of the combination of the individual MPM's for each spring. That is, the modified MPM will contain each row of the CPM which contains either spring or both springs. The modified MPM representing chain 189 is shown in Table XIII. Since the modified MPM does not represent the paths originating from a single element, the procedure for comparing two modified MPM's must be changed. Essentially, what is required is that, before deciding that two rows are not equivalent, a row containing two springs must be compared first with one spring as the first element and second, with the other spring as the first element. In Table XIII, only the last row contains two springs and the row would be the same regardless of which spring was used as the first element. However, for two rows, A and B, as shown in Table XIV, one of the rows must be cycled to obtain an equivalence. If row B is cycled as shown in the table, it becomes the same as row A read in reverse. Thus the two rows are equivalent.

TABLE XI

MPM FOR CHAIN 189 WITH LINKS 3 AND 4 REPLACED BY A SPRING

11	3	2	4	0	0	0	0	0	0	0	4
11	3	4	4	0	0	0	0	0	0	0	4
11	3	4	2	3	4	0	0	0	0	0	6
11	3	4	2	2	3	4	0	0	0	0	7

TABLE XII

MPM FOR CHAIN 189 WITH LINKS 7 AND 8 REPLACED BY A SPRING

11	3	2	4	0	0	0	0	0	0	0	4
11	3	4	4	0	0	0	0	0	0	0	4
11	3	4	2	3	4	0	0	0	0	0	6
11	3	4	2	2	3	4	0	0	0	0	7

TABLE XIII

MODIFIED MPM FOR CHAIN 189 WITH TWO SPRINGS

11	3	2	4	0	0	0	0	0	0	0	4
11	3	2	4	0	0	0	0	0	0	0	4
11	3	4	4	0	0	0	0	0	0	0	4
11	3	4	4	0	0	0	0	0	0	0	4
11	3	4	2	3	4	0	0	0	0	0	6
11	3	4	2	3	4	0	0	0	0	0	6
11	3	4	11	3	4	0	0	0	0	0	6

TABLE XIV

EQUIVALENCE OF ROWS WITH TWO SPRINGS

ROW A	11	3	4	11	4	3	3
ROW B	11	4	3	11	3	3	4
ROW B CYCLED	11	3	3	4	11	4	3

In the last chapter, it was mentioned that a certain portion of Subroutine MECCOM was used for cycling rows when required for the comparison of certain modified MPM's. That portion of MECCOM performs the cycling described above and permits comparison of modified MPM's representing any number of special elements. The number of times any row is cycled is limited to the number of identical special elements inserted in the chain. If a chain contained three springs, for example, a row would not be cycled more than three times.

Computer Program for Developing Chains With Springs

The computer program for developing chains with springs is shown in Appendix G along with a description of the variables used. The program performs the process of adding springs to a chain by proceeding as follows:

1. The loop matrix is read in and converted to the CPM. (Subroutine LOOP)

2. The CPM is examined and the pairs of links to be replaced by springs are assigned a number and stored in array LPAIR. (Subroutine SPRLOC)

3. The identification of the possible springs is printed out. (Subroutine ARRAY)

4. MPM's are developed for a single spring at each one of the locations determined in step 2 and are stored in array MX. (Subroutine SPRNG1)

5. The MPM's are compared and the equivalent chains identified. The results of the comparison are printed out and the unique chains, identified by the spring number assigned in step 2, are stored in array

JSTORE. (Subroutine MECCOM)

6. A second spring is added to the unique chains identified in step 5 and the modified MPM's formed. (Subroutine SPRING2)

7. The modified MPM's are compared and the unique chains identified as in step 5. (Subroutine MECCOM)

8. Additional springs are added and unique chains determined until the chain can accept no more springs. (Subroutines SPRNG3 and SPRNG4)

A sample print out for chain 189 with springs is shown in Table XV. A summary of results obtained from the ten-link chains is shown in Table XVI.

TABLE XV

SAMPLE OUTPUT FOR CHAIN 189 WITH SPRINGS

```
CHAIN  189
***********

IDENTIFICATION OF SPRINGS

   SPRING#       |  1   |  2   |
   ------------------------------
   LINK1, LINK2  |  3, 4 |  7, 8 |

CHAINS WITH ONE SPRING (IDENTIFIED BY THE SPRING NUMBER ABOVE)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

   EQUIVALENT CHAINS

         1= 2

   NUMBER OF UNIQUE CHAINS - 1

IDENTIFICATION OF CHAINS WITH TWO SPRINGS
+++++++++++++++++++++++++++++++++++++++++

   CHAIN# |     1      |
   --------------------
   SPRINGS|    1,2     |

   EQUIVALENT CHAINS

        NONE

   NUMBER OF UNIQUE CHAINS - 1
```

NOTE: "LINK1, LINK-2" indicates the links replaced by the spring.

TABLE XVI

TEN-LINK CHAINS WITH SPRINGS

Total number of 10-link chains with 1 spring	-	234
Total number of 10-link chains with 2 springs	-	83
Total number of 10-link chains with 3 springs	-	12
Total number of 10-link chains with 4 springs	-	1

CHAPTER VI

CHAINS WITH PULLEYS AND BELTS

Theory

A ternary link with two binary links connected to it can be converted to a pulley and belt as shown in Figure 8. In the CPM, such an arrangement of links would appear as a ternary link with a binary link on each side of it. If the ternary link is replaced by some type number designating a pulley, say 11 again, and the binary links are replaced by some type number representing a belt, say 12, the CPM will then represent the chain with one pulley. An MPM can be formed using the pulley as the starting element for each path and will serve as a means of comparing one chain with a pulley to a second chain with a pulley.

As an example of the above, consider link 1 of chain 189 (Figure 6). Since it is a ternary link connected to two binary links, it can be converted to a pulley and belt. This fact is reflected in row 1 of the CPM for chain 189 as shown in Table VI where 301 appears between 208 and 209. To develop the MPM with link 1 converted to a pulley, 301 must be converted to 11 everywhere it appears in the CPM. Similarly, links 208 and 209 must be converted to 12 wherever they appear. The MPM with the above conversions completed is shown in Table XVII.

Some kinematic chains may contain several pulleys just as some may contain several springs. However, some rules must be established to

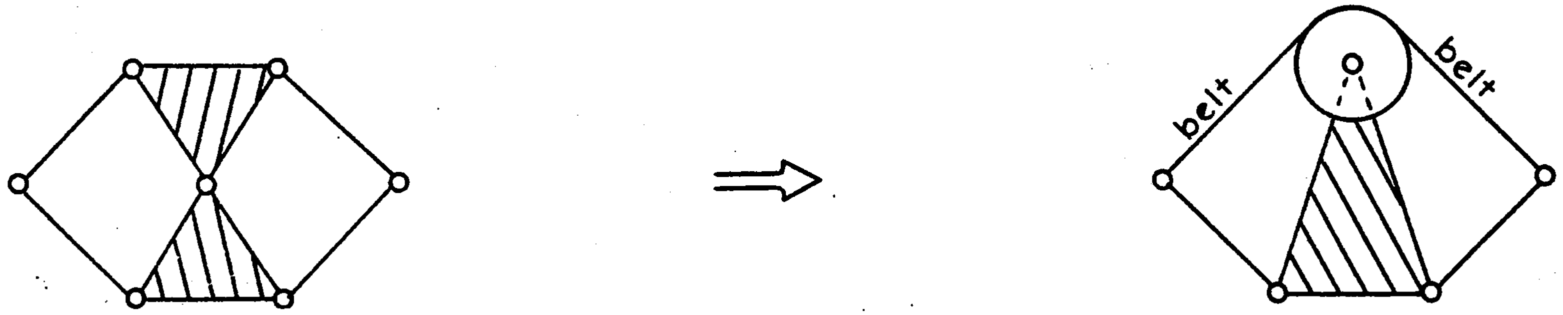


Figure 8. Ternary Link and Two Binary Links Converted to a Pulley and Belt

TABLE XVII

MPM FOR CHAIN 189 WITH LINK 1 CONVERTED TO A PULLEY

AND LINKS 8 AND 9 CONVERTED TO A BELT

11	12	4	2	12	0	0	0	0	0	0	5
11	4	4	12	0	0	0	0	0	0	0	4
11	4	4	2	12	0	0	0	0	0	0	5
11	4	2	3	4	12	0	0	0	0	0	6
11	4	2	3	4	2	12	0	0	0	0	7
11	4	2	2	3	4	12	0	0	0	0	7
11	4	2	2	3	4	2	12	0	0	0	8

prevent the chain from becoming a structure. The rules established in converting the ten-link chains to chains with pulleys are given below and illustrated in the referenced figures.

1. A ternary link joined to three binary links may be converted to three different pulley/belt combinations. However, no two of these can exist simultaneously. (Figure 9)

2. Two pulleys may share a common belt, but two binary links connected by a revolute pair may not be converted to a single belt. (Figure 10)

Computer Program for Converting Basic Kinematic Chains to Chains with Pulleys and Belts

Essentially, the same procedure is followed in developing chains with pulleys as was followed in the preceding chapter where springs were used. The possible locations for pulleys are determined and then, the pulleys are added one at a time to establish the chains. Subroutine PULLOC, which determines the pulley/belt locations, and Subroutine PUL3, which develops the modified MPM for chains with three pulleys, are shown in Appendix H. Subroutines for adding more or less pulleys follow the same procedures as PUL3. The entire program for adding pulleys is the same as the program for adding springs (Appendix G) except that Subroutine PULLOC would replace Subroutine SPRLOC and Subroutines PUL1, PUL2, PUL3, and PUL4 would replace Subroutines SPRNG1, SPRNG2, SPRNG3, and SPRNG4. All other subroutines are unchanged except for format statements. The computer print out for chain 189 with pulleys is shown in Table XVIII. A summary of the results obtained for the ten-link chains with pulleys appears in Table XIX.

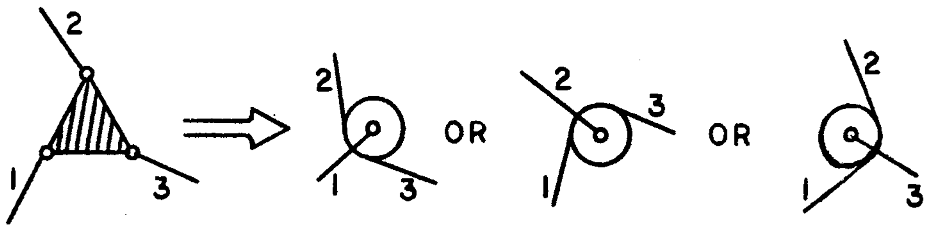


Figure 9. Ternary Link with 3 Binary Links Converted to Pulleys

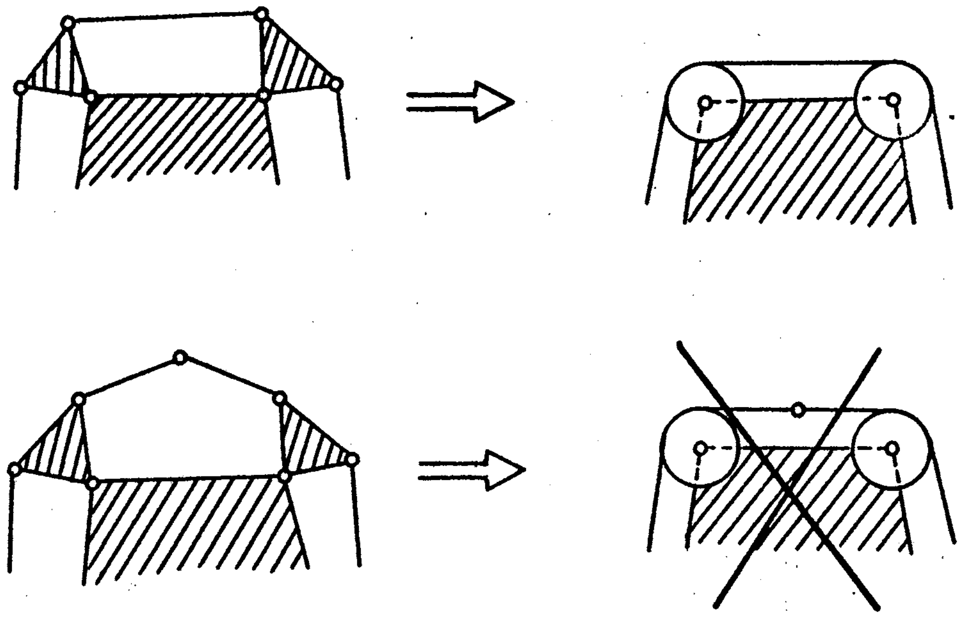


Figure 10. Ternary Links Sharing a Binary Link Converted to Pulleys and Belt

TABLE XVIII

SAMPLE OUTPUT FOR CHAIN 189 WITH PULLEYS

```
CHAIN  189
***********

IDENTIFICATION OF PULLEYS

    PULLEY#           |     1     |     2     |
    -----------------------------------------
    PULLEY(BELT,BELT)|  1( 8, 9)|  5( 4,10)|

CHAINS WITH ONE PULLEY (IDENTIFIED BY PULLEY NUMBER ABOVE)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    EQUIVALENT CHAINS

          1= 2

    NUMBER OF UNIQUE CHAINS - 1

IDENTIFICATION OF CHAINS WITH TWO PULLEYS
+++++++++++++++++++++++++++++++++++++++++

    CHAIN# |     1     |
    ------------------
    PULLEYS|   1,2     |

    EQUIVALENT CHAINS

         NONE

    NUMBER OF UNIQUE CHAINS - 1
```

NOTE: "PULLEY (BELT, BELT)" refers to the link replaced by a pulley and the two links replaced by a belt.

TABLE XIX

TEN-LINK CHAINS WITH PULLEYS

Total number of 10-link chains with 1 pulley	-	358
Total number of 10-link chains with 2 pulleys	-	240
Total number of 10-link chains with 3 pulleys	-	58
Total number of 10-link chains with 4 pulleys	-	7

CHAPTER VII

CHAINS WITH CAM PAIRS

A binary link and two revolute pairs may be converted to a cam pair as shown in Figure 11. In a basic kinematic chain, all links are joined by revolute pairs, therefore, each binary link may be replaced by a cam pair. If a cam pair is represented by a type number, say 11, the CPM may be converted to an MPM representing a chain with one cam pair simply by replacing a binary link with the number 11. Additional cam pairs may be added and modified MPM's formed subject to the restriction that two adjacent binary links cannot be simultaneously converted to cam pairs. This means that, in Figure 11, links 2 and 3 cannot both be converted to cam pairs.

The computer program for developing chains with cam pairs follows the same pattern as those for springs and for pulleys and belts. Subroutine CAMLOC, which locates the cam positions, and Subroutine CAM3, which corresponds to SPRNG3 and PUL3 in previous chapters, are shown in Appendix I. The computer print out for chain 189 with cam pairs is shown in Table XX. A summary of the results obtained for the ten-link chains with cam pairs is shown in Table XXI.

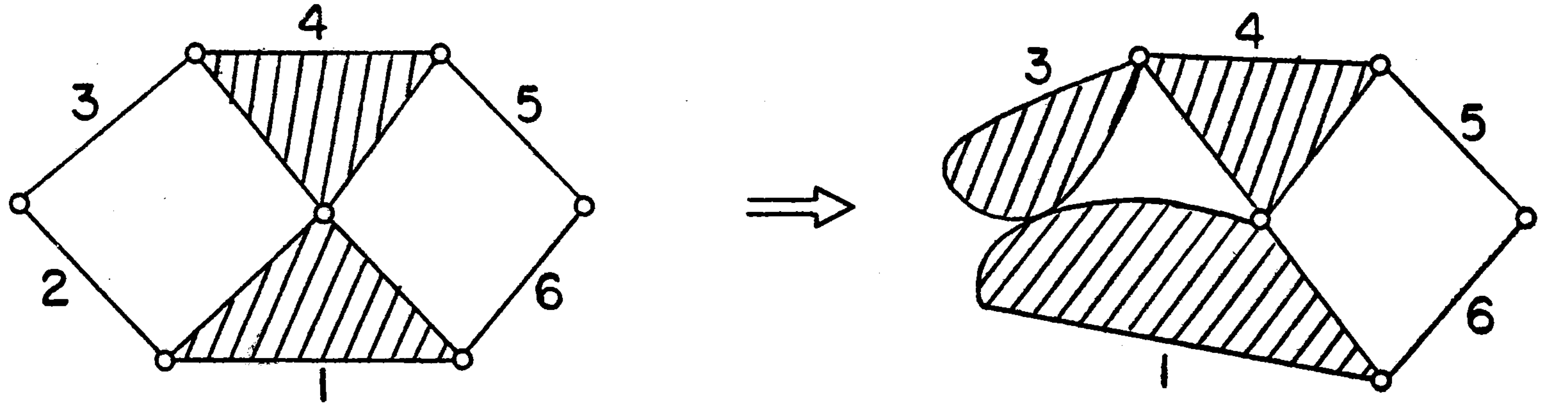


Figure 11. Binary Link Converted to a Cam Pair

TABLE XX

SAMPLE OUTPUT FOR CHAIN 189 WITH CAM PAIRS

```
CHAIN  189
***********

IDENTIFICATION OF CAM PAIRS

   CAM# |  1  |  2  |  3  |  4  |  5  |  6  |
   ------------------------------------------
   LINK |  3  |  4  |  7  |  8  |  9  | 10  |

CHAINS WITH ONE CAM PAIR (IDENTIFIED BY THE CAM NUMBER ABOVE)
*************************************************************


   EQUIVALENT CHAINS

         1= 3
         2= 4
         5= 6

   NUMBER OF UNIQUE CHAINS - 3

IDENTIFICATION OF CHAINS WITH TWO CAM PAIRS
*******************************************

   CHAIN# |    1    |    2    |    3    |    4    |    5    |
   ---------------------------------------------------------
   CAMS   |   1,3   |   1,4   |   1,5   |   1,6   |   2,3   |

   CHAIN# |    6    |    7    |    8    |    9    |
   -----------------------------------------------
   CAMS   |   2,4   |   2,5   |   2,6   |   5,6   |

   EQUIVALENT CHAINS

         2= 5

   NUMBER OF UNIQUE CHAINS - 8

IDENTIFICATION OF CHAINS WITH THREE CAM PAIRS
*********************************************

   CHAIN# |    1    |    2    |    3    |    4    |    5    |
   ---------------------------------------------------------
   CAMS   |  1,3,5  |  1,3,6  |  1,4,5  |  1,4,6  |  1,5,6  |

   CHAIN# |    6    |    7    |    8    |
   -------------------------------------
   CAMS   |  2,4,5  |  2,4,6  |  2,5,6  |
```

TABLE XX (Continued)

```
CHAIN  189  (CONTINUED)
************************

   EQUIVALENT CHAINS

         1= 2
         6= 7

   NUMBER OF UNIQUE CHAINS - 6

IDENTIFICATION OF CHAINS WITH FOUR CAM PAIRS
++++++++++++++++++++++++++++++++++++++++++++
```

CHAIN#	1	2	3
CAMS	1,3,5,6	1,4,5,6	2,4,5,6

```
   EQUIVALENT CHAINS

        NONE

   NUMBER OF UNIQUE CHAINS - 3

IDENTIFICATION OF CHAINS WITH FIVE CAM PAIRS
++++++++++++++++++++++++++++++++++++++++++++

   NO POSSIBLE CHAINS
```

TABLE XXI

TEN-LINK CHAINS WITH CAM PAIRS

Total number of 10-link chains with 1 cam pair	-	913
Total number of 10-link chains with 2 cam pairs	-	1661
Total number of 10-link chains with 3 cam pairs	-	1415
Total number of 10-link chains with 4 cam pairs	-	624
Total number of 10-link chains with 5 cam pairs	-	121
Total number of 10-link chains with 6 cam pairs	-	10

CHAPTER VIII

CHAINS WITH PRISM PAIRS

A prism pair may replace a revolute pair in a kinematic chain thus permitting linear motion rather than rotary motion. If a prism pair is represented by a specific type number, say 11 once more, converting the CPM to represent a chain with one prism pair becomes a simple matter of inserting the 11 between the two links to be joined by the prism pair. The MPM is then formed from each row of the CPM containing a prism pair. Chains with two prism pairs may also be formed in the same manner and a modified MPM used for comparison of those chains just as in previous chapters.

If more than two prism pairs are added to a chain, care must be taken to insure that the mobility of the chain is retained. If three prism pairs are inserted in a circuit or path with only four links, the mobility of the chain is no longer retained. Similarly, four prism pairs in a circuit consisting of five links will result in that circuit becoming underconstrained although constrained motion is achieved with three prism pairs (with non-parallel axes). Since every path represented in the CPM actually represents a complete circuit, the requirement for mobility reduces to the rule that no path may contain more than n-2 prism pairs where n is the number of links in the path. Thus, to insure that mobility of a chain is maintained when adding more than two prism pairs, each row of the CPM must be checked to determine if it will contain more than n-2 prism pairs.

The computer program for developing chains with prism pairs is similar to those developed in previous chapters. Subroutine JOINT locates all joints in the chain and identifies them according to the links incident to that joint. For example, the joint connecting links 2 and 3 in a chain would be identified by the notation 2-3.

Subroutine PRISM2 demonstrates the logic used in developing chains with multiple prism pairs. As was the case with double joints, the number of chains with more than two prism pairs which may be developed from a single basic kinematic chain becomes very large and therefore, the program used for the ten-link chains was limited to two prism pairs. For this reason, the test for mobility was not included in the program. Subroutines JOINT and PRISM2 are shown in Appendix J. The computer print out for chain 189 with prism pairs appears in Table XXII. A summary of the results obtained for all ten-link chains with prism pairs is shown in Table XXIII.

TABLE XXII

SAMPLE OUTPUT FOR CHAIN 189 WITH PRISM PAIRS

```
CHAIN  189
***********

IDENTIFICATION OF JOINTS

    JOINT#        |  1  |  2  |  3  |  4  |  5  |  6  |  7  |
    ------------------------------------------------------------
    LINK1, LINK2  | 1, 2 | 1, 8 | 1, 9 | 2, 3 | 2, 6 | 2,10 | 3, 4 |

    JOINT#        |  8  |  9  |  10  |  11  |  12  |  13  |
    ------------------------------------------------------
    LINK1, LINK2  | 4, 5 | 5, 6 | 5,10 | 6, 7 | 6, 9 | 7, 8 |

CHAINS WITH ONE PRISM PAIR (IDENTIFIED BY THE JOINT NUMBER ABOVE)
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


    EQUIVALENT CHAINS

            1= 9
            2= 8
            3=10
            4=11
            6=12
            7=13

    NUMBER OF UNIQUE CHAINS - 7

IDENTIFICATION OF CHAINS WITH TWO PRISM PAIRS
++++++++++++++++++++++++++++++++++++++++++++++

    CHAIN# |   1    |   2    |   3    |   4    |   5    |
    -----------------------------------------------------
    PULLEYS|  1,2   |  1,3   |  1,4   |  1,5   |  1,6   |

    CHAIN# |   6    |   7    |   8    |   9    |  10    |
    -----------------------------------------------------
    PULLEYS|  1,7   |  1,8   |  1,9   |  1,10  |  1,11  |

    CHAIN# |  11    |  12    |  13    |  14    |  15    |
    -----------------------------------------------------
    PULLEYS| 1,12   |  1,13  |  2,3   |  2,4   |  2,5   |

    CHAIN# |  16    |  17    |  18    |  19    |  20    |
    -----------------------------------------------------
    PULLEYS|  2,6   |  2,7   |  2,8   |  2,9   |  2,10  |
```

NOTE: The word "PULLEYS" should read "JOINTS" in this table.

TABLE XXII (Continued)

```
CHAIN  189  (CONTINUED)
************************
   CHAIN# |  21     |  22     |  23     |  24     |  25     |
   ----------------------------------------------------------
   PULLEYS|  2,11   |  2,12   |  2,13   |  3,4    |  3,5    |

   CHAIN# |  26     |  27     |  28     |  29     |  30     |
   ----------------------------------------------------------
   PULLEYS|  3,6    |  3,7    |  3,8    |  3,9    |  3,10   |

   CHAIN# |  31     |  32     |  33     |  34     |  35     |
   ----------------------------------------------------------
   PULLEYS|  3,11   |  3,12   |  3,13   |  4,5    |  4,6    |

   CHAIN# |  36     |  37     |  38     |  39     |  40     |
   ----------------------------------------------------------
   PULLEYS|  4,7    |  4,8    |  4,9    |  4,10   |  4,11   |

   CHAIN# |  41     |  42     |  43     |  44     |  45     |
   ----------------------------------------------------------
   PULLEYS|  4,12   |  4,13   |  5,6    |  5,7    |  5,8    |

   CHAIN# |  46     |  47     |  48     |  49     |  50     |
   ----------------------------------------------------------
   PULLEYS|  5,9    |  5,10   |  5,11   |  5,12   |  5,13   |

   CHAIN# |  51     |  52     |  53     |  54     |  55     |
   ----------------------------------------------------------
   PULLEYS|  6,7    |  6,8    |  6,9    |  6,10   |  6,11   |

   CHAIN# |  56     |  57     |  58     |  59     |  60     |
   ----------------------------------------------------------
   PULLEYS|  6,12   |  6,13   |  7,8    |  7,9    |  7,10   |

   CHAIN# |  61     |  62     |  63     |
   --------------------------------------
   PULLEYS|  7,11   |  7,12   |  7,13   |

   EQUIVALENT CHAINS

          4=46
          7=19
          9=29
         10=38
         11=53
         12=59
         15=45
         20=28
         21=37
```

TABLE XXII (Continued)

```
CHAIN  189  (CONTINUED)
************************
         22=52
         23=58
         25=47
         31=39
         33=60
         34=48
         41=55
         42=61
         43=49
         44=50
         57=62

   NUMBER OF UNIQUE CHAINS -43
```

TABLE XXIII

TEN-LINK CHAINS WITH PRISM PAIRS

Total number of 10-link chains with 1 prism pair	- 2312
Total number of 10-link chains with 2 prism pairs	- 12969

CHAPTER IX

CHAINS WITH DOUBLE JOINTS

Theory

Ternary and higher links may be converted to links with double joints by reducing a side between two revolute pairs to zero. Some examples of this are shown in Figure 12. When a double joint is added, care must be taken to insure that the chain retains a single degree of freedom. Rules ro prevent all or part of a chain from becoming a structure are as follows:

1. No loop may be reduced to only three links. (Figure 13)

2. No two paths, each containing only four links, may have three links in common. In Figure 14, this rule is demonstrated by loops 2 and 3. Initially, loop 2 contains 5 links, but, when a double joint is added such that links 7 and 5 are joined, loop 2 is reduced to four links. The portion of the chain composed of loops 2 and 3 becomes a structure which cannot move. (Although the figure shows an eight-link chain, this rule was derived from, and applied to, the ten-link chains).

In the discussion that follows, double joints will be identified by a set of numbers of the form 1(2,3). The first number represents the link which has one side reduced to zero and will be referred to as the base link. The two numbers in parentheses represent the two links joined to the base link at the double joint. For example, the double joint shown in Figure 12 (a) would be referred to as 4(3,5).

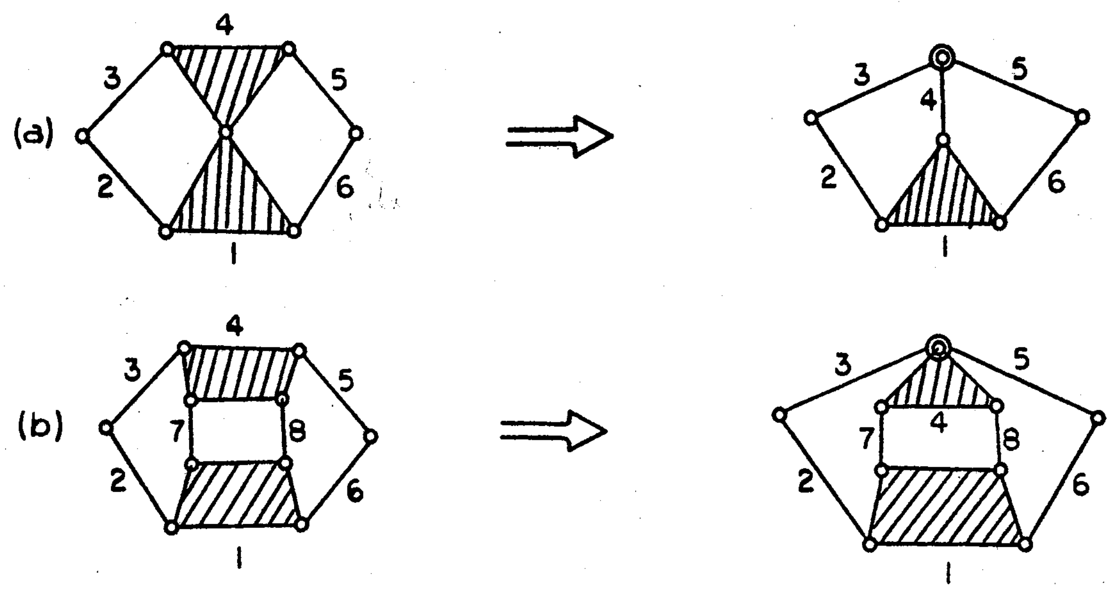


Figure 12. Addition of Double Joint to Basic Kinematic Chain

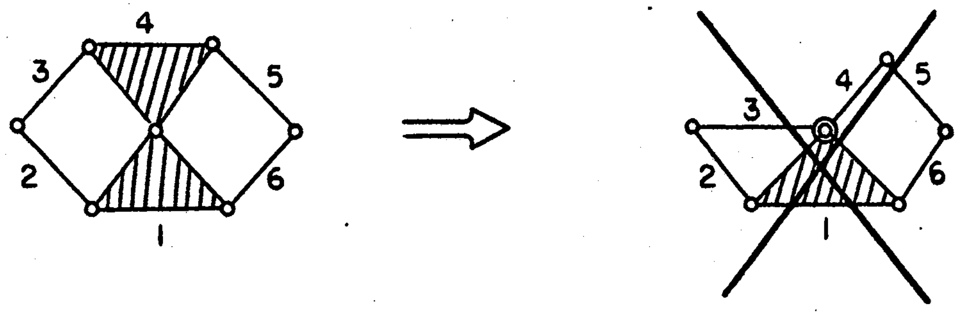


Figure 13. Demonstration of Rule 1 for Double Joints

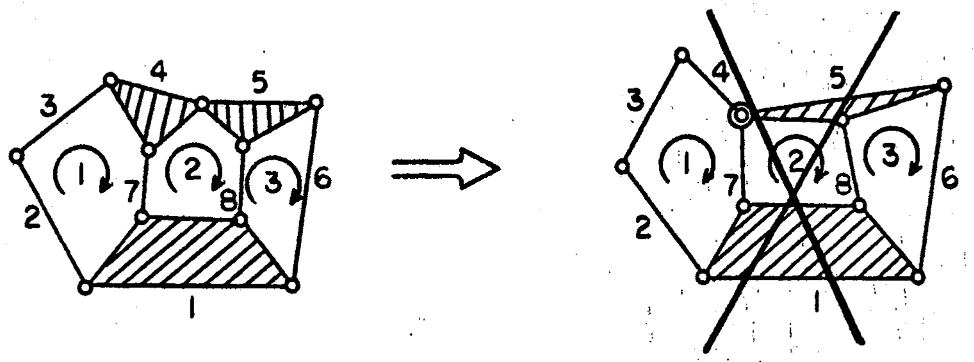


Figure 14. Demonstration of Rule 2 for Double Joints

Since double joints can be formed with any ternary or higher link, all possible double joints in a chain can be easily located by use of the CPM. In a row of the CPM, any link other than a binary link may be a base link, in which case, the two links adjacent to it become the other links incident at the double joint. For example, possible double joints found from the first two rows of the CPM for chain 189 (Table VI) are 1(8,9), 6(7,9), 1(2,9), 2(1,6), and 6(2,9). Some of these cannot actually be used due to rule 1 or 2 above. A double joint at position 1(2,9) would reduce row 2 of the CPM to a row with only three links (link 1 would effectively be removed from that path) thus violating rule 1. Positions 2(1,6) and 6(2,9) cannot be used for the same reason.

As in previous chapters, if a double joint is represented by a specific type number such as 11, an MPM may be developed which represents the chain with a double joint. In forming the MPM, the base link is replaced by the number 11 in those rows where it is flanked by the incident links. In other rows where the base link and only one of the incident links appear, the number 11 is inserted between the base link and the incident link. Table XXIV shows the MPM (with complete link identification numbers) for chain 189 with a double joint at position 1(8,9). The double joint is represented by 1100 to conform with the convention that the type number appears in the hundreds and thousands places.

Once the MPM has been established, chains with one double joint may be compared just as chains with other special elements. Additional double joints may also be added and modified MPM's formed. However, for each double joint added, the rules for mobility must be reapplied.

TABLE XXIV

MPM FOR CHAIN 189 WITH DOUBLE JOINT AT POSITION 1(8,9)

1100	209	406	207	208	0	0	0	0	0	0	5
1100	301	402	406	209	0	0	0	0	0	0	5
1100	301	402	406	207	208	0	0	0	0	0	6
1100	301	402	210	305	406	209	0	0	0	0	7
1100	301	402	210	305	406	207	208	0	0	0	8
1100	301	402	203	204	305	406	209	0	0	0	8
1100	301	402	203	204	305	406	207	208	0	0	9

In addition, a third restriction that arises when more than one double joint is used is that no two double joints may have two elements in common. That is, they may not share two links. For the chain shown in Figure 21 (b), joints 4(3,7) and 4(5,8) could exist simultaneously but 4(3,7) and 4(3,5) could not. The combination of position 4(3,7) and 4(3,8) would, in effect, create a triple joint which is not considered here.

Computer Program for Developing Chains With Double Joints

The basic program for developing chains with double joints follows the same pattern as previous programs. Subroutine DBLOC, which locates the positions of double joints, and subroutine DBL2, which constructs the modified MPM for chains with two double joints, appear in Appendix K. Subroutine DBL2 also tests for mobility by calling Subroutines LNKCNT and MOBCK. The computer print out for chain 189 with double joints appears in Table XXV.

The ten-link chains with more than two double joints were not developed due to the time required to perform the computations. In testing the program, it was found that a number of the ten-link chains would each produce in excess of 130 chains with three double joints. The computer time required to compare these chains runs into hours and was therefore not attempted. A summary of the results obtained for the ten-link chains with double joints appears in Table XXVI.

TABLE XXV

SAMPLE OUTPUT FOR CHAIN 189 WITH DOUBLE JOINTS

```
CHAIN  189
***********

IDENTIFICATION OF DOUBLE JOINTS

   JOINT#            |    1    |    2    |    3    |    4    |
   ------------------------------------------------------------
   BASE(LINK1,LINK2) | 1( 8, 9)| 1( 8, 2)| 2( 1,10)| 2( 1, 3)|

   JOINT#            |    5    |    6    |    7    |    8    |
   ------------------------------------------------------------
   BASE(LINK1,LINK2) | 2( 6, 3)| 2(10, 3)| 5(10, 4)| 5( 6, 4)|

   JOINT#            |    9    |   10    |   11    |   12    |
   ------------------------------------------------------------
   BASE(LINK1,LINK2) | 6( 9, 7)| 6( 9, 5)| 6( 7, 2)| 6( 7, 5)|

CHAINS WITH ONE DOUBLE JOINT (IDENTIFIED BY THE JOINT NUMBER ABOVE)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


   EQUIVALENT CHAINS

         1= 7
         2= 8
         3=10
         4=12
         5=11
         6= 9

   NUMBER OF UNIQUE CHAINS - 6

IDENTIFICATION OF CHAINS WITH TWO DOUBLE JOINTS
+++++++++++++++++++++++++++++++++++++++++++++++

   CHAIN# |   1     |   2     |   3     |   4     |   5     |
   ----------------------------------------------------------
   JOINTS |   1,3   |   1,4   |   1,5   |   1,6   |   1,7   |

   CHAIN# |   6     |   7     |   8     |   9     |  10     |
   ----------------------------------------------------------
   JOINTS |   1,8   |  1,10   |  1,11   |  1,12   |   2,5   |

   CHAIN# |  11     |  12     |  13     |  14     |  15     |
   ----------------------------------------------------------
   JOINTS |   2,6   |   2,7   |   2,8   |   2,9   |  2,10   |
```

TABLE XXV (Continued)

```
CHAIN  189  (CONTINUED)
************************
   CHAIN# |  16      |  17      |  18      |  19      |  20      |
   ---------------------------------------------------------------
   JOINTS | 2,12     |  3,5     |  3,7     |  3,8     |  3,9     |

   CHAIN# |  21      |  22      |  23      |  24      |  25      |
   ---------------------------------------------------------------
   JOINTS | 3,10     | 3,11     | 3,12     |  4,7     |  4,8     |

   CHAIN# |  26      |  27      |  28      |  29      |  30      |
   ---------------------------------------------------------------
   JOINTS |  4,9     | 4,10     | 4,11     | 4,12     |  5,7     |

   CHAIN# |  31      |  32      |  33      |  34      |  35      |
   ---------------------------------------------------------------
   JOINTS |  5,9     | 5,10     | 5,12     |  6,8     |  6,9     |

   CHAIN# |  36      |  37      |  38      |
   -------------------------------------
   JOINTS | 6,10     | 6,11     | 6,12     |

   EQUIVALENT CHAINS

          6=12
          7=18
          8=30
          9=24
         14=34
         15=19
         16=25
         20=36
         22=32
         23=27
         26=38
         28=33
         31=37

   NUMBER OF UNIQUE CHAINS -25
```

TABLE XXVI

TEN-LINK CHAINS WITH DOUBLE JOINTS

Total number of 10-link chains with 1 double joint	-	2039
Total number of 10-link chains with 2 double joints	-	7575

CHAPTER X

COMPARISON OF MECHANISMS DERIVED FROM CHAINS WITH SPECIAL ELEMENTS

Mechanisms derived from chains with special elements may be compared by use of the MPM. To demonstrate this and to further clarify the derivation of the CPM and MPM, three examples will be used. In these examples, the various special elements will be represented by type numbers as follows:

Pulley	- 12
Belt	- 13
Spring	- 14
Prism Pair	- 15
Double Joint	- 16
Cam Pair	- 17

Example 1

As the first example, chain number 4 from Appendix A will be used. Special elements will be added as follows:

1. Links 3 and 4 replaced by spring
2. Prism pair added at joint 8-9
3. Link 7 (and associated revolute pairs) replaced by a cam pair.

The resulting chain would appear as shown in Figure 15.

The loop matrix for the basic chain appears in Table XXVII. To obtain the CPM for the modified chain, the CPM for the basic chain may be formed and the special elements added. An alternate approach would

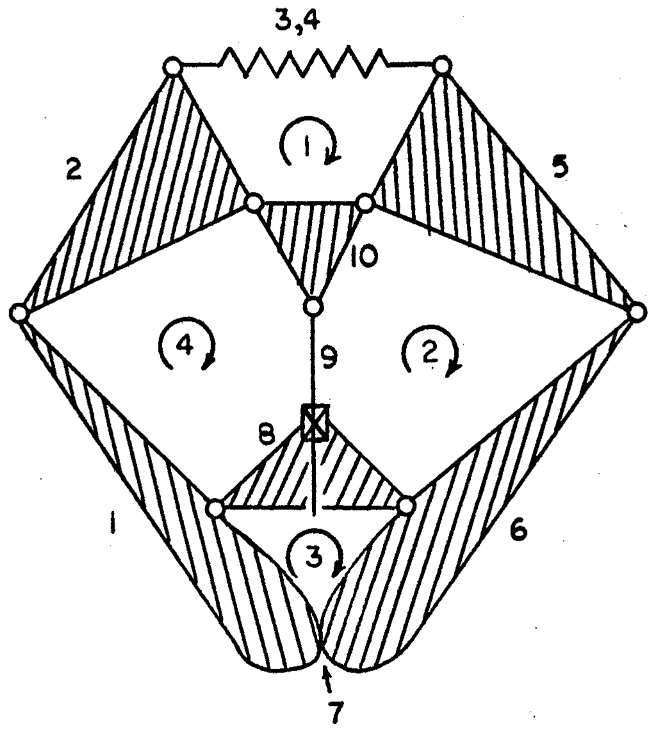


Figure 15. Chain 4 with Special Elements Added

TABLE XXVII

LOOP MATRIX FOR CHAIN 4

302	203	204	305	310	0	0	0	0	0	0	5
305	306	308	209	310	0	0	0	0	0	0	5
306	207	301	308	0	0	0	0	0	0	0	4
301	302	310	209	308	0	0	0	0	0	0	5

TABLE XXVIII

LOOP MATRIX FOR CHAIN 4 WITH SPECIAL ELEMENTS

302	1400	305	310	0	0	0	0	0	0	0	4
305	306	308	1500	209	310	0	0	0	0	0	6
306	1700	301	308	0	0	0	0	0	0	0	4
301	302	310	209	1500	308	0	0	0	0	0	6

TABLE XXIX

CPM FOR CHAIN 4 WITH SPECIAL ELEMENTS

302	1400	305	310	0	0	0	0	0	0	0	4
305	306	308	1500	209	310	0	0	0	0	0	6
306	1700	301	308	0	0	0	0	0	0	0	4
301	302	310	209	1500	308	0	0	0	0	0	6
302	1400	305	306	308	1500	209	310	0	0	0	8
0	0	0	0	0	0	0	0	0	0	0	0
302	1400	305	310	209	1500	308	301	0	0	0	8
305	306	1700	301	308	1500	209	310	0	0	0	8
305	306	308	301	302	310	0	0	0	0	0	6
306	1700	301	302	310	209	1500	308	0	0	0	8
302	1400	305	306	1700	301	308	1500	209	310	0	10
302	1400	305	306	308	301	0	0	0	0	0	6
302	1400	305	310	209	1500	308	306	1700	301	0	10
305	306	1700	301	302	310	0	0	0	0	0	6
302	1400	305	306	1700	301	0	0	0	0	0	6

be to form the loop matrix for the modified chain and then use it to form the CPM. The latter approach will be used here.

The loop matrix for the modified chain appears in Table XXVIII. The special elements have been added according to the rules established in previous chapters. This loop matrix is then used to form the CPM (Table XXIX). The next step is to form the MPM's for all mechanisms which can be derived from the modified chain.

The basic chain could have been converted to 10 mechanisms (1 mechanism per link), however, in the modified chain, links 3, 4, and 7 have been replaced by elements which cannot be fixed. Consequently, mechanisms 3, 4, and 7 cannot exist. This fact is obvious from the CPM since link numbers 3, 4, and 7 do not appear. Forming the MPM's for the remaining links yields the results shown in Table XXX.

To compare the mechanisms, each MPM must be compared with all other MPM's. Starting with MPM's 1 and 2, it is seen that each has 11 rows and therefore, further comparison is justified. Comparing row 1 of MPM #1 with row 1 of MPM #2 (both have 4 elements) shows that the two mechanisms are not equivalent since these rows are not equivalent and no other rows with 4 elements exist in CPM #2. Checking all other matrices shows that only MPM #6 has a 4 element row that is equivalent to row 1 of MPM #1. Further comparisons show the following equivalent rows:

MPM # 1		MPM #6
2	=	1
3	=	3
4	=	6
5	=	5
6	=	4
7	=	9
8	=	8
9	=	7
10	=	10
11	=	11

TABLE XXX

MPM'S FOR CHAIN 4 WITH SPECIAL ELEMENTS

```
MPM # 1
   3    3    3   17    0    0    0    0    0    0    0    4
   3    3    3    2   15    3    0    0    0    0    0    6
   3    3   14    3    3    2   15    3    0    0    0    8
   3    3   15    2    3    3    3   17    0    0    0    8
   3    3    3    3    3    3    0    0    0    0    0    6
   3    3    3    2   15    3    3   17    0    0    0    8
   3    3   15    2    3    3   14    3    3   17    0   10
   3    3   14    3    3    3    0    0    0    0    0    6
   3    3   14    3    3    2   15    3    3   17    0   10
   3    3    3    3    3   17    0    0    0    0    0    6
   3    3   14    3    3   17    0    0    0    0    0    6
............................................................
MPM # 2
   3   14    3    3    0    0    0    0    0    0    0    4
   3    3    2   15    3    3    0    0    0    0    0    6
   3   14    3    3    3   15    2    3    0    0    0    8
   3   14    3    3    2   15    3    3    0    0    0    8
   3    3    3    3    3    3    0    0    0    0    0    6
   3    3    2   15    3    3   17    3    0    0    0    8
   3   14    3    3   17    3    3   15    2    3    0   10
   3   14    3    3    3    3    0    0    0    0    0    6
   3   14    3    3    2   15    3    3   17    3    0   10
   3    3    3    3   17    3    0    0    0    0    0    6
   3   14    3    3   17    3    0    0    0    0    0    6
............................................................
MPM # 5
   3    3    3   14    0    0    0    0    0    0    0    4
   3    3    3   15    2    3    0    0    0    0    0    6
   3    3    3   15    2    3    3   14    0    0    0    8
   3    3    2   15    3    3    3   14    0    0    0    8
   3    3   17    3    3   15    2    3    0    0    0    8
   3    3    3    3    3    3    0    0    0    0    0    6
   3    3   17    3    3   15    2    3    3   14    0   10
   3    3    3    3    3   14    0    0    0    0    0    6
   3    3    2   15    3    3   17    3    3   14    0   10
   3    3   17    3    3    3    0    0    0    0    0    6
   3    3   17    3    3   14    0    0    0    0    0    6
............................................................
```

```
MPM # 6
   3    3   15    2    3    3    0    0    0    0    0    6
   3   17    3    3    0    0    0    0    0    0    0    4
   3    3   15    2    3    3   14    3    0    0    0    8
   3   17    3    3   15    2    3    3    0    0    0    8
   3    3    3    3    3    3    0    0    0    0    0    6
   3   17    3    3    3    2   15    3    0    0    0    8
   3   17    3    3   15    2    3    3   14    3    0   10
   3    3    3    3   14    3    0    0    0    0    0    6
   3   17    3    3   14    3    3    2   15    3    0   10
   3   17    3    3    3    3    0    0    0    0    0    6
   3   17    3    3   14    3    0    0    0    0    0    6
............................................................
MPM # 8
   3   15    2    3    3    3    0    0    0    0    0    6
   3    3   17    3    0    0    0    0    0    0    0    4
   3    3    3    3    2   15    0    0    0    0    0    6
   3   15    2    3    3   14    3    3    0    0    0    8
   3    3    3   14    3    3    2   15    0    0    0    8
   3   15    2    3    3    3   17    3    0    0    0    8
   3    3    3    3    3    3    0    0    0    0    0    6
   3    3   17    3    3    3    2   15    0    0    0    8
   3   15    2    3    3   14    3    3   17    3    0   10
   3    3    3   14    3    3    0    0    0    0    0    6
   3    3   17    3    3   14    3    3    2   15    0   10
............................................................
MPM # 9
   2    3    3    3    3   15    0    0    0    0    0    6
   2   15    3    3    3    3    0    0    0    0    0    6
   2    3    3   14    3    3    3   15    0    0    0    8
   2   15    3    3    3   14    3    3    0    0    0    8
   2    3    3    3   17    3    3   15    0    0    0    8
   2   15    3    3   17    3    3    3    0    0    0    8
   2    3    3   14    3    3   17    3    3   15    0   10
   2   15    3    3   17    3    3   14    3    3    0   10
............................................................
MPM #10
   3    3   14    3    0    0    0    0    0    0    0    4
   3    3    3    3   15    2    0    0    0    0    0    6
   3    2   15    3    3    3    0    0    0    0    0    6
   3    3   14    3    3    3   15    2    0    0    0    8
   3    2   15    3    3    3   14    3    0    0    0    8
   3    3    3   17    3    3   15    2    0    0    0    8
   3    3    3    3    3    3    0    0    0    0    0    6
   3    2   15    3    3   17    3    3    0    0    0    8
   3    3   14    3    3   17    3    3   15    2    0   10
   3    2   15    3    3   17    3    3   14    3    0   10
   3    3    3   17    3    3    0    0    0    0    0    6
............................................................
```

Since each row in MPM #1 is equivalent to a row in MPM #6, the two mechanisms, 1 and 6, are equivalent. Similarly, comparison of MPM's 2 and 5 shows that they are equivalent. No other equivalencies exist, hence, five unique mechanisms (1, 2, 8, 9, and 10) may be formed from chain 4 with the special elements as shown in Figure 15.

Example 2

The second example is based on chain number 48 from Appendix A. Special elements will be added as follows:

1. Link 4 replaced by a pulley
2. Links 3 and 8 replaced by a belt
3. Prism pairs added at joints 5-6 and 1-5

The resulting chain will appear as shown in Figure 16.

Since chain 48 is one of the chains with crossed links, the loops required to form the loop matrix are not easily identified. Therefore, the loop matrix for chain 48 as shown in Appendix B is used to form the loop matrix with special elements added (Table XXXI). Note that the special elements have been assigned "link numbers." This has been done in order to differentiate between elements of the same type so that incorrect row transfers will not take place when forming the CPM.

The CPM (Table XXXII) is formed by combining various rows of the loop matrix as explained in Chapter III. The process of combining rows is especially useful in this case since visual determination of the required paths is quite difficult.

The MPM's for all mechanisms which may be formed from chain 48 with the special elements added are shown in Table XXXIII. Note that MPM's 3, 4, and 8 do not appear since the corresponding links have been

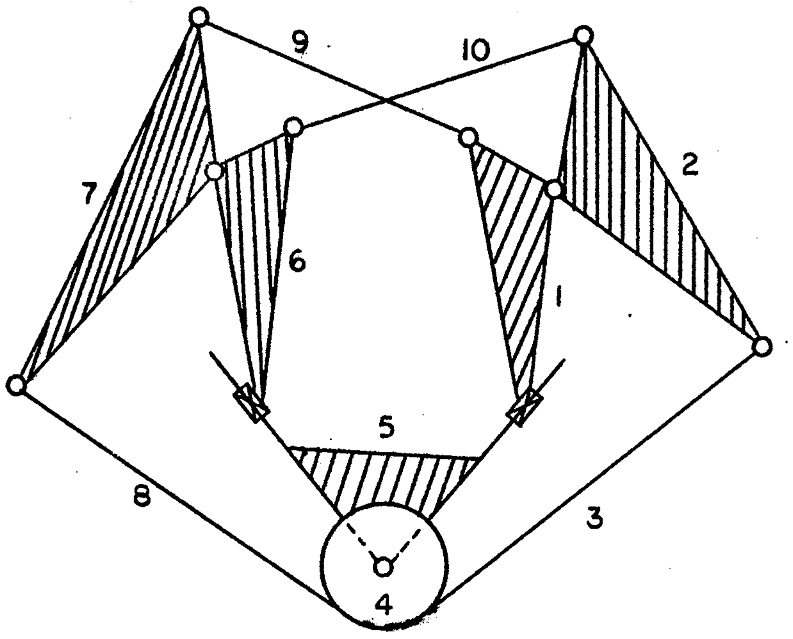


Figure 16. Chain 48 with Special Elements Added

TABLE XXXI

LOOP MATRIX FOR CHAIN 48 WITH SPECIAL ELEMENTS

301	1514	305	1515	306	210	302	0	0	0	0	7
301	302	1312	1211	305	1514	0	0	0	0	0	6
1211	1313	307	306	1515	305	0	0	0	0	0	6
301	1514	305	1515	306	307	209	0	0	0	0	7

TABLE XXXII

CPM FOR CHAIN 48 WITH SPECIAL ELEMENTS

301	1514	305	1515	306	210	302	0	0	0	0	0	0	7
301	302	1312	1211	305	1514	0	0	0	0	0	0	0	6
1211	1313	307	306	1515	305	0	0	0	0	0	0	0	6
301	1514	305	1515	306	307	209	0	0	0	0	0	0	7
305	1515	306	210	302	1312	1211	0	0	0	0	0	0	7
301	1514	305	1211	1313	307	306	210	302	0	0	0	0	9
0	0	0	0	0	0	0	0	0	0	0	0	0	0
301	302	1312	1211	1313	307	306	1515	305	1514	0	0	0	10
301	302	1312	1211	305	1515	306	307	209	0	0	0	0	9
1211	1313	307	209	301	1514	305	0	0	0	0	0	0	7
306	210	302	1312	1211	1313	307	0	0	0	0	0	0	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
301	302	1312	1211	1313	307	209	0	0	0	0	0	0	7
305	1515	306	210	302	1312	1211	1313	307	209	301	1514	0	12

TABLE XXXIII

MPM'S FOR CHAIN 48 WITH SPECIAL ELEMENTS

```
MPM # 1
     3   15    3   15    3    2    3    0    0    0    0    0    0    7
     3    3   13   12    3   15    0    0    0    0    0    0    0    6
     3   15    3   15    3    3    2    0    0    0    0    0    0    7
     3   15    3   12   13    3    3    2    3    0    0    0    0    9
     3    3   13   12   13    3    3   15    3   15    0    0    0   10
     3    3   13   12    3   15    3    3    2    0    0    0    0    9
     3   15    3   12   13    3    2    0    0    0    0    0    0    7
     3    3   13   12   13    3    2    0    0    0    0    0    0    7
     3   15    3   15    3    2    3   13   12   13    3    2    0   12
..........................................................................
MPM # 2
     3    3   15    3   15    3    2    0    0    0    0    0    0    7
     3   13   12    3   15    3    0    0    0    0    0    0    0    6
     3   13   12    3   15    3    2    0    0    0    0    0    0    7
     3    3   15    3   12   13    3    3    2    0    0    0    0    9
     3   13   12   13    3    3   15    3   15    3    0    0    0   10
     3   13   12    3   15    3    3    2    3    0    0    0    0    9
     3   13   12   13    3    3    2    0    0    0    0    0    0    7
     3   13   12   13    3    2    3    0    0    0    0    0    0    7
     3   13   12   13    3    2    3   15    3   15    3    2    0   12
..........................................................................
MPM # 5
     3   15    3    2    3    3   15    0    0    0    0    0    0    7
     3   15    3    3   13   12    0    0    0    0    0    0    0    6
     3   12   13    3    3   15    0    0    0    0    0    0    0    6
     3   15    3    3    2    3   15    0    0    0    0    0    0    7
     3   15    3    2    3   13   12    0    0    0    0    0    0    7
     3   12   13    3    3    2    3    3   15    0    0    0    0    9
     3   15    3    3   13   12   13    3    3   15    0    0    0   10
     3   15    3    3    2    3    3   13   12    0    0    0    0    9
     3   12   13    3    2    3   15    0    0    0    0    0    0    7
     3   15    3    2    3   13   12   13    3    2    3   15    0   12
..........................................................................
MPM # 6
     3    2    3    3   15    3   15    0    0    0    0    0    0    7
     3   15    3   12   13    3    0    0    0    0    0    0    0    6
     3    3    2    3   15    3   15    0    0    0    0    0    0    7
     3    2    3   13   12    3   15    0    0    0    0    0    0    7
     3    2    3    3   15    3   12   13    3    0    0    0    0    9
     3   15    3   15    3    3   13   12   13    3    0    0    0   10
     3    3    2    3    3   13   12    3   15    0    0    0    0    9
     3    2    3   13   12   13    3    0    0    0    0    0    0    7
     3    2    3   13   12   13    3    2    3   15    3   15    0   12
..........................................................................
```

```
MPM # 7
     3    3   15    3   12   13    0    0    0    0    0    0    0    6
     2    2    3   15    3   15    3    0    0    0    0    0    0    7
     3    3    2    3    3   15    3   12   13    0    0    0    0    9
     3    3   15    3   15    3    3   13   12   13    0    0    0   10
     3    2    3    3   13   12    3   15    3    0    0    0    0    9
     3    2    3   15    3   12   13    0    0    0    0    0    0    7
     3    3    2    3   13   12   13    0    0    0    0    0    0    7
     3    2    3    3   13   12   13    0    0    0    0    0    0    7
     3    2    3   15    3   15    3    2    3   13   12   13    0   12
..........................................................................
MPM # 9
     2    3   15    3   15    3    3    0    0    0    0    0    0    7
     2    3    3   13   12    3   15    3    3    0    0    0    0    9
     2    3   15    3   12   13    3    0    0    0    0    0    0    7
     2    3    3   13   12   13    3    0    0    0    0    0    0    7
     2    3   15    3   15    3    2    3   13   12   13    3    0   12
..........................................................................
MPM #10
     2    3    3   15    3   15    3    0    0    0    0    0    0    7
     2    3   13   12    3   15    3    0    0    0    0    0    0    7
     2    3    3   15    3   12   13    3    3    0    0    0    0    9
     2    3   13   12   13    3    3    0    0    0    0    0    0    7
     2    3   13   12   13    3    2    3   15    3   15    3    0   12
..........................................................................
```

replaced by elements which may not be fixed.

Comparison of the MPM's will show that mechanisms 1 and 6 are equivalent, mechanisms 2 and 7 are equivalent, and mechanisms 9 and 10 are equivalent. Thus chain 48 with special elements as shown in Figure 16, may be converted into only four unique mechanisms, mechanisms 1, 2, 5, and 9.

Example 3

The third example will be based on chain 206 from Appendix A. The following special elements will be added:

1. Link 5 replaced by a pulley
2. Links 4 and 6 replaced by a belt
3. Links 8 and 9 replaced by a spring
4. Prism pair placed at joint 3-10
5. Double joint placed at position 3(2,4)

The chain resulting from the above additions would appear as shown in Figure 17. Note that the pulley need not be circular.

The loop matrix for chain 206 without the special elements added is shown in Table XXXIV. As in the previous examples, the special elements are added to form the loop matrix shown in Table XXXV. This loop matrix is then used to generate the CPM in Table XXXVI. The MPM's appear in Table XXXVII and, after comparison, show that all five mechanisms are unique.

The previous examples demonstrate the capabilities of the path matrix approach in describing and comparing kinematic chains with all types of special elements. The following chapter will show how the CPM can be applied to the development of gear trains.

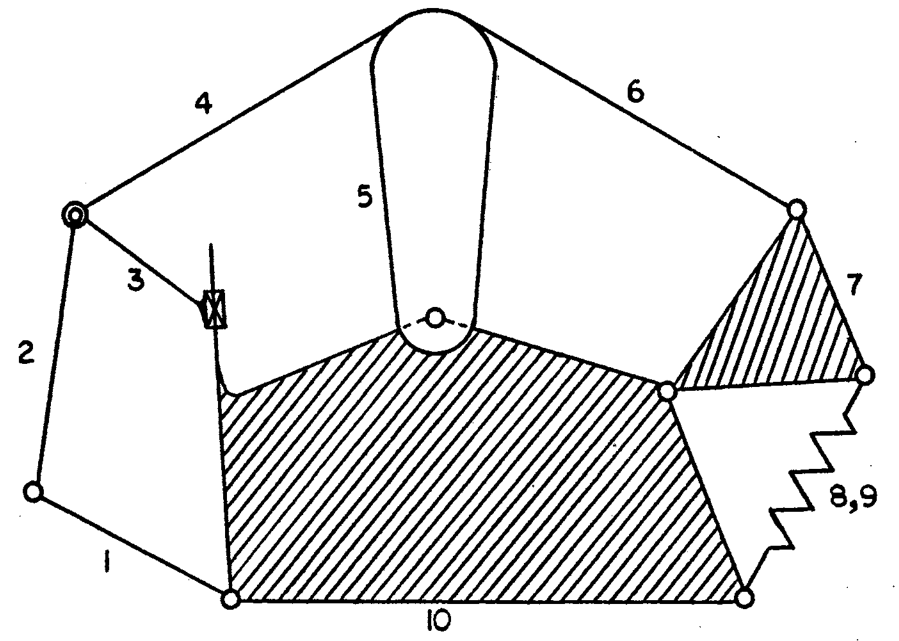


Figure 17. Chain 206 with Special Elements Added

TABLE XXXIV

LOOP MATRIX FOR CHAIN 206

201	202	303	510	0	0	0	0	0	0	0	4
510	303	204	305	0	0	0	0	0	0	0	4
305	206	307	510	0	0	0	0	0	0	0	4
307	208	209	510	0	0	0	0	0	0	0	4

TABLE XXXV

LOOP MATRIX FOR CHAIN 206 WITH SPECIAL ELEMENTS

201	202	1616	303	1515	510	0	0	0	0	0	6
510	1515	303	1616	1312	1211	0	0	0	0	0	6
1211	1313	307	510	0	0	0	0	0	0	0	4
307	1414	510	0	0	0	0	0	0	0	0	3

TABLE XXXVI

CPM FOR CHAIN 206 WITH SPECIAL ELEMENTS

201	202	1616	303	1515	510	0	0	0	0	0	6
510	1515	303	1616	1312	1211	0	0	0	0	0	6
1211	1313	307	510	0	0	0	0	0	0	0	4
307	1414	510	0	0	0	0	0	0	0	0	3
201	202	1616	1312	1211	510	0	0	0	0	0	6
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
510	1515	303	1616	1312	1211	1313	307	0	0	0	8
0	0	0	0	0	0	0	0	0	0	0	0
1211	1313	307	1414	510	0	0	0	0	0	0	5
201	202	1616	1312	1211	1313	307	510	0	0	0	8
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
510	1515	303	1616	1312	1211	1313	307	1414	0	0	9
201	202	1616	1312	1211	1313	307	1414	510	0	0	9

TABLE XXXVII

MPM'S FOR CHAIN 206 WITH SPECIAL ELEMENTS

MPM # 1												
	2	2	16	3	15	5	0	0	0	0	0	6
	2	2	16	13	12	5	0	0	0	0	0	6
	2	2	16	13	12	13	3	5	0	0	0	8
	2	2	16	13	12	13	3	14	5	0	0	9
MPM # 2												
	2	16	3	15	5	2	0	0	0	0	0	6
	2	16	13	12	5	2	0	0	0	0	0	6
	2	16	13	12	13	3	5	2	0	0	0	8
	2	16	13	12	13	3	14	5	2	0	0	9
MPM # 3												
	3	15	5	2	2	16	0	0	0	0	0	6
	3	16	13	12	5	15	0	0	0	0	0	6
	3	16	13	12	13	3	5	15	0	0	0	8
	3	16	13	12	13	3	14	5	15	0	0	9
MPM # 7												
	3	5	12	13	0	0	0	0	0	0	0	4
	3	14	5	0	0	0	0	0	0	0	0	3
	3	5	15	3	16	13	12	13	0	0	0	8
	3	14	5	12	13	0	0	0	0	0	0	5
	3	5	2	2	16	13	12	13	0	0	0	8
	3	14	5	15	3	16	13	12	13	0	0	9
	3	14	5	2	2	16	13	12	13	0	0	9
MPM #10												
	5	2	2	16	3	15	0	0	0	0	0	6
	5	15	3	16	13	12	0	0	0	0	0	6
	5	12	13	3	0	0	0	0	0	0	0	4
	5	3	14	0	0	0	0	0	0	0	0	3
	5	2	2	16	13	12	0	0	0	0	0	6
	5	15	3	16	13	12	13	3	0	0	0	8
	5	12	13	3	14	0	0	0	0	0	0	5
	5	2	2	16	13	12	13	3	0	0	0	8
	5	15	3	16	13	12	13	3	14	0	0	9
	5	2	2	16	13	12	13	3	14	0	0	9

CHAPTER XI

STRUCTURAL SYNTHESIS OF GEAR TRAINS

Johnson and Towfigh (10) have shown that certain kinematic chains may be converted to gear trains, both compound and epicyclic. In order to be convertible to a gear train, the chain must meet the following criteria:

1. Every link in the associated linkage must belong to at least one four-link path.

2. The chain must contain at least one multiple joint (double joint, triple joint, etc).

3. Any binary link incident to a multiple joint must also be connected to another binary link by a revolute pair.

Once the above criteria have been satisfied, the chain may be converted to an equivalent gear linkage. Gear trains may be formed from the gear linkage by fixing various elements with respect to ground, just as mechanisms are formed from kinematic chains by fixing various links.

To construct the equivalent gear linkage, each joint in the kinematic chain must be designated as either a rotation point (axis for a gear) or a base point (point on the edge of a gear). Certain joints are required to be either a base point or a rotation point according to the rules specified in Table XXXVIII. The remaining joints may be chosen as either a base point or a rotation point. However, as each one is chosen, the rules in Table XXXVIII must be applied to determine

TABLE XXXVIII

MANDATORY BASE POINTS AND ROTATION POINTS

I. Mandatory Rotation Points

a. Joints between higher links
b. Multiple joints

II. Mandatory Base Points

a. Joints between two binary links
b. Joint at other end of binary link with one rotation point
c. Both joints of a binary link connected to the link in b
d. Both joints on a binary link which connects two higher links

TABLE XXXIX

RULES APPLYING TO MANDATORY POINTS IN FIGURE 18

JOINT	TYPE POINT	RULE
1-6	R	Ia
3-6	R	Ia
3(4,9)	R	Ib
7-8	B	IIa
9-10	B	IIa
6-10	B	IIc
5-6	B	IIc
1-2	B	IId
2-3	B	IId

if other joints are affected by the choice. Where several choices exist, several gear linkages may be formed.

To illustrate this method, chain number 223 from Appendix A will be converted to a gear linkage, first, by observation of the chain itself, and second, by use of the CPM.

Figure 18 shows chain 223 with a double joint at position 3(4,9). The chain meets all criteria for conversion to a gear linkage. The mandatory rotation points and base points are shown in the figure. The rule pertaining to each mandatory point is shown in Table XXXIX. The only joints which remain free to choice are joints 6-7 and 1-8. If one of these is chosen to be a rotation point, the other automatically becomes a base point because of rule IIb. No symmetry exists for the chain as shown so that two unique gear linkages may be formed. The equivalent gear linkage formed by placing a rotation point at joint 6-7 is shown in Figure 19.

Gear trains may be formed by fixing any link containing one or more rotation points. Thus, gear trains could be formed by fixing any links of Figure 19 except links 2, 5, 8, and 10. Fixing links 4 or 9 would result in identical gear trains due to symmetry of the chain. Therefore, only five unique gear trains exist.

To apply the CPM to the development of gear trains, the following procedure would be followed:

1. Develop chains with one double joint.

2. Determine if each link appears in a path containing four links. This is done by simply taking the links in numerical order and determining if each one appears in at least one row of the CPM containing four links.

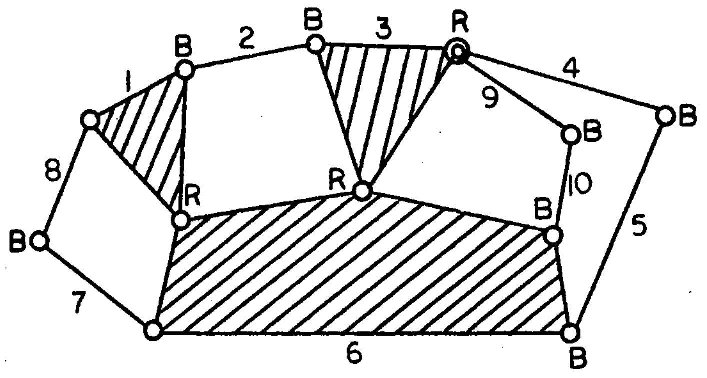


Figure 18. Chain 223 with Double Joint and Base and Rotation Points

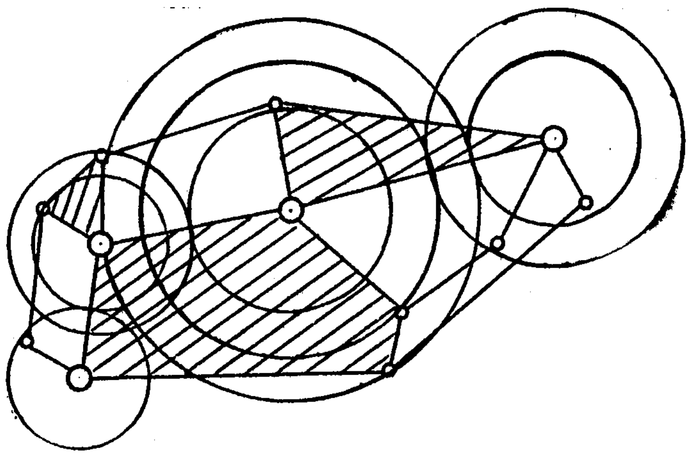


Figure 19. Equivalent Gear Linkage for Figure 18

3. Check each row in which a double joint appears to insure that if a binary link is next to the double joint, a second binary link is next to the first.

4. Identify the mandatory rotation points

a. Any joint where type numbers of the connected links are greater than 2.

b. The double joint

5. Identify the mandatory base points.

a. Any joint connecting binary links

b. Second joint on any binary link with a rotation point at the first joint.

c. Both joints of a binary link connected to the link in b.

d. Both joints of a binary link appearing between two higher links.

6. One at a time, make each remaining joint a rotation point and apply the rules for mandatory base points. The number of possible combinations resulting will yield the number of possible gear linkages. To determine whether or not two chains are equivalent, all that is required is to show that the optional joints converted to rotation points are equivalent. If, for example, joints A and B were found to be equivalent in the original chain (by methods used in Chapter VIII), then, placing a rotation point at joint A would result in the same gear linkage as that produced by placing a rotation point at joint B.

7. Gear trains may be formed by considering each link which contains a rotation point as being fixed. Equivalent trains may be found by determining which links are equivalent (by methods used in Chapter X).

Although a computer program has not yet been written to convert kinematic chains to gear trains, it appears that the CPM contains all the information necessary to perform the steps described above.

CHAPTER XII

SUMMARY AND CONCLUSIONS

The objective of this thesis has been to present the path matrix approach to the description and analysis of kinematic chains and mechanisms. It has been shown that a kinematic chain may be completely defined by the loop matrix which, in turn, is used to generate the Chain Path Matrix. Since the Chain Path Matrix numerically describes all paths or circuits in a kinematic chain, it serves as a powerful tool in evaluating, comparing, and modifying the chain with the aid of the high speed computer. The Mechanism Path Matrix, developed from the Chain Path Matrix, uniquely describes mechanisms derived from a kinematic chain and serves as a means of comparing chains and mechanisms, again using the computer.

The computer programs developed for use in analyzing the ten-link chains were initially tested on the six and eight-link chains. The results from these tests agreed with those obtained by Hain (2,3,4,5) through many months of work. These initial tests vividly illustrated the benefits to be gained by a computerized approach to the analysis of kinematic chains.

Once the programs had been proved on the six and eight-link chains, they were applied to the ten-link chains and catalogs of the following material were obtained (9):

1. Ten-link mechanisms

2. Ten-link chains with 1, 2, 3, and 4 springs

3. Ten-link chains with 1, 2, 3, and 4 pulleys

4. Ten-link chains with 1, 2, 3, 4, 5, and 6 cam pairs

5. Ten-link chains with 1 and 2 prism pairs

6. Ten-link chains with 1 and 2 double joints.

This material provides a great deal of data for further analysis by kinematicians. Moreover, it is felt that the possibilities of the path matrix approach have only been touched upon and that further development of the concept will prove to be of great value in the type synthesis and analysis of kinematic chains.

In addition to providing material for further research, the data obtained from the ten-link chains describes a tremendous number of kinematic chains, each of which can furnish a number of mechanisms. For the designer, this data provides many approaches to a given problem and may also serve to stimulate development along lines not previously considered.

To further develop the path matrix approach, it is recommended that research be conducted in the following areas:

1. Determination of the rules necessary for the development of loop matrices for chains with crossed links. This area will become of more importance in the analysis of chains with more than ten links.

2. Use of the path concept in the structural synthesis of plane kinematic chains. This area appears promising since higher order chains may be developed from chains of a given order by adding additional loops. If this can, in fact, be done, it should eliminate the problems encountered with the crossed linkages since the CPM could be developed directly without having to form the loop matrix.

3. Use of the path concept in the structural analysis and synthesis of spatial kinematic chains. If appropriate requirements for mobility are defined for paths containing various types of joints, it appears that the synthesis of multi-loop space mechanisms should be possible.

4. Development of a means of automatically sketching chains and mechanisms represented by a CPM. This would provide a much better reference for designers than the catalogs presently developed for the ten-link chains.

5. Research into the most advantageous means of cataloging the material obtained from analysis of higher order kinematic chains so as to be of greatest benefit to the designer, research kinematician, etc. As the data on the ten-link chains now stands, the chains may be classified according to the type of elements inserted in the basic chain. However, if chains with combinations of elements are developed, the classification of the chains will require some method which facilitates retrieval of the desired information.

6. Development of a computer program to synthesize gear trains based on kinematic chains. The procedure for this has been outlined in Chapter XI. The application of such a program to the ten-link chains would provide a valuable addition to the data already obtained.

A SELECTED BIBLIOGRAPHY

(1) Davies, Trevor H., and Frank Erskine Crossley. "Structural Analysis of Plane Linkages by Franke's Condensed Notation." Journal of Mechanisms, Vol. 1 (1966), 171-183.

(2) Hain, Kurt. Applied Kinematics. New York: McGraw-Hill, 1967.

(3) _______. "Systematik sechsgliedriger kinematischer Ketten." Maschinenmarkt. Würzburg, Vol. 74 (1968), No. 38, 717-723.

(4) _______. "Die zwangläufigen achtgliedrigen Getriege mit Einfach- und Mehrfachgelenken." Maschinenmarkt, Würzburg, Vol. 70 (1964), No. 64, 10-18.

(5) _______. "Höhre Winkelubertragungen in sechs- und achtgliedrigen, zwangläufigen Getrieben." Maschinenmarkt, Würzburg, Vol. 71 (1965), No. 3, 18-23.

(6) Woo, L. S. "Type Synthesis of Plane Linkages." Transactions of the ASME, Journal of Engineering for Industry, Vol. 89 (February, 1967), 159-172.

(7) Dobrjanskyj, L., and F. Freudenstein. "Some Applications of Graph Theory to the Structural Analysis of Mechanisms." Transactions of the ASME, Journal of Engineering for Industry, Vol. 89 (February, 1967), 153-158.

(8) Buchsbaum, F. "Structural Classification and Type Synthesis of Mechanisms with Multiple Elements." Eng. Sc. D. dissertation, Columbia University, 1967; Publication #65-15,479, University Microfilms, Inc., Ann Arbor, Michigan, 1968, 219 pp.

(9) Quist, F. F. A Catalog of Ten-Link Mechanisms and Ten-Link Chains with Springs, Pulleys, Cams, Prism Pairs, and Double Joints. Unpublished, Oklahoma State University, 1970.

(10) Johnson, R. C., and K. Towfigh. "Creative Design of Epicyclic Gear Trains Using Number Synthesis of Transactions of the ASME, Paper No. 66-MD-A.

(11) Soni, A. H., and Lee Harrisberger. Applied Mechanisms. Preliminary Edition, Oklahoma State University, 1969.

A P P E N D I C E S

APPENDIX A

LINE DRAWINGS OF THE 230 TEN-LINK CHAINS

The line drawings shown in Figure 20 were taken from the Appendix of reference 6. The links of each chain have been numbered and minor corrections have been made in some of the drawings.

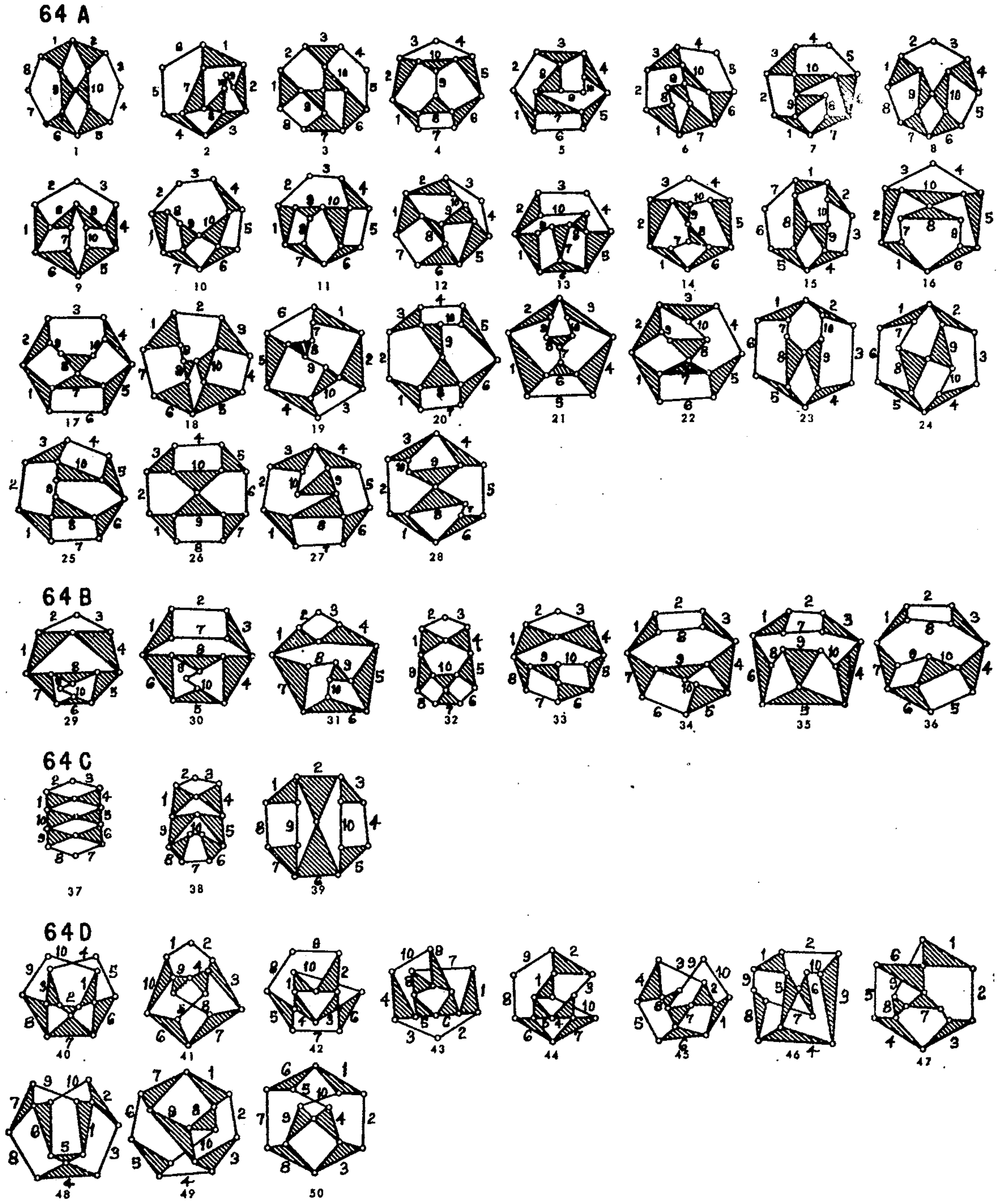


Figure 20. Line Drawings of Ten-Link Chains

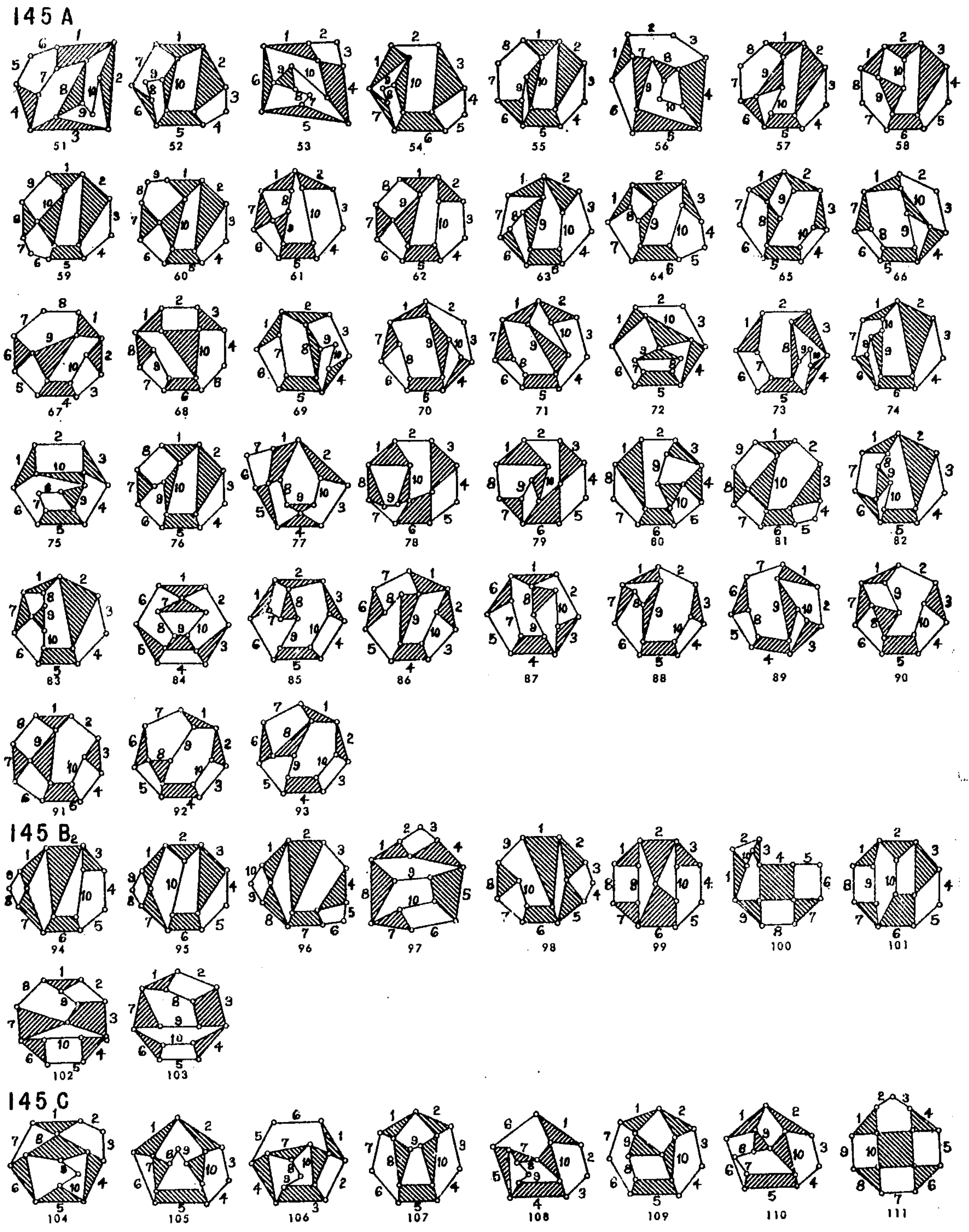


Figure 20 (Continued)

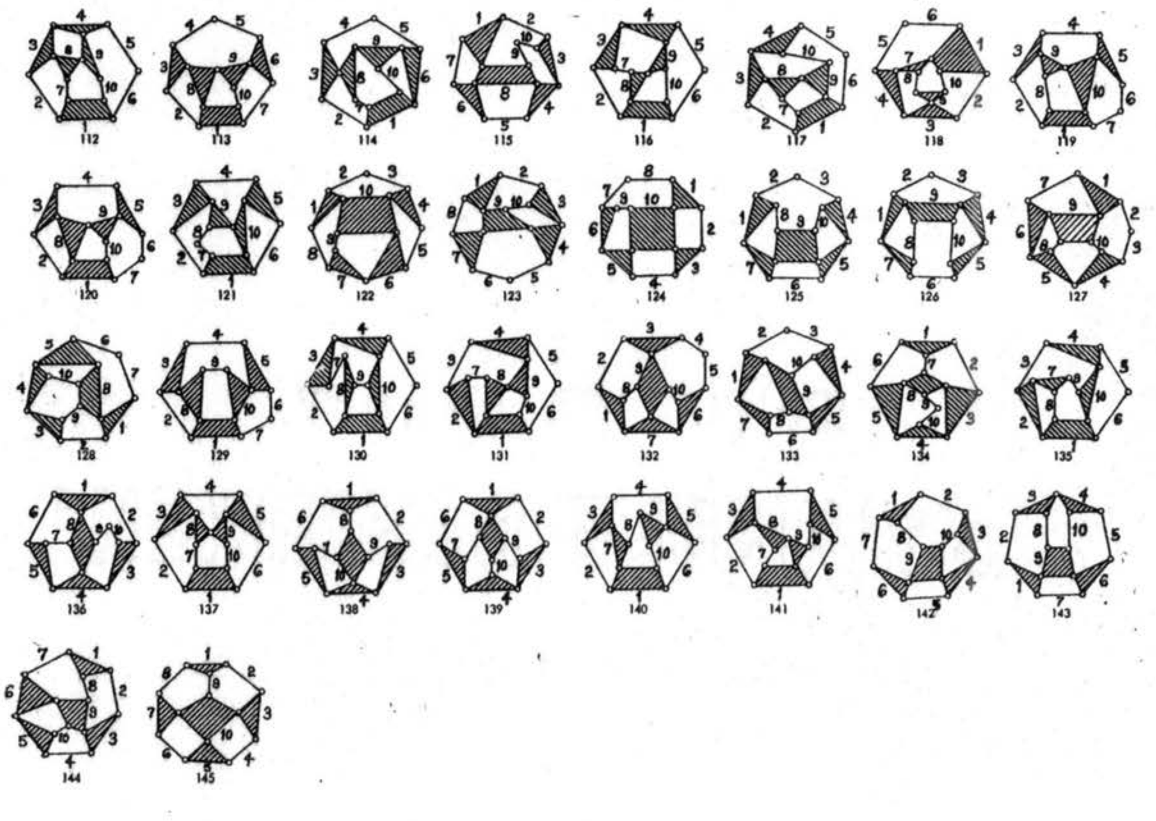


Figure 20 (Continued)

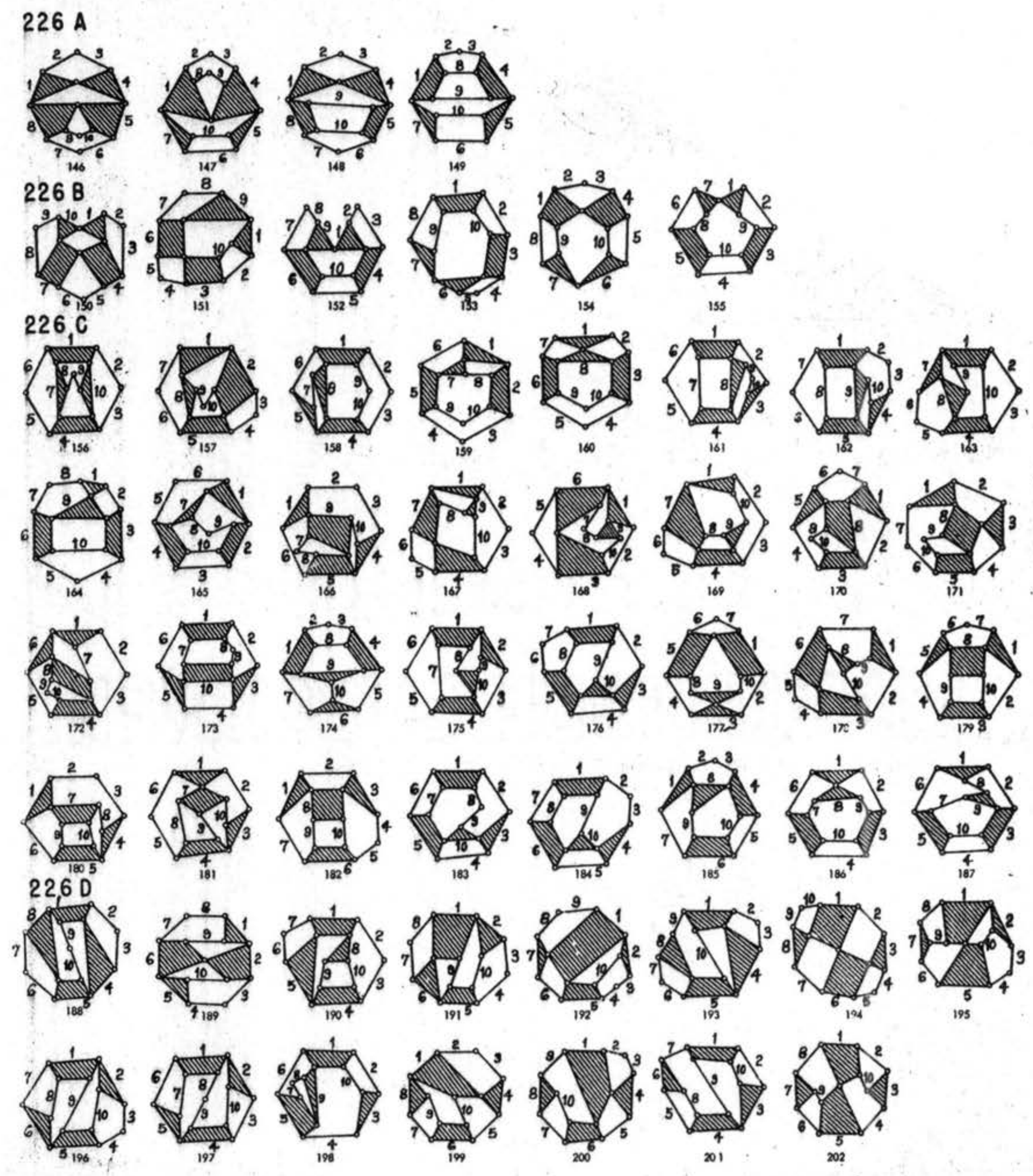


Figure 20 (Continued)

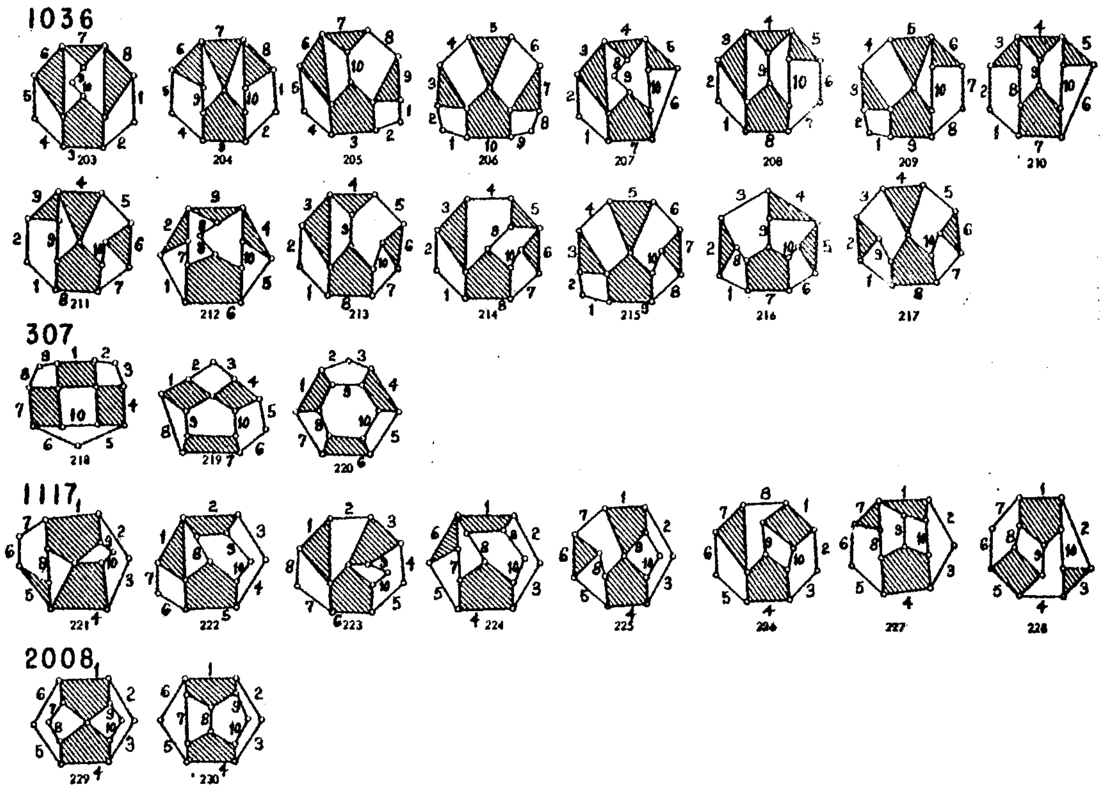


Figure 20 (Continued)

APPENDIX B

LOOP MATRICES FOR TEN-LINK CHAINS WITH CROSSED LINKS

Table XL shows the loop matrices used for the ten-link chains in Group 64D of Appendix A. As mentioned in Chapter III, these loop matrices were determined by trial and error.

TABLE XL

LOOP MATRICES FOR GROUP 64D OF APPENDIX A

CHAIN #40								
	109	110	301	306	307	308		
	301	302	307	306				
	302	303	308	307				
	303	104	105	306	307	308		
CHAIN #41								
	310	306	105	304	303	102	101	
	310	101	102	303	307	306		
	310	309	108	307	303	102	101	
	310	101	102	303	304	309		
CHAIN #42								
	301	302	303	306	110			
	301	110	306	107	305	108	109	302
	301	304	305	107	306	110		
	301	110	306	303	304			
CHAIN #43								
	301	306	305	304	103	102		
	301	102	103	304	110	309	306	
	310	107	308	309	110	304	103	102
	301	102	103	304	305	308	107	
CHAIN #44								
	301	302	103	304	305			
	301	305	306	108	109	302		
	301	110	307	306	305			
	301	305	304	307	110			
CHAIN #45								
	301	306	307	308	109	110		
	301	110	109	308	304	105	306	
	301	302	103	304	308	109	110	
	301	110	109	308	307	302		
CHAIN #46								
	301	102	303	304	305			
	304	303	110	306	305			
	304	305	306	107	308			
	301	305	304	308	109			

Note: A type number of 1 represents a binary link.

TABLE XL (Continued)

CHAIN #47							
	301	306	307	110	303	102	
	301	102	303	304	105	306	
	304	308	307	306	105		
	301	306	307	308	109		
CHAIN #48							
	301	305	306	110	302		
	301	302	103	304	305		
	304	108	307	306	305		
	301	305	306	307	109		
CHAIN #49							
	301	308	110	305	104	303	102
	301	102	303	104	305	106	307
	307	309	308	301			
	301	308	309	303	102		
CHAIN #50							
	301	110	309	304	303	102	
	301	102	303	304	105	306	
	306	107	308	309	110	301	
	301	110	309	308	303	102	

APPENDIX C

SUBROUTINE LOOP

Subroutine LOOP is used to develop the CPM from the loop matrix as described in Chapter III. The variables used in subroutine LOOP are shown in Table XLI. A flow chart for the rubroutine is shown in Figure 21 and the subroutine itself appears in Table XLII.

TABLE XLI

IDENTIFICATION OF VARIABLES USED IN SUBROUTINE LOOP

Array M	-	Chain Path Matrix (CPM)
NB	-	Number of Bars (Links) in Chain
NL	-	Number of Loops in Chain
NR	-	Number of Rows in CPM
NCOL	-	Number of Columns in CPM
NK	-	NR-NL (Number of rows formed by combining rows of Loop Matrix)
KM & LM	-	Primary row combinations used in forming rows NL+1 through NR-1 of the CPM
KS & LS	-	Secondary row combinations used in forming rows 11 through 14 (for 10-link chains) of the CPM
KT & LT	-	Row combinations required for row 15 (for 10-link chains) of the CPM

TABLE XLII

SUBROUTINE LOOP

```
FORTRAN IV G LEVEL 1, MOD 4            LOOP              DATE = 70191         20/49/09              PAGE 0001

                 SUBROUTINE LOOP
                 IMPLICIT INTEGER*2(I-N)
                 COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
                 COMMON/LOO/KM(11),LM(11),KS(11),LS(11),KT(5),LT(5)
            C
            C READ IN LOOPS
            C
                 DO 10 I=1,NL
                 READ(5,1)(M(I,J),J=1,NCCL)
               1 FORMAT(20I4)
              10 CONTINUE
                 DO 12 IA=1,NL
                 ISUM=0
                 DC 11 IB=1,NCOL
                 IF(M(IA,IB).GT.9000) M(IA,IB)=M(IA,IB)-9000
                 IF(M(IA,IB).NE.0) ISUM=ISUM+1
              11 CONTINUE
              12 M(IA,NCOL)=ISUM
                 DO 200 I=1,NK
                 IS=1
                 K=KM(I)
                 L=LM(I)
                 JL=1
              13 J=1
                 IF(I+NL.EQ.NR.AND.(M(K,1).EQ.0.OR.M(L,1).EQ.0)) GOTO 400
                 IF(I+NL.GE.NK.AND.(M(K,1).EQ.0.OR.M(L,1).EQ.0)) GOTO 300
            C
            C START WITH LOOP K
            C
            C CHECK TO SEE IF ELEMENT J EXISTS IN BOTH LOOPS
            C
              20 KN=M(K,NCOL)
                 LN=M(L,NCCL)
                 IF(LN.EQ.0) GOTO 100
              22 DO 21 IL=1,LN
                 IF(M(K,J).EQ.M(L,IL)) GCTC 30
              21 CONTINUE
                 M(I+NL,JL)=M(K,J)
                 J=J+1
                 JL=JL+1
                 IF(M(K,J).EQ.0) GOTO 100
                 GOTO 22
            C
            C CHECK ELEMENTS ON BOTH SIDES OF POSSIBLE TRANSFER ELEMENT
            C
              30 IF(IL.EQ.1) GOTO 31
                 IF(J.EQ.KN) GOTO 32
                 IF(M(K,J+1).EQ.M(L,IL-1)) GOTO 40
                 M(I+NL,JL)=M(K,J)
                 J=J+1
                 IF(M(K,J).EQ.0) GOTO 100
                 JL=JL+1
                 GOTO 22
              31 IF(J.EQ.KN) GOTO 33
                 IF(M(K,J+1).EQ.M(L,LN)) GOTO 40
                 M(I+NL,JL)=M(K,J)
                 J=J+1
                 IF(M(K,J).EQ.0) GOTO 100
```

TABLE XLII (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4          LOOP          DATE = 70191        20/49/09          PAGE 0002

      JL=JL+1
      GOTO 22
   32 IF(M(K,1).EQ.M(L,IL-1)) GOTO 40
      GOTO 100
   33 IF(M(K,1).EQ.M(L,LN)) GOTO 40
      GOTO 100
C
C NEXT ELEMENT IN FIRST LOOP IS COMMON TO BOTH LOOPS.  CHECK TO SEE IF LAST
C ELEMENT WAS COMMON.  IF IT WAS, PROCEED TO NEXT ELEMENT IN FIRST LOOP.  IF NOT
C TRANSFER TO SECOND LOOP.
C
   40 IF(J.EQ.1) GOTO 41
      IF(IL.EQ.LN) GOTO 42
      IF(M(K,J-1).NE.M(L,IL+1)) GOTO 50
      J=J+1
      IF(M(K,J).EQ.0) GOTO 100
      GOTO 22
   41 IF(IL.EQ.LN) GOTO 43
      IF(M(K,KN).NE.M(L,IL+1)) GOTO 50
      J=J+1
      IF(M(K,J).EQ.0) GOTO 100
      GOTO 22
   42 IF(M(K,J-1).NE.M(L,1)) GOTO 50
      J=J+1
      IF(M(K,J).EQ.0) GOTO 100
      GOTO 22
   43 IF(M(K,KN).NE.M(L,1)) GOTO 50
      J=J+1
      IF(M(K,J).EQ.0) GOTO 100
      GOTO 22
C
C TRANSFER TO SECOND LOOP
C
   50 M(I+NL,JL)=M(L,IL)
   51 JL=JL+1
      IL=IL+1
      IF(M(L,IL).EQ.0) IL=1
   52 DO 53 J=1,KN
      IF(M(L,IL).EQ.M(K,J)) GOTO 60
   53 CONTINUE
      M(I+NL,JL)=M(L,IL)
      GOTO 51
C
C CHECK ELEMENTS ON BOTH SIDES OF POSSIBLE TRANSFER ELEMENT
C
   60 IF(J.EQ.1) GOTO 61
      IF(IL.EQ.LN) GOTO 62
      IF(M(L,IL+1).EQ.M(K,J-1)) GOTO 70
      GOTO 100
   61 IF(IL.EQ.LN) GOTO 63
      IF(M(L,IL+1).EQ.M(K,KN)) GOTO 70
      GOTO 100
   62 IF(M(L,1).EQ.M(K,J-1)) GOTO 70
      GOTO 100
   63 IF(M(L,1).EQ.M(K,KN)) GOTO 70
      GOTO 100
C
C NEXT ELEMENT IN SECOND LOOP IS COMMON TO BOTH LOOPS.  TRANSFER BACK TO FIRST

FORTRAN IV G LEVEL 1, MOD 4          LOOP          DATE = 70191        20/49/09          PAGE 0003

C LOOP .
C
   70 IF(M(K,J).EQ.M(I+NL,1)) GOTO 110
      M(I+NL,JL)=M(K,J)
      J=J+1
      IF(M(K,J).EQ.0) J=1
      JL=JL+1
      GOTO 70
  300 K=KS(I)
      L=LS(I)
      GOTO 20
  400 K=KT(IS)
      L=LT(IS)
      IS=IS+1
      IF(IS.GT.5) GOTO 100
      GOTO 13
  100 JL=1
  110 DO 111 J=JL,NCOL
  111 M(I+NL,J)=0
      M(I+NL,NCOL)=JL-1
  200 CONTINUE
      RETURN
      END
```

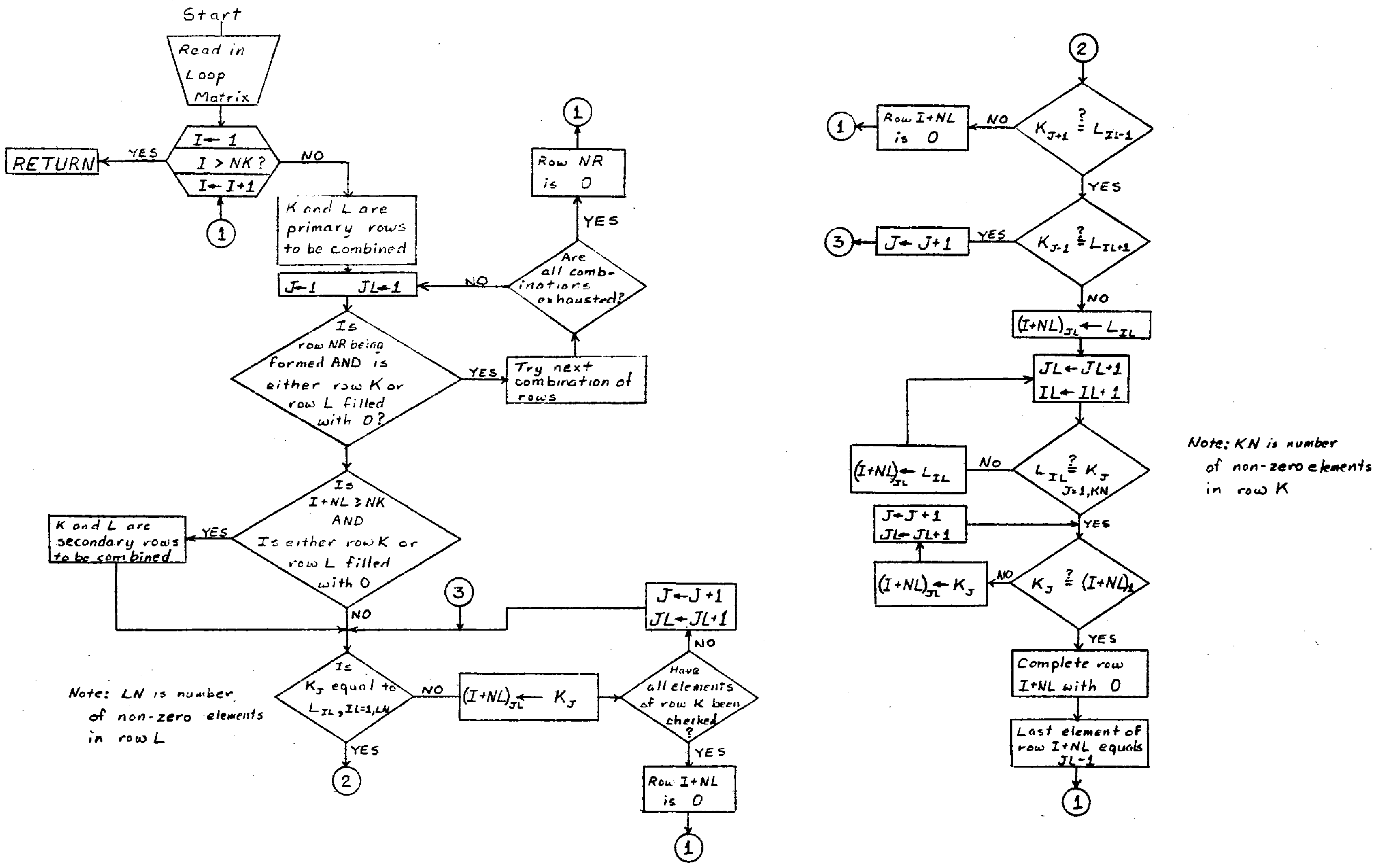

Start
Read in Loop Matrix
I ← 1
I > NK ?
I ← I+1
YES
RETURN
NO
1
K and L are primary rows to be combined
J ← 1 JL ← 1
Is row NR being formed AND is either row K or row L filled with 0?
YES
Try next combination of rows
Are all combinations exhausted?
NO
YES
Row NR is 0
1
Is I+NL ≥ NK AND Is either row K or row L filled with 0
YES
K and L are secondary rows to be combined
NO
3
J ← J+1 JL ← JL+1
Is K_J equal to L_IL, IL=1,LN
NO
(I+NL)_JL ← K_J
Have all elements of row K been checked?
NO
YES
Row I+NL is 0
1
YES
2
Note: LN is number of non-zero elements in row L
2
K_J+1 ?= L_IL-1
NO
Row I+NL is 0
1
YES
K_J-1 ?= L_IL+1
YES
J ← J+1
3
NO
(I+NL)_JL ← L_IL
JL ← JL+1 IL ← IL+1
L_IL ?= K_J J=1,KN
NO
(I+NL)_JL ← L_IL
YES
J ← J+1 JL ← JL+1
K_J ?= (I+NL)_JL
NO
(I+NL)_JL ← K_J
YES
Complete row I+NL with 0
Last element of row I+NL equals JL-1
1
Note: KN is number of non-zero elements in row K

Figure 21. Flow Chart for Subroutine LOOP

APPENDIX D

SUBROUTINE MECH

Subroutine MECH develops the MPM's for mechanisms formed from a kinematic chain as described in Chapter IV. Most of the variables used in MECH are the same as those used in Subroutine LOOP. The only variables used in LOOP are MY, which holds the CPM reduced to an array of link numbers, and MX, which holds the MPM's that are generated. The flow chart for Subroutine MECH appears in Figure 22 and the subroutine itself in Table XLIII.

TABLE XLIII

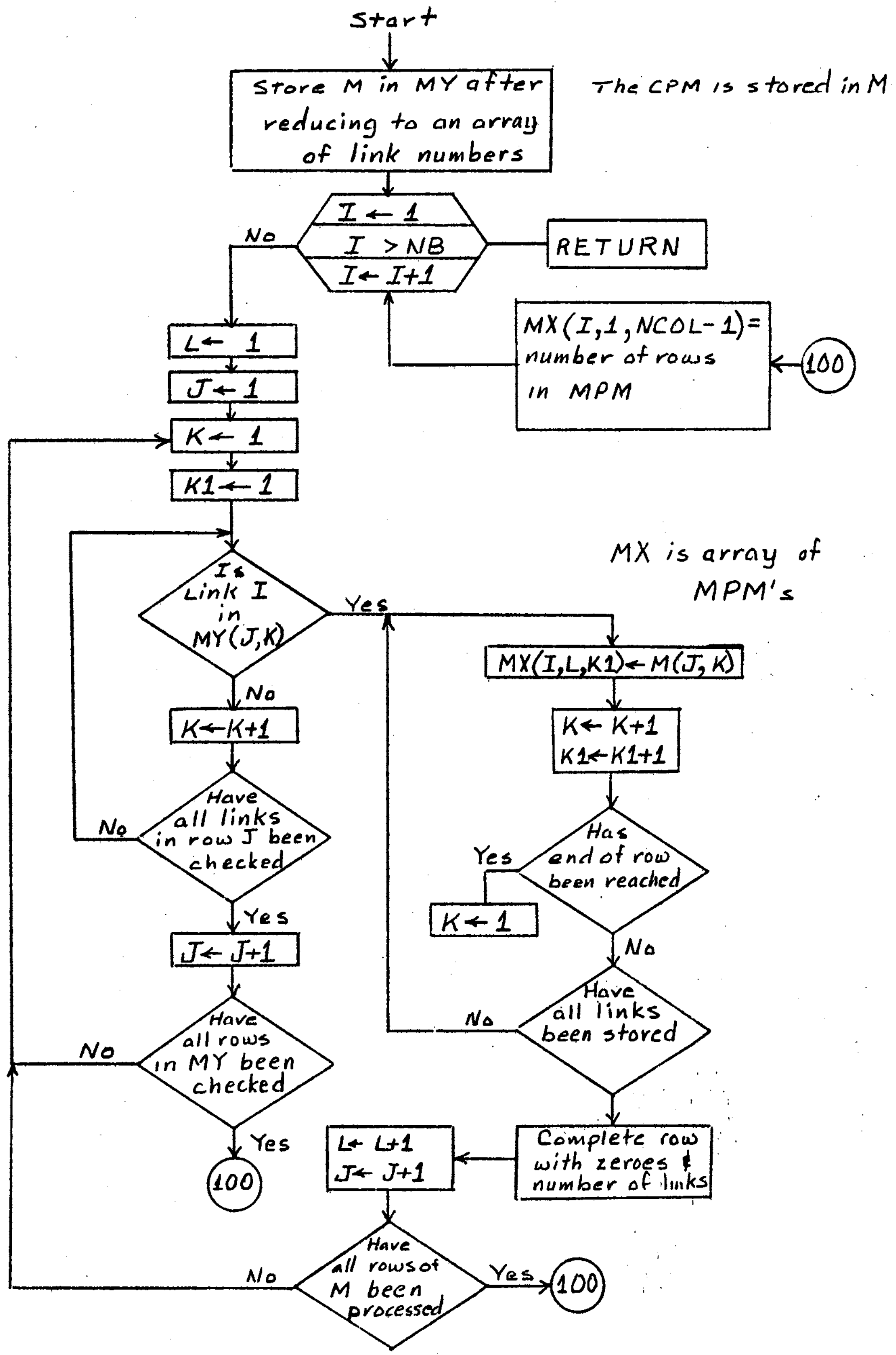
SUBROUTINE MECH

```
FORTRAN IV G LEVEL 1, MOD 4            MECH             DATE = 70192          09/43/16             PAGE 0001

              SUBROUTINE MECH
              IMPLICIT INTEGER*2(I-N)
              COMMON/ALL/M(15,20),NB,NL,NR,NCOL,NK,NCH,NELEM
              COMMON/MCH/MX(10,15,15)
              DIMENSION MY(15,15)
        C
        C STORE M IN MY AFTER REDUCING TO LINK NUMBERS
        C
              DO 11 I=1,NR
              NE=M(I,NCOL)
              IF(NE.EQ.0) GOTO 13
              DO 12 J=NE,NCOL
           12 MY(I,J)=M(I,J)
              DO 10 J=1,NE
           10 MY(I,J)=M(I,J)-(M(I,J)/100)*100
              GOTO 11
           13 MY(I,1)=0
              MY(I,2)=0
           11 CONTINUE
        C
        C DEVELOP MECHANISM MATRICES
        C
              DO 100 I=1,NB
        C
        C SEARCH FOR LINK I IN EACH ROW OF MY
        C
              L=1
              J=1
           21 K=1
              K1=1
           22 IF(MY(J,K).EQ.I) GOTO 30
              K=K+1
              IF(MY(J,K).NE.0) GOTO 22
              J=J+1
              IF(J.GT.NR) GOTO 100
              GOTO 21
        C
        C STORE ROW IN MX STARTING WITH LINK I
        C
           30 MX(I,L,K1)=M(J,K)
              K=K+1
              K1=K1+1
              IF(M(J,K).EQ.0) K=1
              IF(M(J,K).EQ.MX(I,L,1)) GOTO 40
              GOTO 30
        C
        C ALL LINKS HAVE BEEN STORED IN MX.  COMPLETE ROW WITH CORRESPONDING PORTION OF
        C M.
        C
           40 DO 41 J1=K1,NCOL
           41 MX(I,L,J1)=M(J,J1)
              L=L+1
              J=J+1
              IF(J.GT.NR) GOTO 100
              GOTO 21
        C
        C ALL ROWS CONTAINING LINK I HAVE BEEN PROPERLY STORED IN MX
        C


FORTRAN IV G LEVEL 1, MOD 4            MECH             DATE = 70192          09/43/16             PAGE 0002

          100 MX(I,1,NCOL-1)=L
              NCH=NB
              RETURN
              END
```



Start
Store M in MY after reducing to an array of link numbers
The CPM is stored in M
I ← 1
I > NB
I ← I+1
No
RETURN
MX(I,1,NCOL-1)= number of rows in MPM
100
L ← 1
J ← 1
K ← 1
K1 ← 1
MX is array of MPM's
Is Link I in MY(J,K)
Yes
MX(I,L,K1) ← M(J,K)
No
K ← K+1
K ← K+1
K1 ← K1+1
Have all links in row J been checked
No
Has end of row been reached
Yes
K ← 1
Yes
J ← J+1
No
Have all links been stored
No
Have all rows in MY been checked
No
Yes
100
L ← L+1
J ← J+1
Complete row with zeroes & number of links
Have all rows of M been processed
No
Yes
100

Figure 22. Flow Chart for Subroutine MECH

APPENDIX E

SUBROUTINE MECCOM

Subroutine MECCOM compares MPM's as described in Chapter IV. A flow chart for this subroutine is shown in Figure 23. A list of the variables used in MECCOM appears in Table XLIV and the subroutine itself is in Table XLV.

TABLE XLIV

VARIABLES USED IN SUBROUTINE MECCOM

MX	-	Array of MPM's
NCH	-	Number of MPM's to be compared
IEQUIV	-	Array indicating equivalent MPM's
ISAMEM	-	Array which indicates whether an MPM is already equivalent to another MPM
ISAMEY	-	Array which indicates whether a row is already equivalent to another row
IHOLD	-	Array used in cycling rows
ITAG	-	Counter used to determine how many times a row has been cycled
JSTORE	-	Array in which descriptions of unique chains are stores
NSTORE	-	Number of unique chains stored in JSTORE

All variables in COMMON/ALL/ are described in Appendix C in conjunction with Subroutine LOOP.

TABLE XLV

SUBROUTINE MECCOM

```
FORTRAN IV G LEVEL 1, MOD 4            MECCOM            DATE = 70191          20/49/09                PAGE 0001

                    SUBROUTINE MECCOM
                    IMPLICIT INTEGER*2(I-N)
                    COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
                    COMMON/MEC/IEQUIV(50,50)
                    COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
                   *,MZ5(15,15),MX(20,15,15)
                    COMMON/DBALL/JSTORE(100,6),JTEMP(100,6)
                    DIMENSION ISAMEM(50),ISAMEY(50),IHOLD(20)
              C CONVERT TO ARRAYS OF LINK TYPES
                    ITAG=0
                    DO 10 I=1,NCH
                    DO 5 J=1,NCH
                  5 IEQUIV(I,J)=0
                    ISAMEM(I)=0
                    NROW=MX(I,1,NCOL-1)-1
                    MX(I,1,NCCL-1)=NROW
                    DO 10 J=1,NROW
                    NCLM=MX(I,J,NCOL)
                    DO 10 K=1,NCLM
                 10 MX(I,J,K)=MX(1,J,K)/100
                    OC 11 I=1,NR
                 11 ISAMEY(I)=0
              C COMPARE EACH MECHANISM WITH ALL OTHER MECHANISMS
                    IX=1
                 19 IY=IX+1
              C CHECK TO SEE IF MX(IY) IS ALREADY EQUIVALENT
                 20 IF(ISAMEM(IY).EQ.1) GCTO 70
              C CHECK TO SEE IF MX(IY) HAS SAME NUMBER OF ROWS AS MX(IX)
                    IF(MX(IX,1,NCOL-1).NE.MX(IY,1,NCOL-1)) GOTO 70
              C COMPARE EACH ROW OF MX(IX) WITH ROWS OF MX(IY)
                    IXROW=1
                    IYROW=1
                 30 IF(MX(IX,IXROW,NCOL).NE.MX(IY,IYROW,NCOL)) GOTO 90
              C CHECK TO SEE IF MX(IY,IYROW) IS ALREADY EQUIVALENT TO A ROW IN MX(IX)
                    IF(ISAMEY(IYROW).EQ.1) GOTO 90
              C COMPARE ELEMENTS OF MX(IX,IXROW) WITH MX(IY,IYROW) IN FORWARD DIRECTION
                    IXCOL=1
                 40 IF(MX(IX,IXROW,IXCOL).NE.MX(IY,IYROW,IXCOL)) GOTO 110
                    IXCOL=IXCCL+1
                    IF(IXCOL.LE.MX(IX,IXROW,NCCL)) GOTO 40
                    ISAMEY(IYROW)=1
                    GCTC 100
              C INCREMENT IY
                 70 IY=IY+1
                    IF(IY.GT.NCH) GOTC 80
                    GOTO 20
              C INCREMENT IX
                 80 IX=IX+1
                    IF(IX.GT.NCH-1) GOTO 21C
                    GOTO 19
              C INCREMENT IYROW
                 90 IYROW=IYROW+1
                    ITAG=0
                    IF(IYROW.GT.MX(IX,1,NCOL-1)) GOTO 100
                    GOTO 30
              C INCREMENT IXROW
                100 IXROW=IXROW+1
                    IF(IXROW.GT.MX(IX,1,NCOL-1)) GOTO 200
```

TABLE XLV (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4            MECCOM            DATE = 70191          20/49/09               PAGE 0002
            IYROW=1
            GOTO 30
      C COMPARE MX(IY,IYROW) IN REVERSE
        110 IXCOL=1
            IYCOL=MX(IY,IYROW,NCOL)+1
            MX(IY,IYROW,IYCOL)=MX(IY,IYROW,1)
        111 IF(MX(IX,IXROW,IXCOL).NE.MX(IY,IYROW,IYCOL)) GOTO 120
            IXCOL=IXCOL+1
            IYCOL=IYCOL-1
            IF(IXCOL.LE.MX(IX,IXROW,NCOL)) GOTO 111
            ISAMEY(IYROW)=1
            GOTO 100
        120 NE=MX(IY,IYROW,NCOL)
            IF(ITAG.EQ.NELEM-1) GOTO 90
            DO 130 IA=2,NE
            IF(MX(IY,IYROW,IA).EQ.MX(IY,IYROW,1)) GOTO 131
        130 CONTINUE
            GOTO 90
      C CYCLE ROW TO NEXT ELEMENT
        131 ISTOP=IA
            JH=1
        132 IHOLD(JH)=MX(IY,IYROW,IA)
            IA=IA+1
            IF(IA.EQ.ISTOP) GOTO 133
            IF(IA.GT.NE) IA=1
            JH=JH+1
            GOTO 132
        133 DO 134 IH=1,JH
        134 MX(IY,IYROW,IH)=IHOLD(IH)
            ITAG=ITAG+1
            IXCOL=1
            GOTO 40
      C ALL ROWS HAVE BEEN COMPARED.  CHECK TO SEE IF MATRICES ARE EQUIVALENT.
        200 NROW=MX(IX,1,NCOL-1)
            ISUM=0
            DO 201 I=1,NROW
            IF(ISAMEY(I).EQ.1) ISUM=ISUM+1
        201 ISAMEY(I)=0
            IF(ISUM.EQ.NROW) GOTO 220
            GOTO 70
        220 IEQUIV(IX,IY)=IY
            ISAMEM(IY)=1
            DO 1000 IW=1,NELEM
       1000 JSTORE(IY,IW)=0
            GOTO 70
      C WRITE EQUIVALENCE
        210 CALL PRINT
            DO 1001 IW=2,NCH
            IF(JSTORE(IW,1).EQ.0) GOTO 1002
       1001 CONTINUE
            NSTORE=NCH
            RETURN
       1002 IZERO=IW
            IWP1=IW+1
            DO 1006 IW=IWP1,NCH
            IF(JSTORE(IW,1).NE.0) GOTO 1004
            GOTO 1006
       1004 DO 1005 IX=1,NELEM

FORTRAN IV G LEVEL 1, MOD 4            MECCOM            DATE = 70191          20/49/09               PAGE 0003
       1005 JSTORE(IZERO,IX)=JSTORE(IW,IX)
            IZERO=IZERO+1
       1006 CONTINUE
            NSTORE=IZERO-1
            RETURN
            END
```

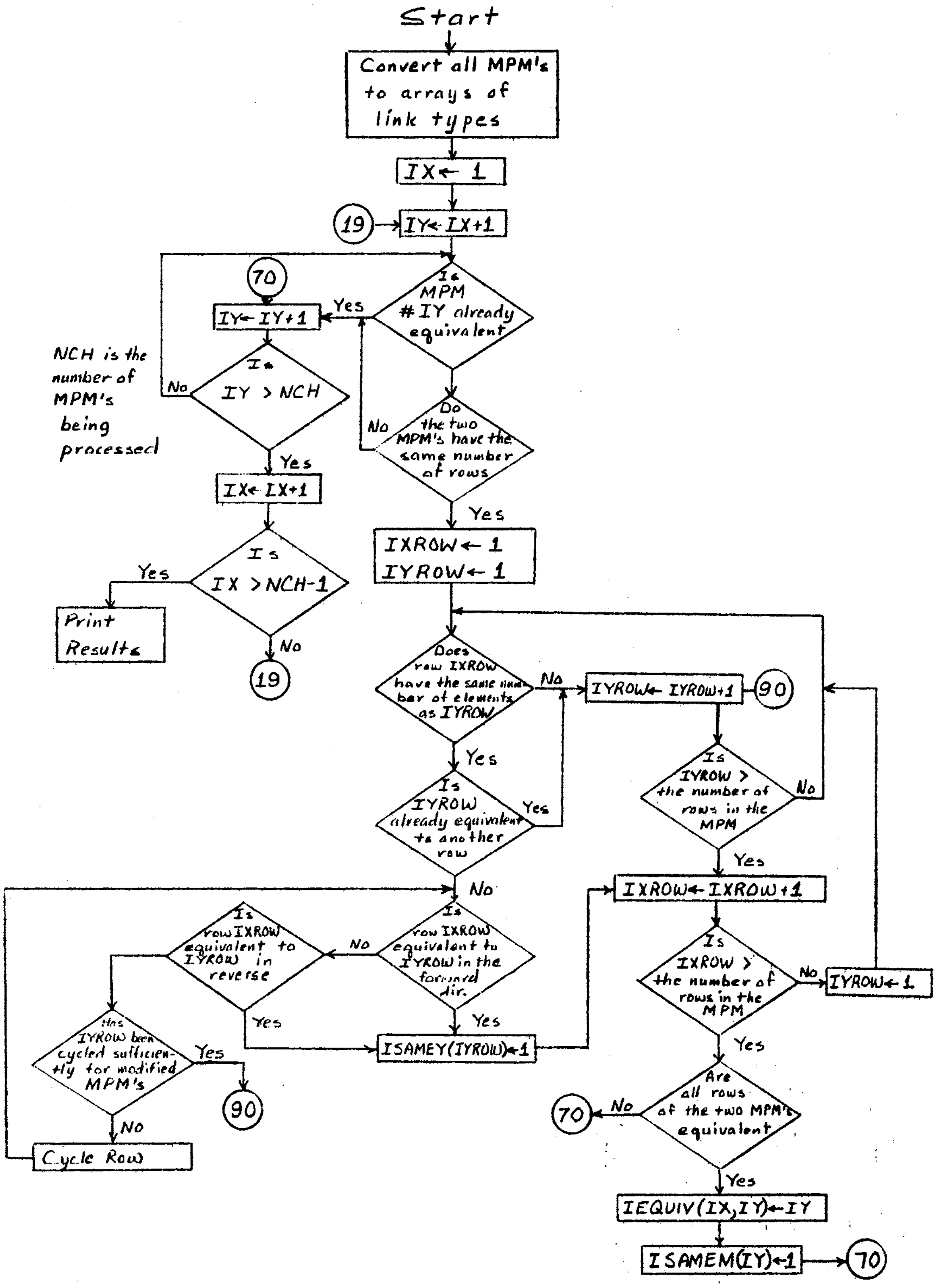

Start
Convert all MPM's to arrays of link types
IX ← 1
19
IY ← IX+1
Is MPM #IY already equivalent
Yes
70
IY ← IY+1
Is IY > NCH
No
Yes
NCH is the number of MPM's being processed
IX ← IX+1
Is IX > NCH-1
Yes
Print Results
No
19
Do the two MPM's have the same number of rows
No
Yes
IXROW ← 1
IYROW ← 1
Does row IXROW have the same number of elements as IYROW
No
IYROW ← IYROW+1
90
Yes
Is IYROW already equivalent to another row
Yes
No
Is IYROW > the number of rows in the MPM
No
Yes
IXROW ← IXROW+1
Is row IXROW equivalent to IYROW in the forward dir.
No
Yes
Is row IXROW equivalent to IYROW in reverse
Yes
ISAMEY(IYROW) ← 1
Has IYROW been cycled sufficiently for modified MPM's
Yes
90
No
Cycle Row
Is IXROW > the number of rows in the MPM
No
IYROW ← 1
Yes
Are all rows of the two MPM's equivalent
No
70
Yes
IEQUIV(IX,IY) ← IY
ISAMEM(IY) ← 1
70

Figure 23. Flow Chart for Subroutine MECCOM

APPENDIX F

PROGRAM FOR DETERMINING THE TEN-LINK MECHANISMS

The program shown in Table XLVI was used to generate the ten-link mechanisms. The input data required for the program (and all other programs) is described in Table XLVII.

TABLE XLVI

PROGRAM FOR DETERMINING TEN-LINK MECHANISMS

```
FORTRAN IV G LEVEL 1, MOD 4           MAIN              DATE = 70191          20/59/33              PAGE 0001

C
C TEN-LINK CHAINS
C
C DEFINITION OF INPUT DATA
C
C      NL   - NUMBER OF LOOPS
C      NB   - NUMBER OF BARS
C      NR   - NUMBER OF ROWS IN LOOP MATRIX
C      NCOL - NUMBER OF COLUMNS IN LOOP MATRIX
C      NK   - NUMBER OF LOOPS IN ADDITION TO BASIC LOOPS
C      NC   - NUMBER OF CHAINS
C      KM&LM- SEQUENCES FOR DERIVING LOOPS
C      KS&LS- SECONDARY SEQUENCES FOR DERIVING LOOPS
C      KT&LT- TERTIARY SEQUENCES FOR DERIVING LOOPS
C
C**************************************************************************
      IMPLICIT INTEGER*2(I-N)
      COMMON/ALL/M(15,20),NB,NL,NR,NCOL,NK,NCH,NELEM
      COMMON/LOO/KM(11),LM(11),KS(11),LS(11),KT(5),LT(5)
      COMMON/MCH/MX(10,15,15)
      COMMON/PRI/ITOTAL,I,IROW
      COMMON/MEC/IEQUIV(10,10)
      READ(5,1) NL,NB,NR,NCOL,NC
      NK=NR-NL
    1 FORMAT(15I5)
      READ(5,1)(KM(I),I=1,NK)
      READ(5,1)(LM(I),I=1,NK)
      NELEM=1
      NS=NK-NL
      READ(5,1)(KS(I),I=NS,NK)
      READ(5,1)(LS(I),I=NS,NK)
      READ(5,1)(KT(I),I=1,5)
      READ(5,1)(LT(I),I=1,5)
      ITOTAL=0
      CALL PRINT1
      DO 2 I=1,NC
      CALL LOOP
      CALL MECH
      CALL MECCOM
      IF(IROW.GT.25) CALL PRINT1
    2 CONTINUE
      WRITE(6,4) ITOTAL
    4 FORMAT('OTOTAL NUMBER OF 10-LINK MECHANISMS - ',
     1 I5)
      WRITE(6,1000)
 1000 FORMAT('1')
      STOP
      END
```

TABLE XLVI (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4           PRINT           DATE = 70191         20/59/33              PAGE 0001

          SUBROUTINE PRINT
          IMPLICIT INTEGER*2(I-N)
          COMMCN/ALL/M(15,20),NB,NL,NR,NCOL,NK,NCH,NELEM
          COMMON/MEC/IEQUIV(10,10)
          COMMON/PRI/ITOTAL,1,IROW
          DIMENSION ANUM(10),EQUIV(10)
          DATA BLANK/' '/
          DATA EX/' X'/
          DATA ANUM/' 1',' 2',' 3',' 4',' 5',' 6',' 7',' 8',' 9','10'/
        1 FORMAT('+',T11,10(A2,2X))
        2 FORMAT(T10,11('|   '))
        3 FORMAT(1X,68('-'))
        6 FORMAT('+  ',I3,'  |')
        7 FORMAT('+',T9,'|',10('|    '),T50,'||')
        8 FORMAT('+',T51,'|',T59,I2)
          ISUM=10
          WRITE(6,3)
          WRITE(6,7)
          WRITE(6,2)
          WRITE(6,6) I
          DO 20 IX=1,10
          DO 10 IY=1,10
          EQUIV(IY)=BLANK
          IF(IEQUIV(IX,IY).EQ.99) EQUIV(IX)=EX
          IF(IEQUIV(IX,IY).EQ.IY)EQUIV(IY)=ANUM(IX)
          IF(EQUIV(IY).NE.BLANK) ISUM=ISUM-1
       10 CONTINUE
          WRITE(6,1) (EQUIV(J),J=1,10)
       20 CONTINUE
          WRITE(6,8) ISUM
          ITOTAL=ITOTAL+ISUM
          IROW=IROW+1
          RETURN
          END
```

```
FORTRAN IV G LEVEL 1, MOD 4           PRINT1          DATE = 70191         20/59/33              PAGE 0001

          SUBROUTINE PRINT1
          IMPLICIT INTEGER*2(I-N)
          COMMON/PRI/ITOTAL,I,IROW
          DIMENSION ANUM(10)
          DATA ANUM/' 1',' 2',' 3',' 4',' 5',' 6',' 7',' 8',' 9','10'/
        1 FORMAT('+',T11,10(A2,2X))
        2 FORMAT(T10,11('|   '))
        4 FORMAT('1',/////,T25,'LINK NUMBER')
        5 FORMAT('+CHAIN  |',T51,'| UNIQUE MECHANISMS')
          WRITE(6,4)
          WRITE(6,2)
          WRITE(6,1)(ANUM(IA),IA=1,10)
          WRITE(6,5)
          IROW=1
          RETURN
          END
```

TABLE XLVII

INPUT DATA FOR COMPUTER PROGRAMS

Card 1

NL	-	Number of loops in chains
NB	-	Number of bars (links) in chains
NR	-	Number of rows in CPM
NCOL	-	Number of columns in CPM
NC	-	Number of chains to be processed

Data is read in with FORMAT (15I5).

Card 2 KM(I)
Card 3 LM(I)

KM and LM are the rows to be combined in forming other rows of the CPM. For the ten-link chains, to form row 5, rows 1 and 2 of the CPM are combined. Therefore, KM(1) would be 1 and LM(1) would be 2. Table V shows all combinations used for the ten-link chains. The data on cards 2 and 3 is read in with FORMAT (15I5).

Card 4 KS(I)
Card 5 LS(I)

KS and LS are the second row combinations to be used if the primary combinations fail. For the ten-link chains, the secondary combinations are required only for rows 9 through 15 of the CPM, therefore, only 7 entries are required on each card. The data on cards 4 and 5 is read in with FORMAT (15I5).

Card 6 KT(I)
Card 7 LT(I)

KT and LT are additional row combinations used only for row 15 of the CPM for ten-link chains. They are read in with FORMAT (15I5).

Cards 8 -

These cards are read in by Subroutine LOOP and furnish the loop matrices. For the ten-link chains, 4 cards are required for each chain, one card per loop. Each card provides one row of the loop matrix. (Do not insert number showing the count of the elements in the row). These cards are read in with FORMAT (20I4).

TABLE XLVII (Continued)

The input cards for the 10-link chains appeared as follows:

Card 1)	4	10	15	15	230						
Card 2)	1	1	4	2	2	3	1	4	10	8	5
Card 3)	2	3	1	3	4	4	8	5	1	4	10
Card 4)	5	7	3	2	6						
Card 5)	3	2	7	10	9						
Card 6)	8	4	3	2	1						
Card 7)	7	11	12	13	14						
Card 8)	301	312	310	309							
Card 9)	302	103	104	305	310						
Card 10)	305	306	309	310							
Card 11)	306	107	108	301	309						

NOTE: The programs were set up for binary links being represented by a type number of 1 (as in Franke's notation) instead of 2. Therefore, the numbers 103 and 104 on card 9 would represent binary links 3 and 4.

APPENDIX G

PROGRAM FOR DETERMINING THE TEN-LINK CHAINS WITH SPRINGS

The complete program for determining the ten-link chains with springs is shown in Table XLVIII. Flow charts for Subroutines SPRLOC and SPRNG3 appear in Figures 24 and 25 respectively. This program serves as the model for all programs which follow.

TABLE XLVIII

PROGRAM FOR DETERMINING TEN-LINK CHAINS WITH SPRINGS

```
FORTRAN IV G LEVEL 1, MOD 4           MAIN              DATE = 70191          20/49/09              PAGE 0001

      C
      C TEN-LINK CHAINS
      C
      C DEFINITION OF INPUT DATA
      C
      C     NL   - NUMBER OF LOOPS
      C     NB   - NUMBER OF BARS
      C     NR   - NUMBER OF ROWS IN LOOP MATRIX
      C     NCOL - NUMBER OF COLUMNS IN LOOP MATRIX
      C     NK   - NUMBER OF LOOPS IN ADDITION TO BASIC LOOPS
      C     NC   - NUMBER OF CHAINS
      C     KM&LM- SEQUENCES FOR DERIVING LOOPS
      C     KS&LS- SECONDARY SEQUENCES FOR DERIVING LOOPS
      C     KT&LT- TERTIARY SEQUENCES FOR DERIVING LOOPS
      C
      C************************************************************************
            IMPLICIT INTEGER*2(I-N)
            COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
            COMMON/LOO/KM(11),LM(11),KS(11),LS(11),KT(5),LT(5)
            COMMON/DBL/JLOC(20,3),JOROW(20,15),JROW(20,15)
            COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
           *,MZ5(15,15),MX(20,15,15)
            COMMON/PRI/ITOTAL(6)
            READ(5,1) NL,NB,NR,NCOL,NC
            NK=NR-NL
          1 FORMAT(15I5)
            READ(5,1)(KM(I),I=1,NK)
            READ(5,1)(LM(I),I=1,NK)
            NS=NK-NL
            READ(5,1)(KS(I),I=NS,NK)
            READ(5,1)(LS(I),I=NS,NK)
            READ(5,1)(KT(I),I=1,5)
            READ(5,1)(LT(I),I=1,5)
            NCHS=1
            DO 2 I=1,NC
            NELEM=1
            CALL LOOP
            CALL SPRLOC
            IF(NCHS.EQ.0.AND.NCH.EQ.0) GOTO 40
            WRITE(6,3) I
          3 FORMAT('1',//,' CHAIN  ',I3,/,1X,11('*'))
            GOTO 41
         40 WRITE(6,5) I
          5 FORMAT(//,' CHAIN  ',I3,/,1X,11('*'))
            GOTO 31
         41 NCHS=NCH
            IF(NCH.EQ.0) GOTO 31
            CALL ARRAY
            GOTO 32
         31 WRITE(6,8)
          8 FORMAT(T5,'NO POSSIBLE CHAINS')
            GOTO 2
         32 IF(NCHS.LT.NELEM) GOTO 2
            NCH=NCHS
            GOTO (100,200,300,400),NELEM
        100 CALL SPRNG1
            GOTO 33
        200 CALL SPRNG2


FORTRAN IV G LEVEL 1, MOD 4           MAIN              DATE = 70191          20/49/09              PAGE 0002

            GOTO 33
        300 CALL SPRNG3
            GOTO 33
        400 CALL SPRNG4
         33 IF(NCH-1)31,35,34
         34 CALL MECCOM
            GOTO 36
         35 CALL PRINT
         36 NELEM=NELEM+1
            IF(NELEM-4) 32,32,2
          2 CONTINUE
            NELEM=NELEM-1
            WRITE(6,1000)
            DO 6 I=1,4
          6 WRITE(6,4) I,ITOTAL(I)
          4 FORMAT('0TOTAL NUMBER OF 10-LINK CHAINS WITH',I3,' SPRING(S) - ',
           * I5)
            WRITE(6,1000)
       1000 FORMAT('1')
            STOP
            END
```

TABLE XLVIII (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4          SPRLOC          DATE = 70191          20/49/09          PAGE 0001

      SUBROUTINE SPRLOC
      IMPLICIT INTEGER*2(I-N)
      COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMON/SPL/LPAIR(20,2),LROW(20,15),LINK2(10)
      ISPRNG=1
      DO 1 I=1,NB
    1 LINK2(I)=0
      DO 2 J=1,4
      LPAIR(J,1)=0
      DO 2 I=1,NR
    2 LROW(J,I)=0
C
C SEARCH FOR ADJACENT BINARY LINKS
C
      DO 10 I=1,NB
      IF(LINK2(I).EQ.1) GOTO 10
      DO 20 J=1,NL
      NE=M(J,NCOL)
      DO 20 K=1,NE
      IF(M(J,K)-(M(J,K)/100)*100.EQ.I) GOTO 40
   20 CONTINUE
      GOTO 10
   40 IF(M(J,K)/100.EQ.1) GOTO 50
      IF((M(J,K)-9000)/100.EQ.1) GOTO 60
      GOTO 10
C
C LINK I IS A BINARY LINK.  CHECK ADJACENT LINKS.
C
   50 IF(M(J,K+1)/100.EQ.1) GOTO 70
      IF(K.EQ.1) GOTO 51
      IF(M(J,K-1)/100.EQ.1) GOTO 71
      GOTO 10
   51 IF(M(J,NE)/100.EQ.1) GOTO 72
      GOTO 10
   60 IF((M(J,K+1)-9000)/100.EQ.1) GOTO 70
      IF(K.EQ.1) GOTO 61
      IF((M(J,K-1)-9000)/100.EQ.1) GOTO 71
      GOTO 10
   61 IF((M(J,NE)-9000)/100.EQ.1) GOTO 72
      GOTO 10
C
C LINK I IS ADJACENT TO ANOTHER BINARY LINK.
C
   70 LPAIR(ISPRNG,1)=M(J,K)
      LPAIR(ISPRNG,2)=M(J,K+1)
      L=M(J,K+1)-(M(J,K+1)/100)*100
      LINK2(L)=1
      GOTO 80
   71 LPAIR(ISPRNG,1)=M(J,K)
      LPAIR(ISPRNG,2)=M(J,K-1)
      L=M(J,K-1)-(M(J,K-1)/100)*100
      LINK2(L)=1
      GOTO 80
   72 LPAIR(ISPRNG,1)=M(J,K)
      LPAIR(ISPRNG,2)=M(J,NE)
      L=M(J,NE)-(M(J,NE)/100)*100
      LINK2(L)=1
C

FORTRAN IV G LEVEL 1, MOD 4          SPRLOC          DATE = 70191          20/49/09          PAGE 0002

C LOCATE ROWS CONTAINING ADJACENT BINARY LINKS.
C
   80 DO 81 IA=1,NR
      NE=M(IA,NCOL)
      IF(NE.EQ.0) GOTO 81
      DO 82 IB=1,NE
      IF(M(IA,IB).EQ.LPAIR(ISPRNG,1)) LROW(ISPRNG,IA)=IA
   82 CONTINUE
   81 CONTINUE
      ISPRNG=ISPRNG+1
   10 CONTINUE
      NCH=ISPRNG-1
      RETURN
      END
```

TABLE XLVIII (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4            CHAIN               DATE = 70191           20/49/09              PAGE 0001

      SUBROUTINE CHAIN
      IMPLICIT INTEGER*2(I-N)
      COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMON/DBALL/JSTORE(100,6),JTEMP(100,6)
      CCMPON/CHA/IDBL,KSTCRE,KCUNT
      DIMENSION ANUM(10),ARRAY(8,12)
      DATA C/'--'/,D1/'-'/,C/','/,B/'|'/
      DATA ANUM/'0','1','2','3','4','5','6','7','8','9'/,BLANK/' '/
      WRITE(6,1)(B,I,I=KSTORE,IDBL),B
    1 FORMAT(T5,'CHAIN# ',8(A1,4X,I2,6X),A1)
      WRITE(6,2)(D,D,D,D,D,D,D1,I=KSTORE,IDBL)
    2 FORMAT(T5,'--------',8(6A2,A1))
      I=0
      DO 10 IA=KSTORE,IDBL
      ISUM=0
      I=I+1
      DO 11 J=1,NELEM
      IF(JSTORE(IA,J).GE.10) GOTO 12
      ISUM=ISUM+2
      GOTO 11
   12 ISUP=ISUM+3
   11 CONTINUE
      IT=(12-ISUM)/2
      DO 13 J=1,IT
   13 ARRAY(I,J)=BLANK
      IT=IT+1
      DO 14 J=1,NELEM
      NUM=JSTORE(IA,J)
      NDIG=2
      IF(NUM.LT.10) NDIG=1
      IF(NDIG.EC.2) GOTO 15
      ARRAY(I,IT)=ANUM(NUM+1)
      IT=IT+1
      IF(J.EQ.NELEM) GOTO 14
      GOTO 16
   15 NUM1=NUM/10
      ARRAY(I,IT)=ANUM(NUM1+1)
      NUM2=NUM-NUM1*10
      IT=IT+1
      ARRAY(I,IT)=ANUM(NUM2+1)
      IT=IT+1
      IF(J.EC.NELEM) GCTO 14
   16 ARRAY(I,IT)=C
      IT=IT+1
   14 CONTINUE
      DO 17 J=IT,12
   17 ARRAY(I,J)=BLANK
   10 CONTINUE
      KOUNT=KOUNT-1
      WRITE(6,3)(B,(ARRAY(I,J),J=1,12),I=1,KOUNT),B
    3 FORMAT(T5,'SPRINGS',8(13A1),A1,//)
      KCUNT=1
      KSTORE=IDBL+1
      RETURN
      END
FORTRAN IV G LEVEL 1, MOD 4            SPRADD              DATE = 70191           20/49/09              PAGE 0001

      SUBRCUTINE SPRADD(NR,NCOL,MY,MZ,ISPRNG)
      IMPLICIT INTEGER*2(I-N)
      CCMMON/SPL/LPAIR(20,2),LROW(20,15),LINK2(10)
      DIMENSION MY(15,20),MZ(15,20)
      DO 1 I=1,NR
      NE=MY(I,NCOL)
      IF(NE.EQ.0) NE=1
      DO 1 J=NE,NCOL
    1 MZ(I,J)=0
      L1=LPAIR(ISPRNG,1)
      L2=LPAIR(ISPRNG,2)
C
C LOCATE LINK L1
C
      DC 10 I=1,NR
      NE=MY(I,NCOL)
      IF(NE.EQ.0) GOTO 10
      IF(LROW(ISPRNG,I).NE.T) GCTO 9
      DO 20 J=1,NE
      IF(MY(I,J).EQ.L1.OR.MY(I,J).EQ.L2) GOTO 30
   20 CONTINUE
   30 IF(MY(I,J+1).EQ.L2.OR.MY(I,J+1).EQ.L1) GOTO 40
      GOTO 50
   40 ICOL=1
      JS=J
      MZ(I,ICOL)=1111
      J=J+2
      IF(J.GT.NE)J=J-NE
   41 ICOL=ICOL+1
      MZ(I,ICOL)=MY(I,J)
      J=J+1
      IF(J.GT.NE)J=1
      IF(J.EQ.JS) GOTO 70
      GOTO 41
   50 J=NE
      GCTO 40
   70 MZ(I,NCOL)=NE-1
      GOTO 10
    9 DO 8 J=1,NE
      MZ(I,J)=MY(I,J)
    8 MZ(I,NCOL)=NE
   10 CONTINUE
      RETURN
      END
```

TABLE XLVIII (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4              SPRNG2              DATE = 70191          20/49/09              PAGE 0001

              SUBROUTINE SPRNG2
              IMPLICIT INTEGER*2(I-N)
              COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
              COMMON/OBALL/JSTORE(100,6),JTEMP(100,6)
              COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
             *,MZ5(15,15),MX(20,15,15)
              COMMON/SPL/LPAIR(20,2),LROW(20,15),LINK2(10)
              COMMON/CHN/ISPRNG,KSTORE,KCUNT
              ISPRNG=1
              KCUNT=1
              KSTORE=1
              WRITE(6,1)
            1 FORMAT(/,'OIDENTIFICATION OF CHAINS WITH TWO SPRINGS',/,1X,
             *41('*'),/)
              DO 100 I=1,NSTORE
          100 JTEMP(I,1)=JSTORE(I,1)
              DO 10 IZ=1,NSTORE
            9 I=JTEMP(IZ,1)
              CALL SPRADD(NR,NCOL,M,MZ1,I)
              IP1=I+1
              IF(IP1.GT.NCH) GOTO 10
              DO 30 J=IP1,NCH
              JSTORE(ISPRNG,1)=I
              JSTORE(ISPRNG,2)=J
              KOUNT=KOUNT+1
              IF(KOUNT.GT.8) CALL CHAIN
              CALL SPRADD(NR,NCOL,MZ1,MZ,J)
C FILL MX
              IROW=1
              DO 20 IA=1,NR
              IF(LROW(I,IA).EQ.0.AND.LROW(J,IA).EQ.0) GOTO 20
              DO 40 IB=1,NCOL
           40 MX(ISPRNG,IROW,IB)=MZ(IA,IB)
              IROW=IROW+1
           20 CONTINUE
              MX(ISPRNG,1,NCOL-1)=IROW
              ISPRNG=ISPRNG+1
           30 CONTINUE
           10 CONTINUE
              ISPRNG=ISPRNG-1
              IF(ISPRNG.GE.KSTORE) CALL CHAIN
              NCH=ISPRNG
              RETURN
              END
```

```
FORTRAN IV G LEVEL 1, MOD 4              SPRNG1              DATE = 70191          20/49/09              PAGE 0001

              SUBROUTINE SPRNG1
              IMPLICIT INTEGER*2(I-N)
              COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
              COMMON/SPL/LPAIR(20,2),LROW(20,15),LINK2(10)
              COMMON/OBALL/JSTORE(100,6),JTEMP(100,6)
              COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
             *,MZ5(15,15),MX(20,15,15)
              WRITE(6,1)
            1 FORMAT(/,'OCHAINS WITH ONE SPRING (IDENTIFIED BY THE SPRING NUMBER
             * ABOVE)',/,1X,62('*'),/)
              DO 10 I=1,NCH
              ISPRNG=I
              JSTORE(I,1)=I
              CALL SPRADD(NR,NCOL,M,MZ,I)
C FILL MX
              IROW=1
              DO 20 IA=1,NR
              IF(LROW(I,IA).EQ.0) GOTO 20
              DO 40 IB=1,NCOL
           40 MX(ISPRNG,IROW,IB)=MZ(IA,IB)
              IROW=IROW+1
           20 CONTINUE
              MX(ISPRNG,1,NCOL-1)=IROW
           10 CONTINUE
              NCH=ISPRNG
              RETURN
              END
```

TABLE XLVIII (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4            SPRNG3           DATE = 70191          20/49/09                  PAGE 0001

            SUBROUTINE SPRNG3
            IMPLICIT INTEGER*2(I-N)
            COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
            COMMON/DBALL/JSTORE(100,6),JTEMP(100,6)
            COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
           *,MZ5(15,15),MX(20,15,15)
            COMMON/SPL/LPAIR(20,2),LROW(20,15),LINK2(10)
            COMMON/CHN/ISPRNG,KSTORE,KOUNT
            ISPRNG=1
            KOUNT=1
            KSTORE=1
            WRITE(6,1)
          1 FORMAT(/,'OIDENTIFICATION OF CHAINS WITH THREE SPRINGS',/,1X,
           *43('+'),/)
            DO 100 I=1,NSTORE
            JTEMP(I,2)=JSTORE(I,2)
        100 JTEMP(I,1)=JSTORE(I,1)
            DO 10 IZ=1,NSTORE
            IF(IZ.EQ.1) GOTO 9
            IF(JTEMP(IZ,1).NE.JTEMP(IZ-1,1)) GOTO 9
            GOTO 8
          9 I=JTEMP(IZ,1)
            CALL SPRADD(NR,NCOL,M,MZ1,I)
          8 J=JTEMP(IZ,2)
            CALL SPRADD(NR,NCOL,MZ1,MZ2,J)
            JP1=J+1
            IF(JP1.GT.NCH) GOTO 10
            DO 30 K=JP1,NCH
            JSTORE(ISPRNG,1)=I
            JSTORE(ISPRNG,2)=J
            JSTORE(ISPRNG,3)=K
            KOUNT=KOUNT+1
            IF(KOUNT.GT.8) CALL CHAIN
            CALL SPRADD(NR,NCOL,MZ2,MZ,K)
      *C FILL MX
            IROW=1
            DO 20 IA=1,NR
            IF(LROW(I,IA).EQ.0.AND.LROW(J,IA).EQ.0.AND.LROW(K,IA).EQ.0) GOTO
           * 20
            DO 40 IB=1,NCOL
         40 MX(ISPRNG,IROW,IB)=MZ(IA,IB)
            IROW=IROW+1
         20 CONTINUE
            MX(ISPRNG,1,NCOL-1)=IROW
            ISPRNG=ISPRNG+1
         30 CONTINUE
         10 CONTINUE
            ISPRNG=ISPRNG-1
            IF(ISPRNG.GE.KSTORE) CALL CHAIN
            NCH=ISPRNG
            RETURN
            END
```

TABLE XLVIII (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4            SPRNG4            DATE = 70191          20/49/09                PAGE 0001

      SUBROUTINE SPRNG4
      IMPLICIT INTEGER*2(I-N)
      COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMON/DBALL/JSTORE(100,6),JTEMP(100,6)
      COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
     *,MZ5(15,15),MX(20,15,15)
      COMMON/SPL/LPAIR(20,2),LROW(20,15),LINK2(10)
      COMMON/CHN/ISPRNG,KSTORE,KOUNT
      ISPRNG=1
      KOUNT=1
      KSTORE=1
      WRITE(6,1)
    1 FORMAT(/,'0IDENTIFICATION OF CHAINS WITH FOUR SPRINGS',/,1X,
     *42('*'),/)
      DO 100 I=1,NSTORE
      JTEMP(I,3)=JSTORE(I,3)
      JTEMP(I,2)=JSTORE(I,2)
  100 JTEMP(I,1)=JSTORE(I,1)
      DO 10 IZ=1,NSTORE
      IF(IZ.EQ.1) GOTO 9
      IF(JTEMP(IZ,1).NE.JTEMP(IZ-1,1)) GOTO 9
      IF(JTEMP(IZ,2).NE.JTEMP(IZ-1,2)) GOTO 8
      GOTO 7
    9 I=JTEMP(IZ,1)
      CALL SPRADD(NR,NCOL,M,MZ1,I)
    8 J=JTEMP(IZ,2)
      CALL SPRADD(NR,NCOL,MZ1,MZ2,J)
    7 K=JTEMP(IZ,3)
      CALL SPRADD(NR,NCOL,MZ2,MZ3,K)
      KP1=K+1
      IF(KP1.GT.NCH) GOTO 10
      DO 30 L=KP1,NCH
      JSTORE(ISPRNG,1)=I
      JSTORE(ISPRNG,2)=J
      JSTORE(ISPRNG,3)=K
      JSTORE(ISPRNG,4)=L
      KOUNT=KOUNT+1
      IF(KOUNT.GT.8) CALL CHAIN
      CALL SPRADD(NR,NCOL,MZ3,MZ,L)
C FILL MX
      IROW=1
      DO 20 IA=1,NR
      IF(LROW(I,IA).EQ.0.AND.LROW(J,IA).EQ.0.AND.LROW(K,IA).EQ.0.AND.
     * LROW(L,IA).EQ.0) GOTO 20
      DO 40 IB=1,NCOL
   40 MX(ISPRNG,IROW,IB)=MZ(IA,IB)
      IROW=IROW+1
   20 CONTINUE
      MX(ISPRNG,1,NCOL-1)=IROW
      ISPRNG=ISPRNG+1
   30 CONTINUE
   10 CONTINUE
      ISPRNG=ISPRNG-1
      IF(ISPRNG.GE.KSTORE) CALL CHAIN
      NCH=ISPRNG
      RETURN
      END
```

TABLE XLVIII (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4          PRINT          DATE = 70191          20/49/09          PAGE 0001

      SUBROUTINE PRINT
      IMPLICIT INTEGER*2(I-N)
      INTEGER IEQ
      COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMCN/PRI/ITCTAL(6)
      COMMON/MEC/IEQLIV(50,50)
      DIMENSION IPRINT(20)
      DATA IEQ/'='/
      ISUBT=NCH
      WRITE(6,3)
    3 FORMAT('0',T5,'EQUIVALENT CHAINS',/)
      IF(NCH.EQ.1) GOTO 40
      DC 30 IX=1,NCH
      ISUM=0
      DO 20 IY=1,NCH
      IF(IEQUIV(IX,IY).EQ.0) GCTO 20
      ISUM=ISUM+1
      IPRINT(ISUM)=IEQUIV(IX,IY)
   20 CONTINUE
      IF(ISUM.EQ.0) GOTO 30
      WRITE(6,1) IX,(IEQ,IPRINT(I),I=1,ISUM)
    1 FORMAT(T10,I2,10(A1,I2))
      ISUBT=ISUBT-ISUM
   30 CONTINUE
      IF(ISUBT.EQ.NCH) GOTO 40
   31 WRITE(6,2) ISUBT
    2 FORMAT('0',T5,'NUMBER CF UNICUE CHAINS -',I2)
      ITCTAL(NELEM)=ITOTAL(NELEM)+ISUBT
      RETURN
   40 WRITE(6,4)
    4 FORMAT(T10,'NONE')
      GOTO 31
      END
```

```
FCRTRAN IV G LEVEL 1, MOD 4          ARRAY          DATE = 70191          20/49/09          PAGE 0001

      SUBROUTINE ARRAY
      IMPLICIT INTEGER*2(I-N)
      COMMCN/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMON/SPL/LPAIR(20,2),LRCW(20,15),LINK2(10)
      DIMENSION L(1C0,3)
      DATA C/'--'/,PL/'('/,PR/')'/,C/','/
      DO 1C I=1,NCH
      L(I,1)=LPAIR(I,1)-(LPAIR(I,1)/100)*100
   10 L(I,2)=LPAIR(I,2)-(LPAIR(I,2)/100)*100
      WRITE(6,1)
    1 FORMAT('0IDENTIFICATION OF SPRINGS')
      WRITE(6,2)(I,I=1,NCH)
    2 FORMAT('0',T5,'SPRING#',T18,11('|  ',I2,3X))
      WRITE(6,3)(D,C,D,D,I=1,NCH)
    3 FORMAT(T5,13('-'),10(4A2))
      WRITE(6,4)(L(I,1),C,L(I,2),I=1,NCH)
    4 FORMAT(T5,'LINK1, LINK2 ',11('| ',I2,A1,I2,1X),//)
      RETURN
      END
```

```
FCRTRAN IV G LEVEL 1, MOD 4          BLK DATA          DATE = 70191          20/49/09          PAGE 0001

      BLCCK DATA
      IMPLICIT INTEGER*2(I-N)
      COMMCN/DBALL/JSTORE(100,6),JTEMP(100,6)
      COMMCN/PRI/ITOTAL(6)
      DATA ITOTAL/6*C/
      DATA JSTCRE/600*0/,JTEMP/600*0/
      END
```

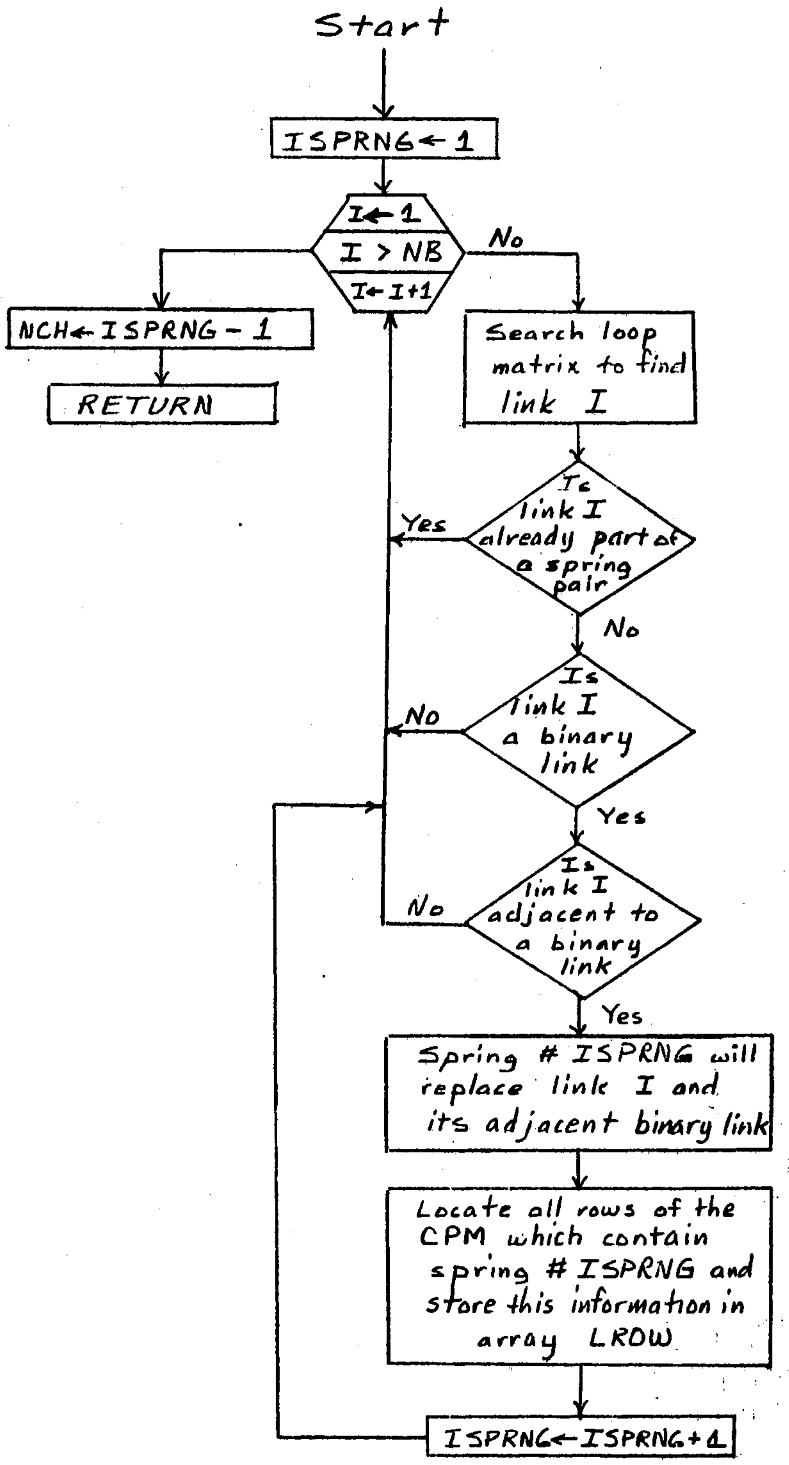

Start
ISPRNG ← 1
I ← 1
I > NB
I ← I+1
No
NCH ← ISPRNG − 1
RETURN
Search loop matrix to find link I
Is link I already part of a spring pair
Yes
No
Is link I a binary link
No
Yes
Is link I adjacent to a binary link
No
Yes
Spring # ISPRNG will replace link I and its adjacent binary link
Locate all rows of the CPM which contain spring # ISPRNG and store this information in array LROW
ISPRNG ← ISPRNG + 1

Figure 24. Flow Chart for Subroutine SPRLOC

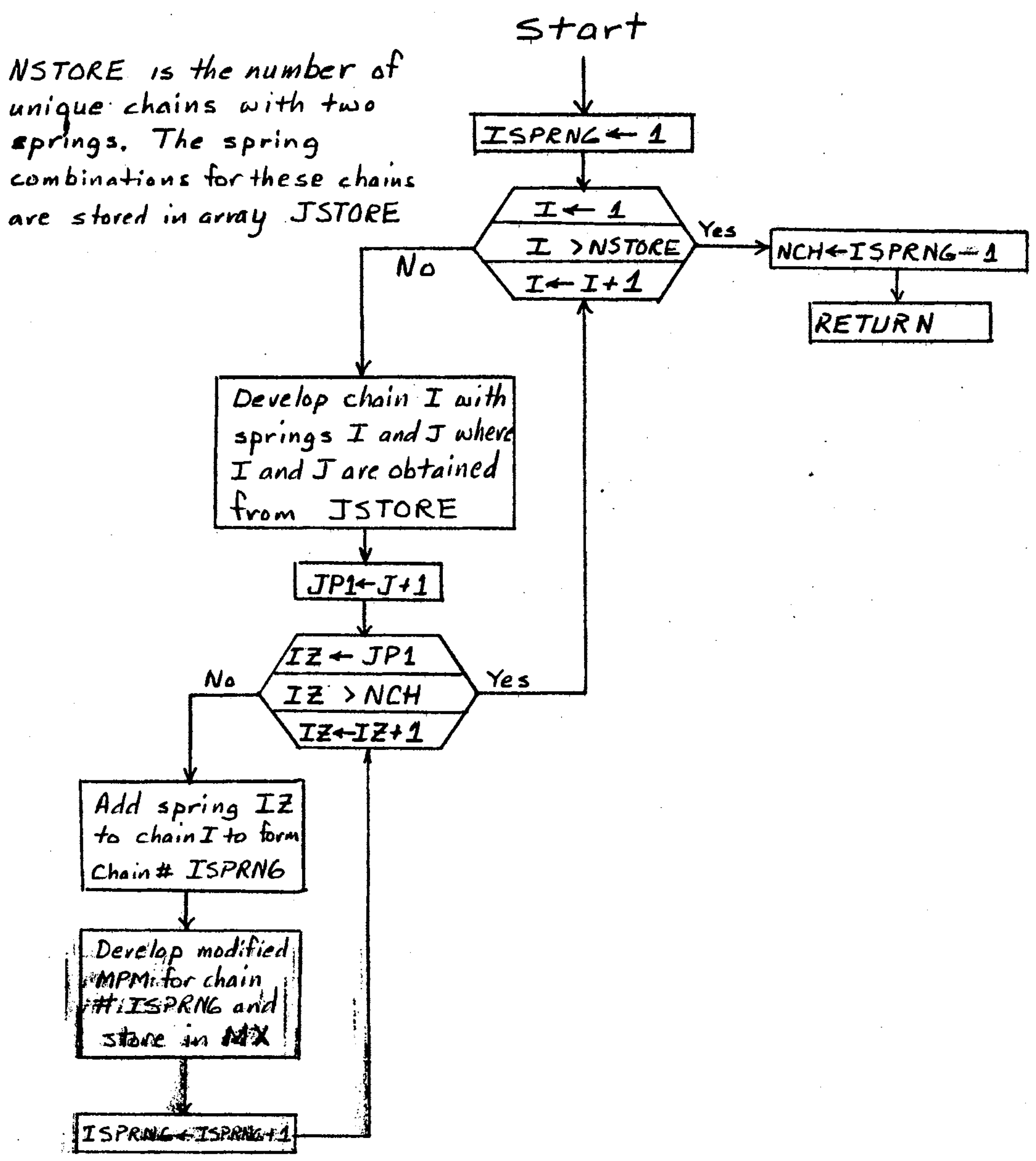

NSTORE is the number of unique chains with two springs. The spring combinations for these chains are stored in array JSTORE
Start
ISPRNG ← 1
I ← 1
I > NSTORE
I ← I+1
Yes
NCH←ISPRNG−1
RETURN
No
Develop chain I with springs I and J where I and J are obtained from JSTORE
JP1←J+1
IZ ← JP1
IZ > NCH
IZ←IZ+1
No
Yes
Add spring IZ to chain I to form Chain # ISPRNG
Develop modified MPMi for chain #ISPRNG and store in MX
ISPRNG←ISPRNG+1

Figure 25. Flow Chart for Subroutine SPRNG3

APPENDIX H

SUBROUTINES USED IN DETERMINING THE TEN-LINK CHAINS WITH PULLEYS AND BELTS

The program for determining the chains with pulleys and belts is the same as that shown in Appendix G with Subroutine SPRLOC being replaced by Subroutine PULLOC and Subroutine SPRNG1 through SPRNG4 being replaced by PUL1 through PUL4. Also, some format statements in the output subroutines require changes.

Subroutine PULLOC is shown in Table XLIX and its flow chart in Figure 26. Subroutine PUL3, which demonstrates the logic for all PUL subroutines, appears in Table L and its flow chart is found in Figure 27.

TABLE XLIX

SUBROUTINE PULLOC

```
FORTRAN IV G LEVEL 1, MOD 4          PULLOC          DATE = 70192          09/56/51          PAGE 0001

      SUBROUTINE PULLOC
      IMPLICIT INTEGER*2(I-N)
      COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMON/PUL/JPUL(10,6),JPROW(10,15)
      IPUL=1
C
C SEARCH FOR TERNARY LINKS
C
      DO 100 K=1,NB
      DO 10 I=1,NL
      NE=M(I,NCOL)
      IF(NE.EQ.0) GOTO 10
      DO 11 J=1,NE
      IF(M(I,J)-(M(I,J)/100)*100.EQ.K) GOTO 20
   11 CONTINUE
   10 CONTINUE
   20 IF(M(I,J)/100.NE.3) GOTO 100
C
C DETERMINE IF TERNARY LINK IS JOINED TO TWO BINARY LINKS
C
      DO 30 IA=1,NR
      NE=M(IA,NCOL)
      IF(NE.EQ.0) GOTO 30
      DO 40 IB=1,NE
      IF(M(IA,IB).EQ.M(I,J)) GOTO 41
   40 CONTINUE
      GOTO 30
   41 IY=IB-1
      IZ=IB+1
      IF(IB.EQ.1) IY=NE
      IY1=IY-1
      IF(IY.EQ.1) IY1=NE
      IF(IB.EQ.NE) IZ=1
      IZ1=IZ+1
      IF(IZ.EQ.NE) IZ1=1
      IF(M(IA,IY)/100.EQ.1.AND.M(IA,IZ)/100.EQ.1) GOTO 42
      GOTO 30
   42 JPUL(IPUL,1)=M(IA,IB)
      JPUL(IPUL,2)=M(IA,IY)
      JPUL(IPUL,3)=M(IA,IY1)
      JPUL(IPUL,4)=M(IA,IZ)
      JPUL(IPUL,5)=M(IA,IZ1)
      IPUL=IPUL+1
      GOTO 43
   30 CONTINUE
      GOTO 100
C
C DETERMINE IF A THIRD BINARY LINK IS CONNECTED TO THE TERNARY LINK
C
   43 DO 60 IC=1,NR
      NE=M(IC,NCOL)
      IF(NE.EQ.0) GOTO 60
      DO 50 ID=1,NE
      IF(M(IC,ID).EQ.M(I,J)) GOTO 61
   50 CONTINUE
      GOTO 60
   61 I1=ID-1
      I2=ID+1
```

TABLE XLIX (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4            PULLOC            DATE = 70192            09/56/51                PAGE 0002

            IF(ID.EQ.1) I1=NE
            I11=I1-1
            IF(I1.EQ.1) I11=NE
            IF(ID.EQ.NE) I2=1
            I21=I2+1
            IF(I2.EQ.NE)I21=1
            IF(M(IC,I1).NE.JPUL(IPUL-1,2).AND.M(IC,I1).NE.JPUL(IPUL-1,4))
           1 GOTO 70
            IF(M(IC,I2).NE.JPUL(IPUL-1,2).AND.M(IC,I2).NE.JPUL(IPUL-1,4))
           2 GOTO 71
         60 CONTINUE
         70 JPUL(IPUL-1,6)=M(IC,I1)
            IF(M(IC,I1)/100.NE.1) GOTO 100
            JPUL(IPUL,2)=M(IC,I1)
            JPUL(IPUL,3)=M(IC,I11)
            JPUL(IPUL+1,2)=M(IC,I1)
            JPUL(IPUL+1,3)=M(IC,I11)
            GOTO 72
         71 JPUL(IPUL-1,6)=M(IC,I2)
            IF(M(IC,I2)/100.NE.1) GOTO 100
            JPUL(IPUL,2)=M(IC,I2)
            JPUL(IPUL,3)=M(IC,I21)
            JPUL(IPUL+1,2)=M(IC,I2)
            JPUL(IPUL+1,3)=M(IC,I21)
         72 JPUL(IPUL,1)=M(I,J)
            JPUL(IPUL,4)=JPUL(IPUL-1,2)
            JPUL(IPUL,5)=JPUL(IPUL-1,3)
            JPUL(IPUL,6)=JPUL(IPUL-1,4)
            IPUL=IPUL+1
            JPUL(IPUL,1)=M(I,J)
            JPUL(IPUL,4)=JPUL(IPUL-2,4)
            JPUL(IPUL,5)=JPUL(IPUL-1,5)
            JPUL(IPUL,6)=JPUL(IPUL-2,2)
            IPUL=IPUL+1
        100 CONTINUE
            NCH=IPUL-1
      C
      C DETERMINE ROWS CONTAINING PULLEY
      C
            IF(NCH.EQ.0) RETURN
            DO 110 I=1,NCH
            DO 110 J=1,NR
            JPROW(I,J)=0
            NE=M(J,NCOL)
            IF(NE.EQ.0) GOTO 110
            DO 120 K=1,NE
            IF(M(J,K).EQ.JPUL(I,1)) GOTO 121
        120 CONTINUE
            GOTO 110
        121 JPROW(I,J)=J
        110 CONTINUE
            RETURN
            END
```

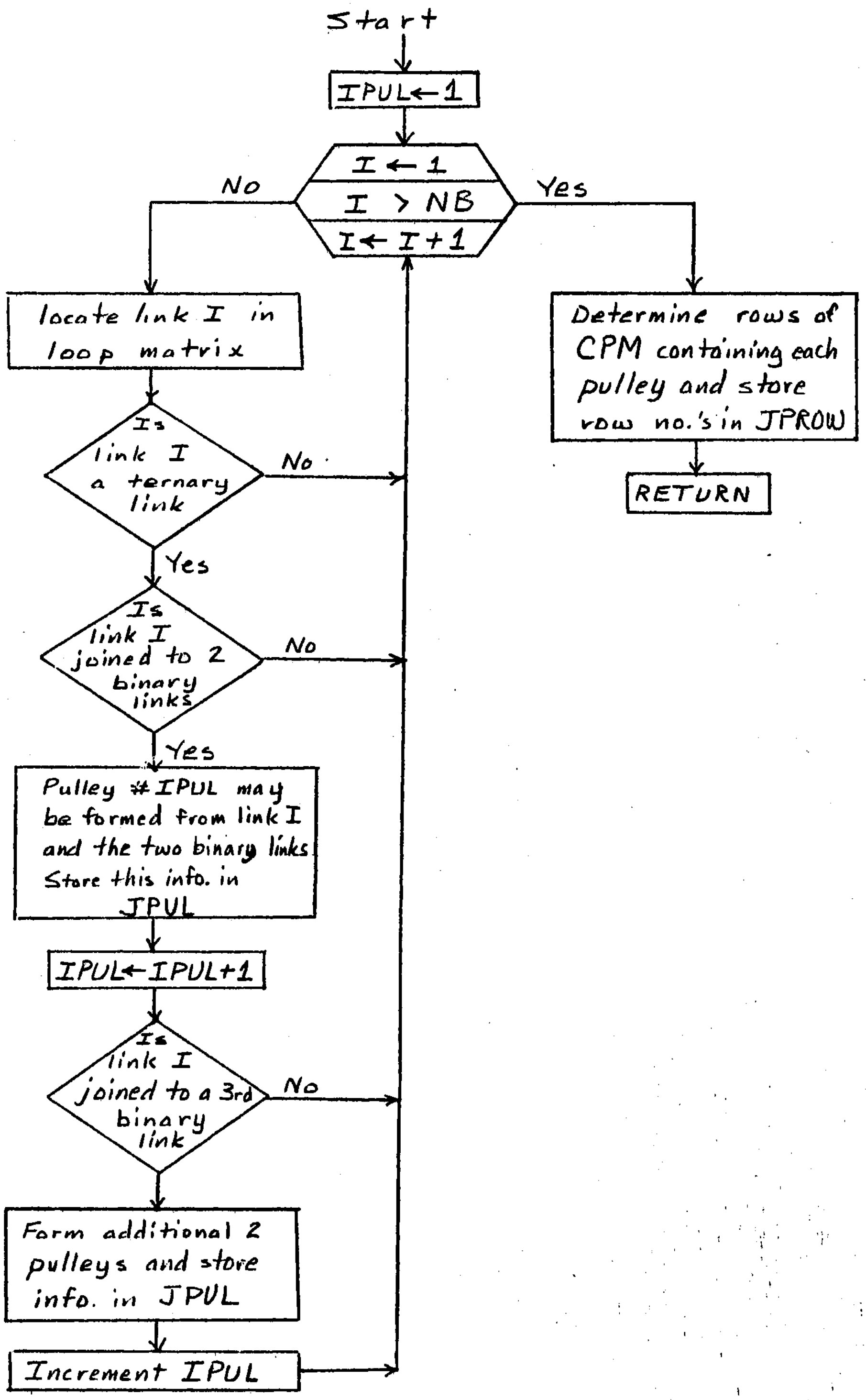

Start
IPUL←1
I←1
I > NB
I←I+1
No
Yes
locate link I in loop matrix
Is link I a ternary link
No
Yes
Is link I joined to 2 binary links
No
Yes
Pulley #IPUL may be formed from link I and the two binary links. Store this info. in JPUL
IPUL←IPUL+1
Is link I joined to a 3rd binary link
No
Form additional 2 pulleys and store info. in JPUL
Increment IPUL
Determine rows of CPM containing each pulley and store row no.'s in JPROW
RETURN

Figure 26. Flow Chart for Subroutine PULLOC

TABLE L

SUBROUTINE PUL3

```
FORTRAN IV G LEVEL 1, MOD 4            PUL3           DATE = 70192        09/56/51           PAGE 0001

            SUBROUTINE PUL3
            IMPLICIT INTEGER*2(I-N)
            COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
            COMMON/DBALL/JSTORE(100,6),JTEMP(100,6)
            COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
           * ,MZ5(15,15),MX(15,15,15)
            COMMON/PUL/JPUL(10,6),JPROW(10,15)
            COMMON/CHN/IPUL,KSTORE,KOUNT
            COMMON/LIN/LINE,IPAGE
            DIMENSION L1(5),L2(5),L3(5),L4(5)
            IPUL=1
            KOUNT=1
            KSTORE=1
            IF(LINE.GT.47) CALL PAGE
            WRITE(6,1)
          1 FORMAT(/,'0IDENTIFICATION OF CHAINS WITH THREE PULLEYS',/,1X,
           * 43('+'),/)
            LINE=LINE+5
            DO 100 I=1,NSTORE
            JTEMP(I,2)=JSTORE(I,2)
        100 JTEMP(I,1)=JSTORE(I,1)
            DO 10 IZ=1,NSTORE
            IF(IZ.EQ.1) GOTO 9
            IF(JTEMP(IZ,1).NE.JTEMP(IZ-1,1)) GOTO 9
            GOTO 8
          9 I=JTEMP(IZ,1)
            DO 91 IA=1,5
         91 L1(IA)=JPUL(I,IA)
            CALL PULADD(I,NR,NCOL,M,MZ1)
          8 J=JTEMP(IZ,2)
            DO 81 IA=1,5
         81 L2(IA)=JPUL(J,IA)
            CALL PULADD(J,NR,NCOL,MZ1,MZ2)
            JP1=J+1
            IF(JP1.GT.NCH) GOTO 10
            DO 30 K=JP1,NCH
            DO 71 IA=1,5
         71 L3(IA)=JPUL(K,IA)
            IF(L1(1).EQ.L3(1)) GOTO 30
            IF(L1(2).EQ.L3(3).OR.L1(2).EQ.L3(5)) GOTO 30
            IF(L1(4).EQ.L3(3).OR.L1(4).EQ.L3(5)) GOTO 30
            IF(L2(1).EQ.L3(1)) GOTO 30
            IF(L2(2).EQ.L3(3).OR.L2(2).EQ.L3(5)) GOTO 30
            IF(L2(4).EQ.L3(3).OR.L2(4).EQ.L3(5)) GOTO 30
            JSTORE(IPUL,1)=I
            JSTORE(IPUL,2)=J
            JSTORE(IPUL,3)=K
            KOUNT=KOUNT+1
            IF(KOUNT.GT.5) CALL CHAIN1
            CALL PULADD(K,NR,NCOL,MZ2,MZ)
      C FILL MX
            IROW=1
            DO 20 IA=1,NR
            IF(JPROW(I,IA).NE.IA.AND.JPROW(J,IA).NE.IA.AND.JPROW(K,IA).NE.IA)
           * GOTO 20
            DO 40 IB=1,NCOL
         40 MX(IPUL,IROW,IB)=MZ(IA,IB)
            IROW=IROW+1

FORTRAN IV G LEVEL 1, MOD 4            PUL3           DATE = 70192        09/56/51           PAGE 0002

         20 CONTINUE
            MX(IPUL,1,NCOL-1)=IROW
            IPUL=IPUL+1
         30 CONTINUE
         10 CONTINUE
            IPUL=IPUL-1
            IF(IPUL.GE.KSTORE) CALL CHAIN1
            NCH=IPUL
            NSTORE=NCH
            RETURN
            END
```

TABLE L (Continued)

```
FORTRAN IV G LEVEL 1, MOD 4              PULADD              DATE = 70192              14/06/25              PAGE 0001

      SUBROUTINE PULADD(I,NR,NCOL,MY,MZ)
      IMPLICIT INTEGER*2(I-N)
      COMMON/PUL/JPUL(10,6),JPROW(10,15)
      DIMENSION MY(15,20),MZ(15,20)
      L1=JPUL(I,1)
      L2=JPUL(I,2)
      L3=JPUL(I,4)
      DO 10 J=1,NR
      IF(JPROW(I,J).NE.J) GOTO 2
      NE=MY(J,NCOL)
      DO 20 K=1,NE
      IF(MY(J,K).EQ.L1) GOTO 30
   20 CONTINUE
   30 ISTOP=K
      ICOL=1
      MZ(J,1)=1111
   31 ICOL=ICOL+1
      K=K+1
      IF(K.GT.NE)K=1
      IF(K.EQ.ISTOP) GOTO 3
      MZ(J,ICOL)=MY(J,K)
      IF(MY(J,K).EQ.L2.OR.MY(J,K).EQ.L3) MZ(J,ICOL)=1211
      GOTO 31
    3 K=NE+1
      DO 4 KK=K,NCOL
    4 MZ(J,KK)=MY(J,KK)
      GOTO 10
    2 DO 5 K=1,NCOL
    5 MZ(J,K)=MY(J,K)
   10 CONTINUE
      RETURN
      END
```

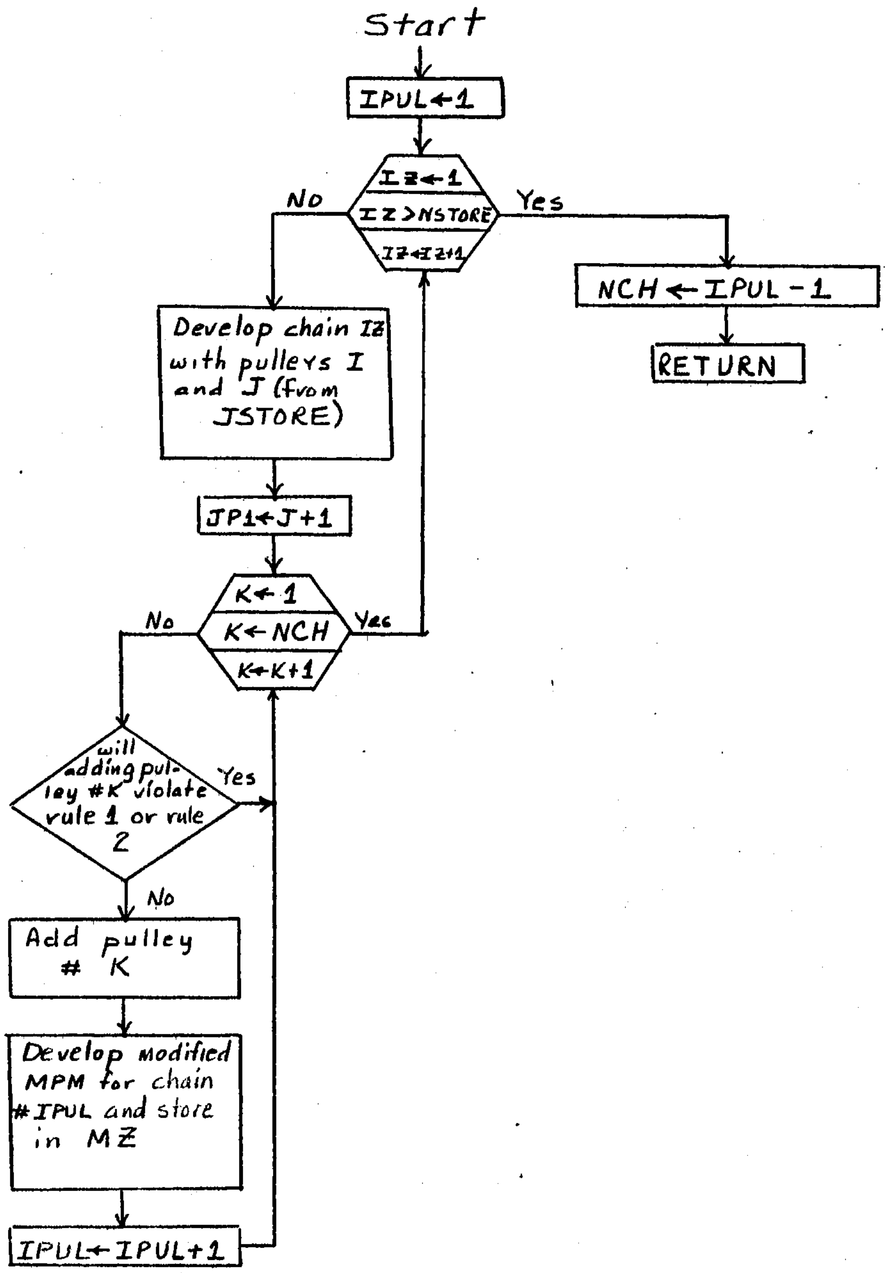

Start
IPUL←1
IZ←1
IZ>NSTORE
IZ←IZ+1
No
Yes
NCH←IPUL-1
RETURN
Develop chain IZ with pulleys I and J (from JSTORE)
JP1←J+1
K←1
K←NCH
K←K+1
No
Yes
will adding pulley #K violate rule 1 or rule 2
Yes
No
Add pulley # K
Develop modified MPM for chain #IPUL and store in MZ
IPUL←IPUL+1

Figure 27. Flow Chart for Subroutine PUL3

APPENDIX I

SUBROUTINES USED IN DETERMINING THE TEN-LINK CHAINS WITH CAM PAIRS

As with the program for determining chains with pulleys (Appendix H), this program follows the basic form shown in Appendix G. Subroutine CAMLOC, which locates possible cam pairs, appears in Table LI and its flow chart in Figure 28. Subroutine CAM3, which develops the chains with 3 cam pairs, is shown in Table LII and its flow chart in Figure 29.

TABLE LI

SUBROUTINE CAMLOC

```
FORTRAN IV G LEVEL 1, MOD 4            CAMADD             DATE = 70193          14/26/56             PAGE 0001

                 SUBROUTINE CAMADD(K,NR,NCOL,MY,MZ)
                 IMPLICIT INTEGER*2(I-N)
                 COMMON/CAL/JCAM(8,3),JCROW(8,15)
                 DIMENSION MY(15,15),MZ(15,15)
                 L1=JCAM(K,1)
                 DO 20 I=1,NR
                 IF(JCROW(K,I).NE.I) GOTO 25
                 NE=MY(I,NCOL)
                 DO 30 J=1,NE
                 IF(MY(I,J).EQ.L1) GOTO 40
              30 CONTINUE
              40 ISTOP=J
                 ICOL=1
                 MZ(I,ICOL)=1111
              41 ICOL=ICOL+1
                 J=J+1
                 IF(J.GT.NE)J=1
                 IF(J.EQ.ISTOP) GOTO 42
                 MZ(I,ICOL)=MY(I,J)
                 GOTO 41
              42 J=NE+1
              43 DO 50 L=J,NCOL
              50 MZ(I,L)=MY(I,L)
                 GOTO 20
              25 J=1
                 GOTO 43
              20 CONTINUE
                 RETURN
                 END

FORTRAN IV G LEVEL 1, MOD 4            CAMLOC             DATE = 70193          14/26/56             PAGE 0001

                 SUBROUTINE CAMLOC
                 IMPLICIT INTEGER*2(I-N)
                 COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
                 COMMON/CAL/JCAM(8,3),JCROW(8,15)
                 ICAM=1
            C
            C SEARCH FOR BINARY LINKS
            C
                 DO 100 K=1,NB
                 DO 10 I=1,NL
                 JCROW(ICAM,I)=0
                 NE=M(I,NCOL)
                 DO 10 J=1,NE
                 IF(M(I,J)-(M(I,J)/100)*100.EQ.K) GOTO 20
              10 CONTINUE
              20 IF(M(I,J)/100.NE.1) GOTO 100
                 JCAM(ICAM,1)=M(I,J)
                 IX=J+1
                 IY=J-1
                 IF(J.EQ.NE)IX=1
                 IF(J.EQ.1)IY=NE
                 JCAM(ICAM,2)=M(I,IY)
                 JCAM(ICAM,3)=M(I,IX)
            C
            C DETERMINE ROWS CONTAINING BINARY LINK
            C
                 JCROW(ICAM,I)=I
                 IP1=I+1
                 DO 30 IA=IP1,NR
                 JCROW(ICAM,IA)=0
                 NE=M(IA,NCOL)
                 IF(NE.EQ.0) GOTO 30
                 DO 31 IB=1,NE
                 IF(M(IA,IB).EQ.JCAM(ICAM,1)) GOTO 32
              31 CONTINUE
                 GOTO 30
              32 JCROW(ICAM,IA)=IA
              30 CONTINUE
                 ICAM=ICAM+1
             100 CONTINUE
                 NCH=ICAM-1
                 RETURN
                 END
```

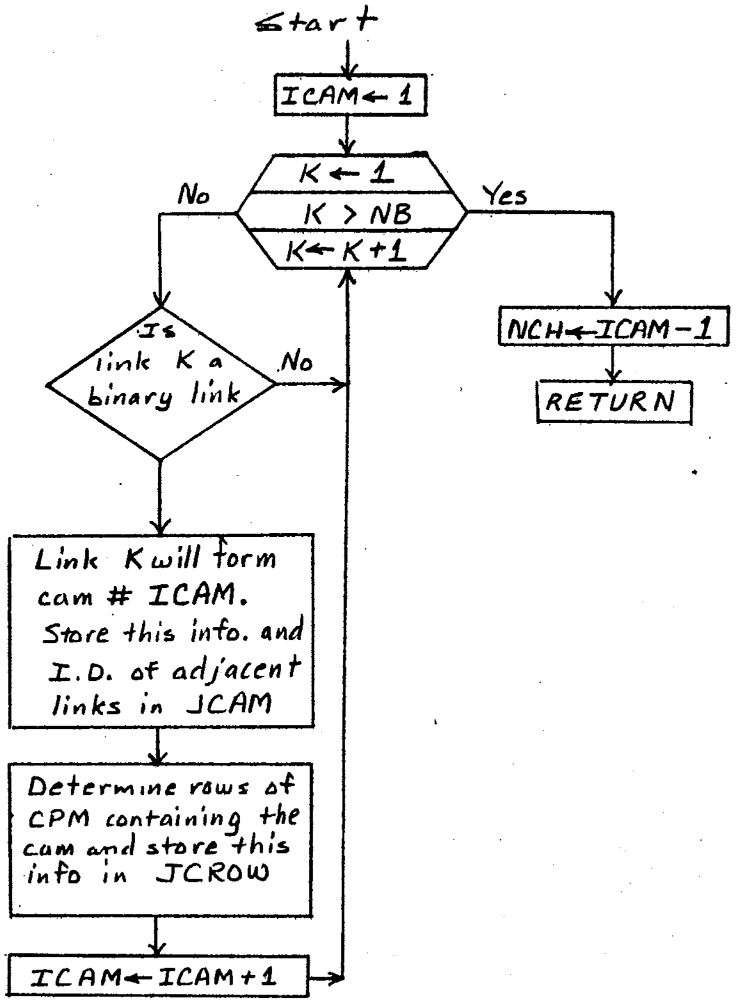

Start
ICAM ← 1
K ← 1
K > NB
K ← K+1
No
Yes
Is link K a binary link
No
NCH ← ICAM−1
RETURN
Link K will form cam # ICAM. Store this info. and I.D. of adjacent links in JCAM
Determine rows of CPM containing the cam and store this info in JCROW
ICAM ← ICAM+1

Figure 28. Flow Chart for Subroutine CAMLOC

TABLE LII

SUBROUTINE CAM3

```
FORTRAN IV G LEVEL 1, MOD 4          CAM3          DATE = 70193          14/26/56          PAGE 0001

            SUBROUTINE CAM3
            IMPLICIT INTEGER*2(I-N)
            COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
            COMMON/DBALL/JSTORE(100,6),JTEMP(100,6)
            COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
           *,MZ5(15,15),MX(50,15,15)
            COMMON/CAL/JCAM(8,3),JCROW(8,15)
            COMMON/CHN/ICAM,KSTORE,KOUNT
            COMMON/LIN/LINE,IPAGE
            ICAM=1
            KOUNT=1
            KSTORE=1
            IF(LINE.GT.47) CALL PAGE
            WRITE(6,1)
          1 FORMAT(/,'0IDENTIFICATION OF CHAINS WITH THREE CAM PAIRS',/,1X,
           *45('*'),/)
            LINE=LINE+5
            DO 100 I=1,NSTORE
            JTEMP(I,2)=JSTORE(I,2)
        100 JTEMP(I,1)=JSTORE(I,1)
            DO 10 IZ=1,NSTORE
            IF(IZ.EQ.1) GOTO 9
            IF(JTEMP(IZ,1).NE.JTEMP(IZ-1,1)) GOTO 9
            GOTO 8
          9 I=JTEMP(IZ,1)
            I1=JCAM(I,1)
            CALL CAMADD(I,NR,NCOL,M,MZ1)
          8 J=JTEMP(IZ,2)
            J1=JCAM(J,1)
            CALL CAMADD(J,NR,NCOL,MZ1,MZ2)
            JP1=J+1
            IF(JP1.GT.NCH) GOTO 10
            DO 30 K=JP1,NCH
            K2=JCAM(K,2)
            K3=JCAM(K,3)
            IF(I1.EQ.K2.OR.I1.EQ.K3) GOTO 30
            IF(J1.EQ.K2.OR.J1.EQ.K3) GOTO 30
            JSTORE(ICAM,1)=I
            JSTORE(ICAM,2)=J
            JSTORE(ICAM,3)=K
            KOUNT=KOUNT+1
            IF(KOUNT.GT.5) CALL CHAIN1
            CALL CAMADD(K,NR,NCOL,MZ2,MZ)
      C FILL MX
            IROW=1
            DO 20 IA=1,NR
            IF(JCROW(I,IA).NE.IA.AND.JCROW(J,IA).NE.IA.AND.JCROW(K,IA).NE.IA)
           * GOTO 20
            DO 40 IB=1,NCOL
         40 MX(ICAM,IROW,IB)=MZ(IA,IB)
            IROW=IROW+1
         20 CONTINUE
            MX(ICAM,1,NCOL-1)=IROW
            ICAM=ICAM+1
         30 CONTINUE
         10 CONTINUE
            ICAM=ICAM-1
            IF(ICAM.GE.KSTORE) CALL CHAIN1

FORTRAN IV G LEVEL 1, MOD 4          CAM3          DATE = 70193          14/26/56          PAGE 0002

            NCH=ICAM
            NSTORE=NCH
            RETURN
            END
```

NOTE: Subroutine CAMADD appears in Table LVI.

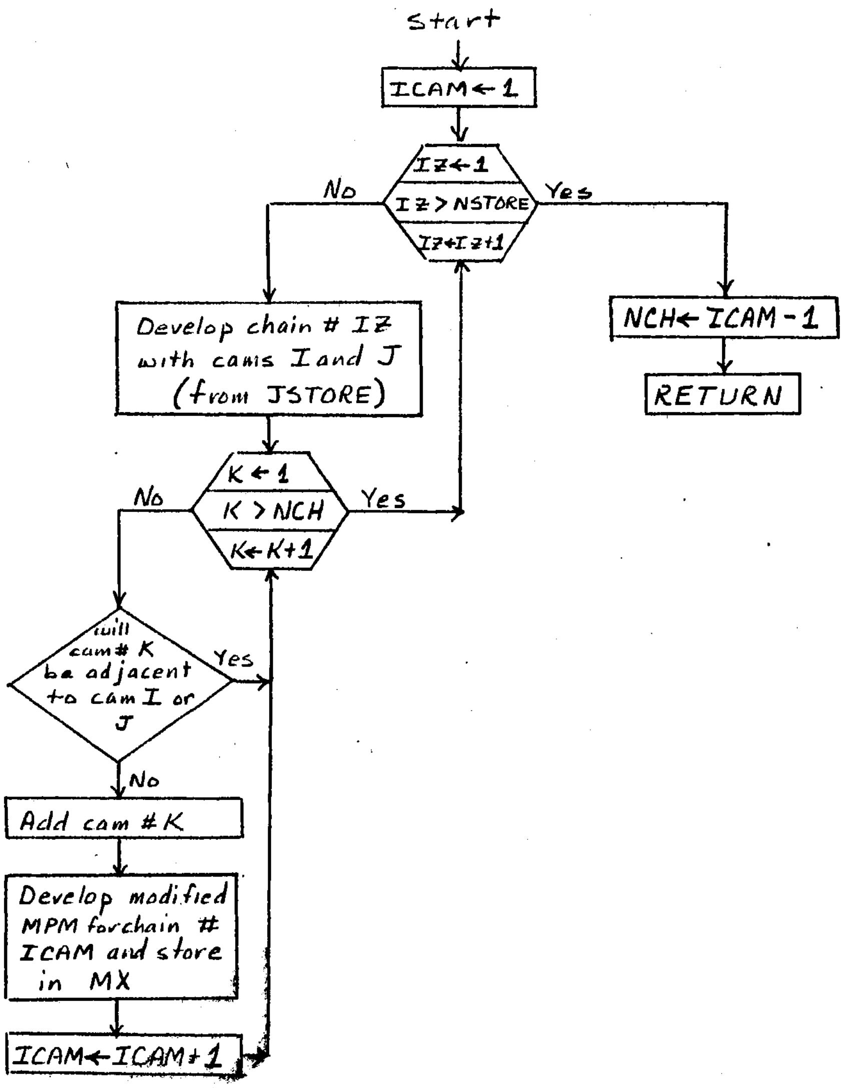

Start
ICAM ← 1
IZ ← 1
IZ > NSTORE
IZ ← IZ+1
No
Yes
NCH ← ICAM - 1
RETURN
Develop chain # IZ with cams I and J (from JSTORE)
K ← 1
K > NCH
K ← K+1
No
Yes
will cam # K be adjacent to cam I or J
Yes
No
Add cam # K
Develop modified MPM for chain # ICAM and store in MX
ICAM ← ICAM+1

Figure 29. Flow Chart for Subroutine CAM3

APPENDIX J

SUBROUTINES USED IN DETERMINING THE TEN-LINK CHAINS WITH PRISM PAIRS

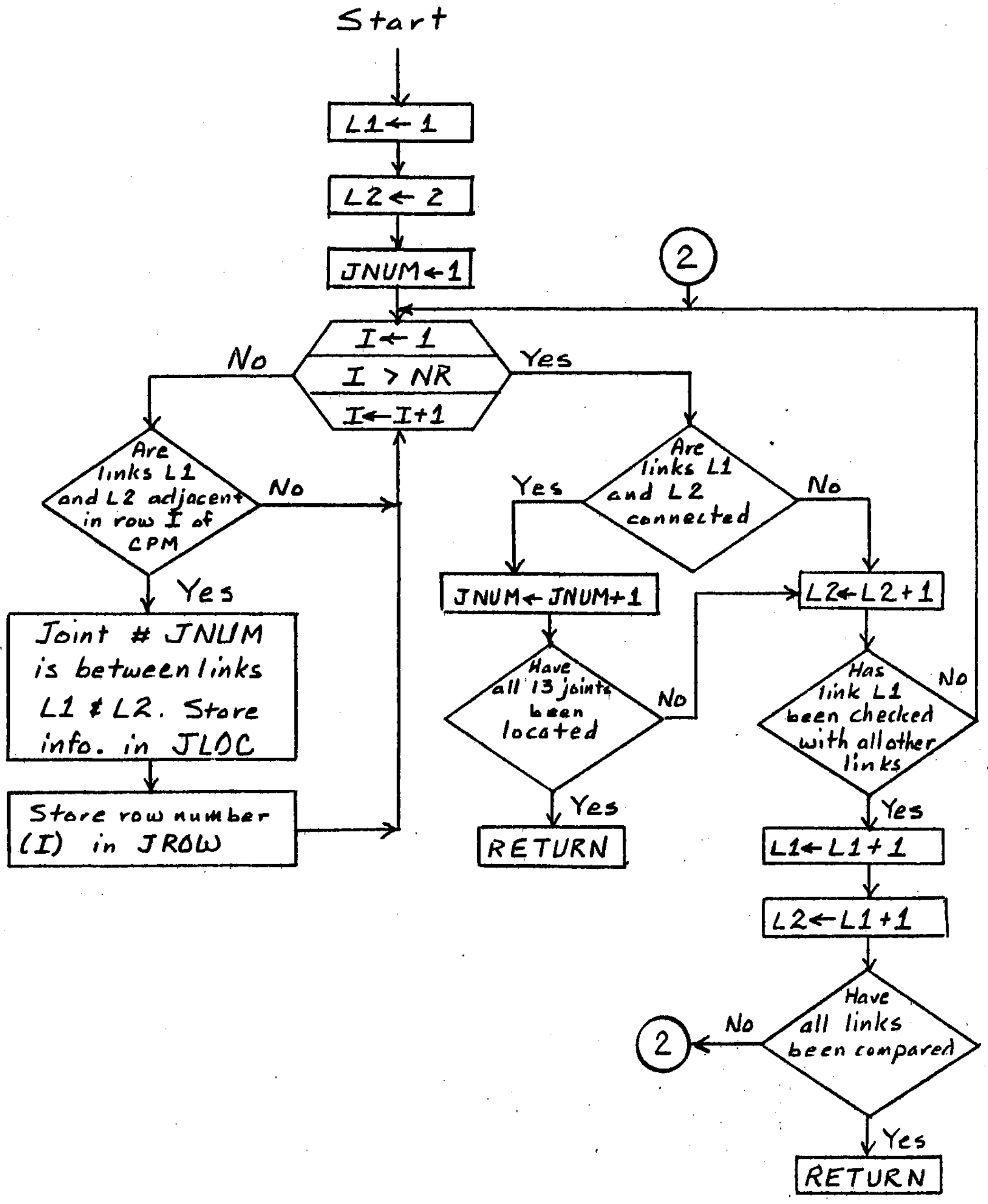
The program for prism pairs follows the same form as all previous programs. Subroutines JOINT (which locates all joints) and PRISM2 (which adds 2 prism pairs) are shown in Tables LIII and LIV respectively. Their flow charts appear in Figures 30 and 31.

TABLE LIII

SUBROUTINE JOINT

```
FORTRAN IV G LEVEL 1, MOD 4            JOINT            DATE = 70195          02/53/33             PAGE 0001

      SUBROUTINE JOINT
      IMPLICIT INTEGER*2(I-N)
      COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMON/JO1/JLOC(13,2),JROW(13,15)
      NCH=13
      L1=1
      L2=2
      JNUM=1
      DO 1 I=1,13
      DO 1 J=1,15
    1 JROW(I,J)=0
C
C SEARCH FOR EITHER LINK L1 OR LINK L2.
C
    2 ISUM=0
      DO 10 I=1,NR
      NE=M(I,NCOL)
      IF(NE.EQ.0) GOTO 10
      DO 20 J=1,NE
      IF(M(I,J)-(M(I,J)/100)*100.EQ.L1) GOTO 30
      IF(M(I,J)-(M(I,J)/100)*100.EQ.L2) GOTO 40
   20 CONTINUE
      GOTO 10
   30 IF(M(I,J+1)-(M(I,J+1)/100)*100.EQ.L2) GOTO 50
      IF(J.NE.1) GOTO 10
      IF(M(I,NE)-(M(I,NE)/100)*100.EQ.L2) GOTO 60
      GOTO 10
   40 IF(M(I,J+1)-(M(I,J+1)/100)*100.EQ.L1) GOTO 70
      IF(J.NE.1) GOTO 10
      IF(M(I,NE)-(M(I,NE)/100)*100.EQ.L1) GOTO 80
      GOTO 10
C
C LINKS L1 AND L2 ARE ADJACENT IN ROW I.
C
   50 JLOC(JNUM,1)=M(I,J)
      JLOC(JNUM,2)=M(I,J+1)
      GOTO 90
   60 JLOC(JNUM,1)=M(I,J)
      JLOC(JNUM,2)=M(I,NE)
      GOTO 90
   70 JLOC(JNUM,1)=M(I,J+1)
      JLOC(JNUM,2)=M(I,J)
      GOTO 90
   80 JLOC(JNUM,1)=M(I,NE)
      JLOC(JNUM,2)=M(I,J)
   90 JROW(JNUM,I)=I
      ISUM=1
   10 CONTINUE
      IF(ISUM.EQ.0) GOTO 100
      JNUM=JNUM+1
      IF(JNUM.GT.13) RETURN
  100 L2=L2+1
      IF(L2.GT.NB) GOTO 110
      GOTO 2
  110 L1=L1+1
      L2=L1+1
      IF(L2.GT.NB) RETURN
      GOTO 2
```



Start
L1 ← 1
L2 ← 2
JNUM ← 1
2
I ← 1
I > NR
I ← I+1
No
Yes
Are links L1 and L2 adjacent in row I of CPM
No
Yes
Joint # JNUM is between links L1 & L2. Store info. in JLOC
Store row number (I) in JROW
Are links L1 and L2 connected
Yes
No
JNUM ← JNUM+1
L2 ← L2+1
Have all 13 joints been located
No
Yes
RETURN
Has link L1 been checked with all other links
No
Yes
L1 ← L1+1
L2 ← L1+1
Have all links been compared
No
2
Yes
RETURN

Figure 30. Flow Chart for Subroutine JOINT

TABLE LIV

SUBROUTINE PRISM2

```
FORTRAN IV G LEVEL 1, MOD 4              PRISM2             DATE = 70195           02/53/33                 PAGE 0001

                  SUBROUTINE PRISM2
                  IMPLICIT INTEGER*2(I-N)
                  COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
                  COMMON/DBALL/JSTORE(100,6),JTEMP(100,6)
                  COMMON/PRISM/MZ(15,15),MZ1(15,15),MX(100,15,15)
                  COMMON/JOI/LPAIR(13,2),LROW(13,15)
                  COMMON/CHN/IPRISM,KSTORE,KOUNT
                  COMMON/LIN/LINE,IPAGE
                  IPRISM=1
                  KOUNT=1
                  KSTORE=1
                  IF(LINE.GT.47) CALL PAGE
                  WRITE(6,1)
                1 FORMAT(/,'0IDENTIFICATION OF CHAINS WITH TWO PRISM PAIRS',/,1X,
                 *45('*'),/)
                  LINE=LINE+5
                  DO 100 I=1,NSTORE
              100 JTEMP(I,1)=JSTORE(I,1)
                  DO 10 IZ=1,NSTORE
                9 I=JTEMP(IZ,1)
                  CALL LOCATE(I,NR,NCOL,M,MZ1)
                  IP1=I+1
                  IF(IP1.GT.NCH) GOTO 10
                  DO 30 J=IP1,NCH
                  JSTORE(IPRISM,1)=I
                  JSTORE(IPRISM,2)=J
                  KOUNT=KOUNT+1
                  IF(KOUNT.GT.5) CALL CHAIN1
                  CALL LOCATE(J,NR,NCOL,MZ1,MZ)
            C FILL MX
                  IROW=1
                  DO 20 IA=1,NR
                  IF(LROW(I,IA).EQ.0.AND.LROW(J,IA).EQ.0) GOTO 20
                  DO 40 IB=1,NCOL
               40 MX(IPRISM,IROW,IB)=MZ(IA,IB)
                  IROW=IROW+1
               20 CONTINUE
                  MX(IPRISM,1,NCOL-1)=IROW
                  IPRISM=IPRISM+1
               30 CONTINUE
               10 CONTINUE
                  IPRISM=IPRISM-1
                  IF(IPRISM.GE.KSTORE) CALL CHAIN1
                  NCH=IPRISM
                  RETURN
                  END
```

```
FORTRAN IV G LEVEL 1, MOD 4              LOCATE             DATE = 70195           02/53/33                 PAGE 0001

                  SUBROUTINE LOCATE(JN,NR,NCOL,MY,MZ)
                  IMPLICIT INTEGER*2(I-N)
                  COMMON/JOI/JLOC(13,2),JROW(13,15)
                  DIMENSION MY(15,20),MZ(15,20)
                  DO 10 I=1,NR
                  IF(JROW(JN,I).EQ.0) GOTO 18
            C
            C SEARCH FOR LINKS INCIDENT TO JOINT JN
            C
                  NE=MY(I,NCOL)
                  DO 20 J=1,NE
                  IF(MY(I,J).EQ.JLOC(JN,1)) GOTO 40
                  IF(MY(I,J).EQ.JLOC(JN,2)) GOTO 50
               20 CONTINUE
               40 IF(MY(I,J+1).EQ.JLOC(JN,2)) GOTO 60
                  GOTO 70
               50 IF(MY(I,J+1).EQ.JLOC(JN,1)) GOTO 60
                  GOTO 70
            C
            C INSERT PRISM PAIR IN PROPER LOCATION
            C
               70 KS=NE
                  J=0
                  GOTO 61
               60 KS=J
               61 MZ(I,1)=1011
                  NE=NE+1
                  J=J+1
                  DO 62 K=2,NE
                  MZ(I,K)=MY(I,J)
                  J=J+1
                  IF(J.EQ.KS+1) GOTO 63
                  IF(J.EQ.NE) J=1
               62 CONTINUE
               63 K=K+1
                  DO 64 IK=K,NCOL
               64 MZ(I,IK)=0
                  MZ(I,NCOL)=NE
                  GOTO 10
               18 DO 19 J=1,NCOL
               19 MZ(I,J)=MY(I,J)
               10 CONTINUE
                  RETURN
                  END
```

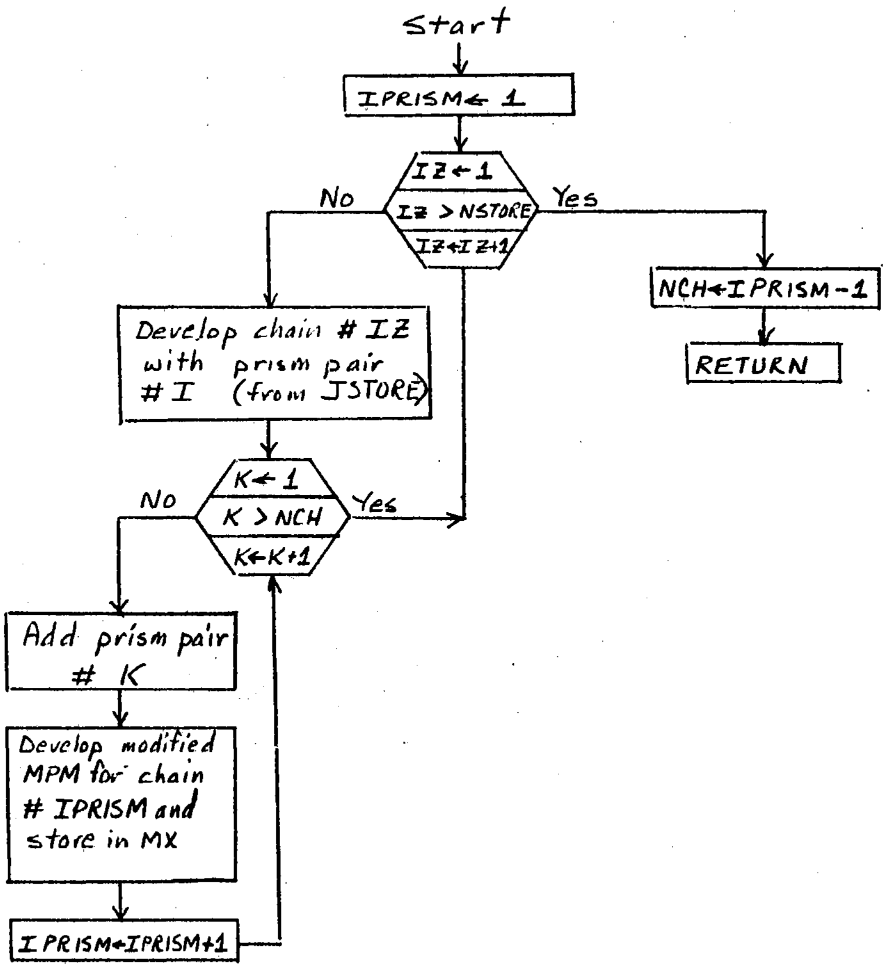


Figure 31. Flow Chart for Subroutine PRISM2

APPENDIX K

SUBROUTINES USED IN DETERMINING THE TEN-LINK CHAINS WITH DOUBLE JOINTS

This program follows the same form as all previous programs. Subroutine DBLOC (Table LV) locates all possible double joints. The flow chart for Subroutine DBLOC appears in Figure 32. Subroutine DBL2, which forms modified MPM's for chains with two double joints, is shown in Table LVI and its flow chart in Figure 33. In addition, Subroutines LNKCNT and MOBCK, which check for mobility of the chain by applying rules 1 and 2 of Chapter X, are shown in Table LVII. Their flow charts appear in Figure 34.

TABLE LV

SUBROUTINE DBLOC

```
FORTRAN IV G LEVEL 1, MOD 4          DBLOC          DATE = 70199          08/35/08          PAGE 0001

      SUBROUTINE DBLOC
      IMPLICIT INTEGER*2(I-N)
      COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMON/DBL/JLOC(20,3),JOROW(20,15),JROW(20,15)
      DIMENSION LSTORE(5)
C SEARCH FOR BASE LINKS
      NCH=1
      DO 10 LN=1,NB
      DO 11 I=1,NL
      NE=M(I,NCOL)
      DO 11 J=1,NE
      IF(M(I,J)-(M(I,J)/100)*100.NE.LN) GOTO 11
      IF(M(I,J)/100.LT.3) GOTO 10
      GOTO 12
   11 CONTINUE
C LINK LN IS A BASE LINK.  FIND ALL LINKS JOINED TO BASE LINK
   12 IBASE=M(I,J)
      DO 13 IS=1,5
   13 LSTORE(IS)=0
      IS=1
   20 IX=J-1
      IY=J+1
      IF(J.EQ.1) IX=NE
      IF(J.EQ.NE)IY=1
      DO 21 IA=1,IS
      IF(M(I,IX).EQ.LSTORE(IA)) GOTO 22
   21 CONTINUE
      LSTORE(IS)=M(I,IX)
      IS=IS+1
      IF(IS.GT.IBASE/100) GOTO 100
   22 DO 23 IB=1,IS
      IF(M(I,IY).EQ.LSTORE(IB)) GOTO 30
   23 CONTINUE
      LSTORE(IS)=M(I,IY)
      IS=IS+1
      IF(IS.GT.IBASE/100) GOTO 100
   30 I1=I+1
      DO 36 I=I1,NR
      NE=M(I,NCOL)
      IF(NE.EQ.0) GOTO 36
      DO 35 J=1,NE
      IF(M(I,J).EQ.IBASE) GOTO 20
   35 CONTINUE
   36 CONTINUE
C ALL LINKS JOINED TO BASE LINK HAVE BEEN FOUND.  DETERMINE ALL COMBINATIONS
C OF THESE LINKS TAKEN TWO AT A TIME.
  100 IS=IS-1
      ISM1=IS-1
      DO 110 IA=1,ISM1
      IAP1=IA+1
      DO 110 IB=IAP1,IS
C INSURE THAT AT LEAST FOUR LINKS WILL REMAIN IN EACH LOOP CONTAINING THE
C DOUBLE JOINT
      DO 101 JA=1,NR
      IF(M(JA,NCOL).GT.4) GOTO 101
      ISUM=0
      DO 102 JB=1,4
      IZ=M(JA,JB)

FORTRAN IV G LEVEL 1, MOD 4          DBLOC          DATE = 70199          08/35/08          PAGE 0002

  102 IF(IZ.EQ.IBASE.OR.IZ.EQ.LSTORE(IA).OR.IZ.EQ.LSTORE(IB))ISUM=ISUM+1
      IF(ISUM.EQ.3) GOTO 110
  101 CONTINUE
      JLOC(NCH,1)=IBASE
      JLOC(NCH,2)=LSTORE(IA)
      JLOC(NCH,3)=LSTORE(IB)
C LOCATE ALL ROWS INVOLVING JOINT NCH.
      JN=NCH
      DO 132 I=1,NR
      JOROW(JN,I)=0
      JROW(JN,I)=0
      NE=M(I,NCOL)
      IF(NE.EQ.0) GOTO 132
      DO 130 J=1,NE
      IF(M(I,J).NE.JLOC(JN,1)) GOTO 130
      IX=J-1
      IY=J+1
      IF(J.EQ.1) IX=NE
      IF(J.EQ.NE) IY=1
      IF(M(I,IX).EQ.JLOC(JN,2).AND.M(I,IY).EQ.JLOC(JN,3)) GOTO 133
      IF(M(I,IX).EQ.JLOC(JN,3).AND.M(I,IY).EQ.JLOC(JN,2)) GOTO 133
      IF(M(I,IX).EQ.JLOC(JN,2).OR.M(I,IX).EQ.JLOC(JN,3)) GOTO 131
      IF(M(I,IY).EQ.JLOC(JN,2).OR.M(I,IY).EQ.JLOC(JN,3)) GOTO 131
      GOTO 130
  133 JROW(JN,I)=I
  131 JOROW(JN,I)=I
  130 CONTINUE
  132 CONTINUE
      CALL MOBCK(NCH,M,NR,NCOL,MOB)
      IF(MOB.EQ.0) GOTO 110
      NCH=NCH+1
  110 CONTINUE
   10 CONTINUE
      NCH=NCH-1
      RETURN
      END
```

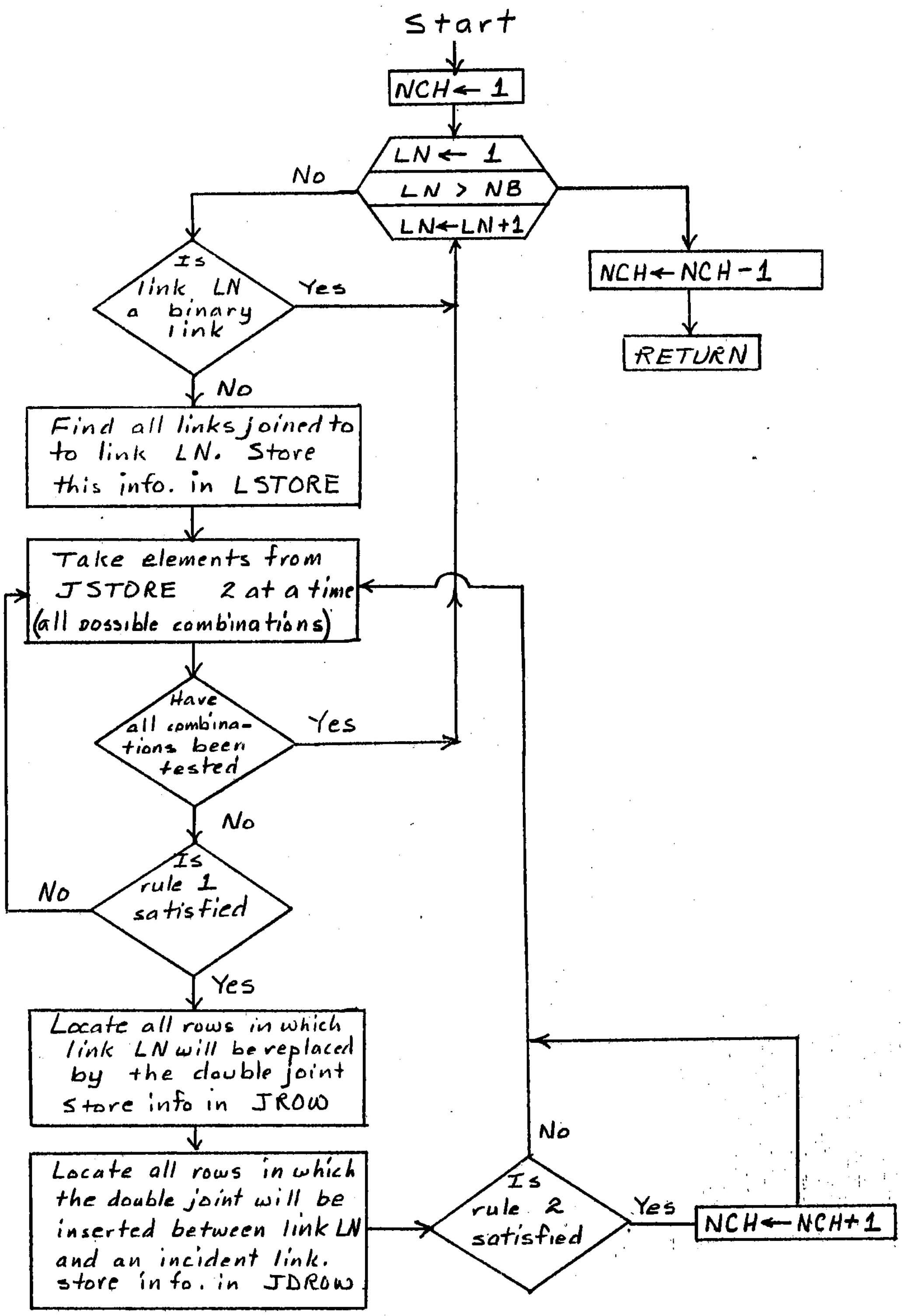

Start
NCH ← 1
LN ← 1
LN > NB
LN ← LN+1
No
NCH ← NCH−1
RETURN
Is link LN a binary link
Yes
No
Find all links joined to to link LN. Store this info. in LSTORE
Take elements from JSTORE 2 at a time (all possible combinations)
Have all combinations been tested
Yes
No
Is rule 1 satisfied
No
Yes
Locate all rows in which link LN will be replaced by the double joint Store info in JROW
Locate all rows in which the double joint will be inserted between link LN and an incident link. store info. in JDROW
Is rule 2 satisfied
No
Yes
NCH ← NCH+1

Figure 32. Flow Chart for Subroutine DBLOC

TABLE LVI

SUBROUTINE DBL2

```
FORTRAN IV G LEVEL 1, MOD 4          DBL2          DATE = 70199          08/35/08          PAGE 0001

          SUBROUTINE DBL2
          IMPLICIT INTEGER*2(I-N)
          COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
          COMMON/DBALL/JSTORE(100,6),JTEMP(100,6)
          COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
         * ,MZ5(15,15),MX(100,15,15)
          COMMON/DBL/JLOC(20,3),JCRCW(20,15),JROW(20,15)
          COMMUN/CHN/IDBL,KSTORE,KOUNT
          COMMON/DBL6A/I1(3),J1(3),K1(3),L1(3),M1(3),N1(3)
          COMMON/LIN/LINE,IPAGE
          IDBL=1
          KCUNT=1
          KSTORE=1
          IF(LINE.GT.47) CALL PAGE
          WRITE(6,1)
        1 FORMAT(/,'OIDENTIFICATICN CF CHAINS WITH TWO DOUBLE JOINTS',/,1X,
         * 47('*'),/)
          LINE=LINE+5
          DO 100 I=1,NSTORE
      100 JTEMP(I,1)=JSTORE(I,1)
          DO 10 IZ=1,NSTCRE
        9 I=JTEMP(IZ,1)
          CALL CBLADD(I,NR,NCOL,M,MZ1)
          IP1=I+1
          IF(IP1.GT.NCH) GOTO 10
          DO 90 IA=1,3
       90 I1(IA)=JLOC(I,IA)
          DO 30 J=IP1,NCH
          ISUM=C
          DO 91 IA=1,3
          J1(IA)=JLCC(J,IA)
          DO 91 IB=1,3
       91 IF(J1(IA).EQ.I1(IB)) ISUM=ISUM+1
          IF(ISUM.GT.1) GOTO 30
          CALL LNKCNT(MZ1,J,IPASS)
          IF(IPASS.EQ.1) GOTO 30
          CALL MOBCK(J,MZ1,NR,NCOL,MOB)
          IF(MOB.EC.0) GOTO 30
          JSTORE(IDBL,1)=I
          JSTORF(IDBL,2)=J
          KCUNT=KOUNT+1
          IF(KOUNT.GT.5) CALL CHAIN1
          CALL CBLADD(J,NR,NCOL,MZ1,MZ)
    C FILL MX
          IROW=1
          DO 20 IA=1,NR
          IF(JOROW(I,IA).NE.IA.AND.JOROW(J,IA).NE.IA) GOTO 20
          CALL MXFILL(IDBL,IA,IROW)
       20 CCNTINUE
          MX(IDBL,1,NCOL-1)=IRCW
          IDBL=IDBL+1
       30 CCNTINUE
       10 CONTINUE
          IDBL=IDBL-1
          IF(IDBL.GE.KSTORE) CALL CHAIN1
          NCH=IDBL
          NSTORE=NCH
          RETURN
```

TABLE LVI (Continued)

```
FCRTRAN IV G LEVEL 1, MOD 4            DBLADD          DATE = 70199         08/35/08              PAGE 0001

            SUBROUTINE DBLADD(I,NR,NCCL,MY,MZ)
            IMPLICIT INTEGER*2(I-N)
            COMMON/DBL/JLOC(20,3),JOROW(20,15),JROW(20,15)
            DIMENSION MY(15,20),MZ(15,20)
            IBASE=JLOC(I,1)
            DO 10 IA=1,NR
            IF(JOROW(I,IA).NE.IA) GOTC 11
            NE=MY(IA,NCOL)
            DO 20 IB=1,NE
            IF(MY(IA,IB).EQ.IBASE) GOTC 21
         20 CONTINUE
         21 IX=IB-1
            IY=IB+1
            IF(IB.EQ.1) IX=NE
            IF(IB.EQ.NE) IY=1
            IF(MY(IA,IX).EQ.JLOC(I,2).AND.MY(IA,IY).EQ.JLCC(I,3)) GOTO 30
            IF(MY(IA,IX).EQ.JLOC(I,3).AND.MY(IA,IY).EQ.JLOC(I,2)) GOTO 30
            IF(MY(IA,IX).EQ.JLOC(I,2).CR.MY(IA,IX).EQ.JLOC(I,3)) GOTO 40
            IF(MY(IA,IY).EQ.JLOC(I,2).OR.MY(IA,IY).EQ.JLOC(I,3)) GOTO 50
         30 DO 31 IC=1,NE
            MZ(IA,IC)=MY(IA,IC)
         31 IF(MZ(IA,IC).EQ.JLOC(I,1)) MZ(IA,IC)=1211
            K=NE+1
            DO 32 IC=K,NCOL
         32 MZ(IA,IC)=MY(IA,IC)
            GOTO 10
         40 ICCL=1
            DO 41 IC=1,NE
            MZ(IA,ICOL)=MY(IA,IC)
            ICCL=ICOL+1
            IF(ICOL.EQ.IX+1) GOTO 42
            GOTO 41
         42 MZ(IA,ICCL)=1211
            ICOL=ICOL+1
         41 CONTINUE
            DO 43 IC=ICOL,NCOL
         43 MZ(IA,IC)=0
            MZ(IA,NCOL)=NE+1
            GOTO 10
         50 IX=IB
            GOTO 40
         11 DO 12 IB=1,NCOL
         12 MZ(IA,IB)=MY(IA,IB)
         10 CONTINUE
            RETURN
            END
```

```
FORTRAN IV G LEVEL 1, MCD 4            MXFILL          DATE = 70199         08/35/08              PAGE 0001

            SUBROUTINE MXFILL(I,IA,IROW)
            IMPLICIT INTEGER*2(I-N)
            COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
            COMMON/DBLMX/MZ(15,15),MZ1(15,15),MZ2(15,15),MZ3(15,15),MZ4(15,15)
          * ,MZ5(15,15),MX(100,15,15)
            NE=MZ(IA,NCOL)
            DO 30 IB=1,NE
            IF(MZ(IA,IB).EQ.1211) GOTO 40
         30 CONTINUE
         40 ICCL=1
            ISTOP=IB
         41 MX(I,IROW,ICOL)=MZ(IA,IB)
            ICOL=ICOL+1
            IB=IB+1
            IF(IB.GT.NE) IB=1
            IF(IB.EQ.ISTOP) GOTO 50
            GOTO 41
         50 DO 51 IB=ICOL,NCOL
         51 MX(I,IROW,IB)=MZ(IA,IB)
            IROW=IROW+1
            RETURN
            END
```

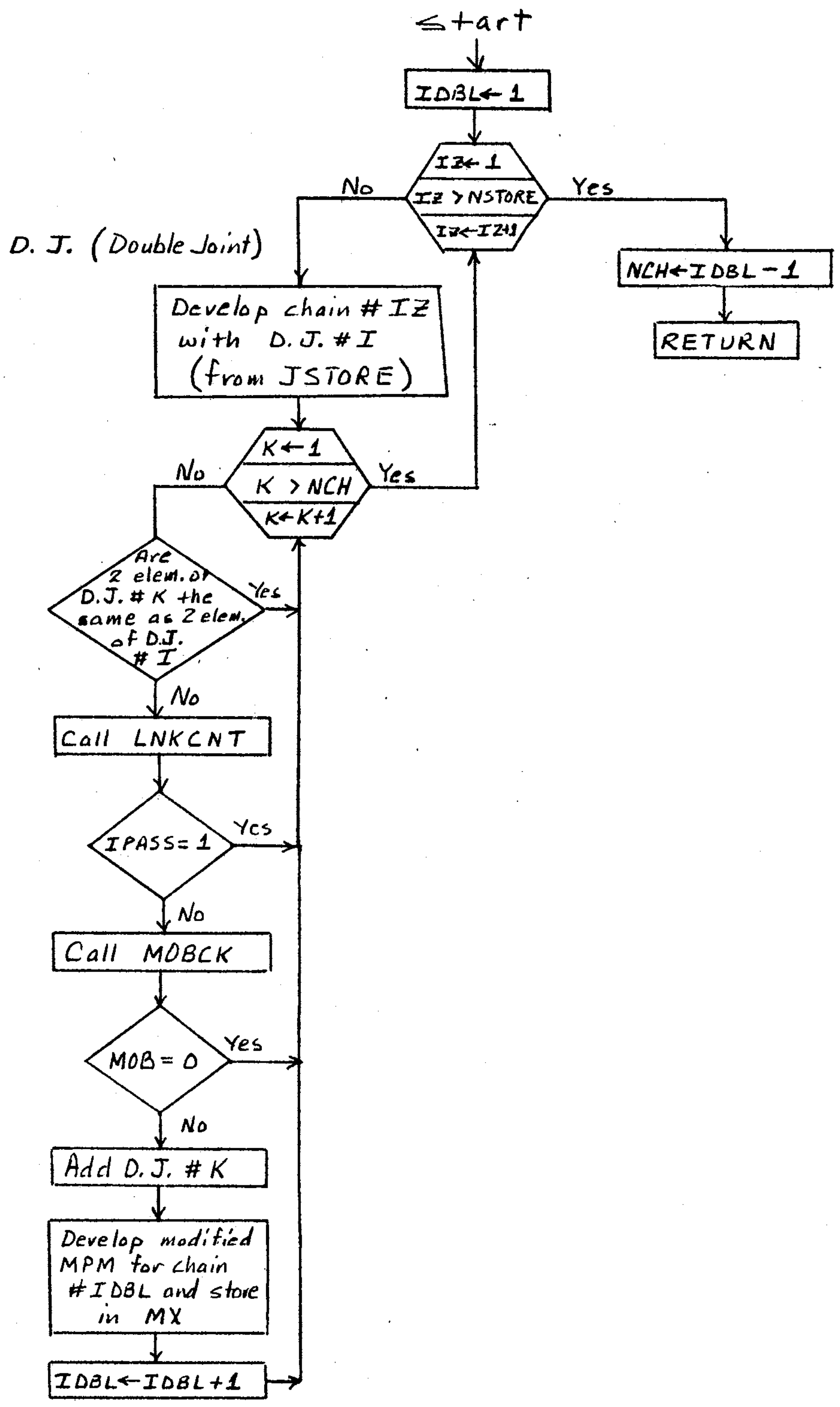


Figure 33. Flow Chart for Subroutine DBL2

TABLE LVII

SUBROUTINES LNKCNT AND MOBCK

```
FORTRAN IV G LEVEL 1, MOD 4           MOBCK            DATE = 70199          08/35/08            PAGE 0001

      SUBROUTINE MOBCK(N,M,NR,NCOL,MOB)
      IMPLICIT INTEGER*2(I-N)
      COMMON/DBL/JLOC(20,3),JDROW(20,15),JROW(20,15)
      DIMENSION M(15,15)
C CHECK ALL ROWS CONTAINING DOUBLE JOINT TO INSURE THAT NO TWO ROWS WHICH WILL
C CONTAIN ONLY FOUR LINKS, HAVE THREE LINKS IN COMMON.
      L1=JLOC(N,1)
      L2=JLOC(N,2)
      L3=JLOC(N,3)
      NRM1=NR-1
      DO 10 I=1,NRM1
      IF(JROW(N,I).NE.I) GOTO 10
C COUNT NUMBER OF LINKS EXCLUSIVE OF L1
      NE1=M(I,NCOL)
      ISUM=0
      DO 20 J=1,NE1
   20 IF(M(I,J).LT.1000.AND.M(I,J).NE.L1) ISUM=ISUM+1
      IF(ISUM.GT.4) GOTO 10
      IP1=I+1
      DO 30 J=IP1,NR
      IF(JROW(N,J).NE.J) GOTO 30
C COUNT NUMBER OF LINKS EXCLUSIVE OF L1
      NE2=M(J,NCOL)
      ISUM=0
      DO 40 K=1,NE2
   40 IF(M(J,K).LT.1000.AND.M(J,K).NE.L1) ISUM=ISUM+1
      IF(ISUM.GT.4) GOTO 30
C DETERMINE IF THE TWO LOOPS HAVE MORE THAN TWO LINKS IN COMMON
      ISUM=0
      DO 50 K=1,NE1
      IF(M(I,K).GT.1000.OR.M(I,K).EQ.L1) GOTO 50
      DO 60 L=1,NE2
   60 IF(M(I,K).EQ.M(J,L)) ISUM=ISUM+1
   50 CONTINUE
      MOB=0
      IF(ISUM.GT.2) RETURN
   30 CONTINUE
   10 CONTINUE
      MOB=1
      RETURN
      END

FORTRAN IV G LEVEL 1, MOD 4           LNKCNT           DATE = 70199          08/35/08            PAGE 0001

      SUBROUTINE LNKCNT(MZ,L,IPASS)
      IMPLICIT INTEGER*2(I-N)
      COMMON/ALL/M(15,15),NB,NL,NR,NCOL,NK,NCH,NELEM,NSTORE
      COMMON/DBL/JLOC(20,3),JDROW(20,15),JROW(20,15)
      DIMENSION MZ(15,15)
      L1=JLOC(L,1)
      L2=JLOC(L,2)
      L3=JLOC(L,3)
      IPASS=0
      DO 10 I=1,NR
      IF(MZ(I,NCOL).GT.NELEM+3) GOTO 10
      NE=MZ(I,NCOL)
      IF(NE.EQ.0) GOTO 10
      ISUM=0
      JSUM=0
      DO 5 J=1,NE
      K=MZ(I,J)
      IF(K.GT.1000) JSUM=JSUM+1
    5 IF(K.EQ.L1.OR.K.EQ.L2.OR.K.EQ.L3) ISUM=ISUM+1
      IF(ISUM.EQ.3) JSUM=JSUM+1
      IF(NE-JSUM.LE.3) IPASS=1
      IF(IPASS.EQ.1) RETURN
   10 CONTINUE
      RETURN
      END
```

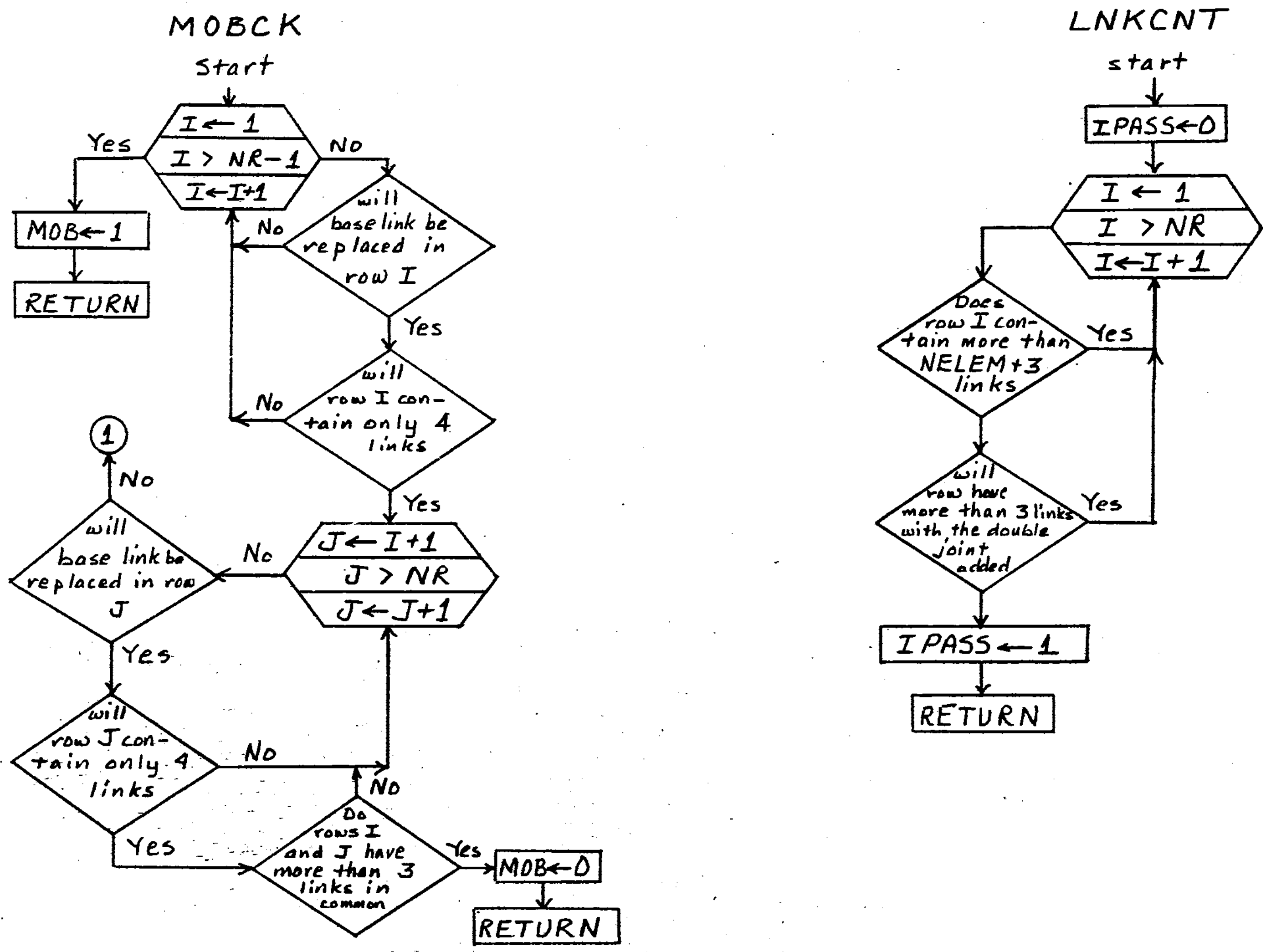

MOBCK
Start
I ← 1
I > NR−1
I←I+1
Yes
No
MOB←1
RETURN
will base link be replaced in row I
No
Yes
will row I contain only 4 links
No
Yes
J←I+1
J > NR
J←J+1
No
will base link be replaced in row J
No
1
Yes
will row J contain only 4 links
No
Yes
Do rows I and J have more than 3 links in common
No
Yes
MOB←0
RETURN
LNKCNT
start
IPASS←0
I ← 1
I > NR
I←I+1
Does row I contain more than NELEM+3 links
Yes
will row have more than 3 links with the double joint added
Yes
IPASS←1
RETURN

Figure 34. Flow Charts for Subroutines LNKCNT and MOBCK

VITA 2

Frederick Fenn Quist, Jr.

Candidate for the Degree of

Master of Science

Thesis: STRUCTURAL SYNTHESIS AND ANALYSIS OF KINEMATIC CHAINS USING PATH MATRICES

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Sault Ste. Marie, Michigan, September 29, 1942, the son of Col. and Mrs. Fred F. Quist.

Education: Graduated from Wakefield High School, Arlington, Virginia, in June, 1960; received the Bachelor of Science degree from the United States Military Academy Academy, West Point, New York, in June, 1964; completed requirements for the Master of Science degree at Oklahoma State University in May, 1971.

Professional Experience: U. S. Army Officer, 1964-1969; graduate research assistant, School of Mechanical Engineering, Oklahoma State University, 1969-1970.