[illegible]

MATHEMATICAL MODELS FOR INFORMATION PROCESSING SYSTEMS

A THESIS

Presented to

The Faculty of the Graduate Division

by

George Henry Brooks

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy in the

School of Industrial Engineering

Georgia Institute of Technology

December, 1964

MATHEMATICAL MODELS FOR INFORMATION PROCESSING SYSTEMS

Approved:

[illegible]
Chairman

[illegible]

[illegible]

Date approved by Chairman Jan. 15, 1965

ACKNOWLEDGMENTS

For one to whom so many people have been kind and helpful, it is difficult to uniquely acknowledge each person's contribution, and to do so would require as much space as the substantive portion of this thesis. At the risk of major omission, however, the author wishes to make specific mention of the following individuals:

Dr. Joseph J. Moder, the thesis advisor, for his continuing support and encouragement throughout the doctoral program. His advice and counsel during the research period were of utmost value.

Dr. Robert N. Lehrer, a member of the thesis advisory committee, for his advice, counsel, persuasion, encouragement, assistance and friendship over a period of many years. His influence on my career and my study has been profound and most beneficial.

Dr. William F. Atchison, for his participation on the thesis advisory committee.

Aside from these direct participants in this research effort, I am also deeply appreciative of the encouragement and help given by Professor Frank F. Groseclose, again for many years. Professor Earle P. Martinson, former Head of the Department of Industrial Engineering, University of Florida, now retired, has, since my undergraduate years, been a source of counsel and encouragement in my industrial and academic career. Mr. Howard Ellis, Senior Systems Consultant of the E. I. duPont de Nemours and Company, with whom it was my good fortune to associate for several years, has provided a source of inspiration for this research, and the systems experience gained under his wise and visionary tutelage has been invaluable.

The author was supported in this research by a National Science Foundation Science Faculty Fellowship. The cooperation of this agency has been of the highest type.

Perhaps the greatest debt of gratitude is that which I owe to my wife, Hope, and to my children, Marcia and Ralph, who have patiently endured the many inconveniences caused by my academic activities and the times of loneliness when the work prevented my participation in family activities. It is also they who have furnished solicitude and reinforcement when the work seemed unprofitable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

SUMMARY

The principal objective of this research is to develop mathematical models by means of which information processing systems can be described and which can be manipulated to afford a better means of analysis and synthesis of these systems. A secondary objective is to develop these models in such a manner that they are readily adaptable for computation by means of a digital computer and the associated peripheral equipment, so that this principal processor within the information processing system may be of assistance in the analysis and development of the system itself. These objectives were attained by means of the following specific developments:

1. A basic model was developed which is capable of duplicating the results of previous investigators, but with substantial improvements in ease of computation and in the amount of information made available relative to the system under study.

2. Two improved models were developed which are shown to be more adequate representations of an information processing system in that they recognize differences in types of data and the transitions which data undergo during the processing steps. These data types and data transitions were ignored in the prior work and in the basic model.

3. Algorithms were developed for conditioning the study data gathered by an analyst for computation using the two improved models.

4. Detailed machine procedures and computer programs were written for the implementation of the computational algorithms on equipment

of the sort commonly found in a typical manufacturing, commercial, or institutional situation.

The usual approach to analysis and design of information processing systems can be characterized as being essentially intuitional. Extensive use is made, in practice, of conceptual aids in the nature of flow charts and block diagrams, and much of the system description is done in prose form. The relations which exist between the system inputs and the outputs is frequently only vaguely defined.

Prior investigators Lieberman (18) and Homer (14) proposed the representation of the simple relations between systems elements in matrix form, and manipulation of these relations by multiplication of the matrices (Lieberman) or a series of column operations (Homer). In each case, the result was a solution matrix which purported to represent the compound relations of the system, and which gave a measure of the redundancy of data within the system. In the case of Lieberman's multiplication technique, data input was permitted at only the lowest level of the system hierarchy, and the multiplication was tedious and time consuming. In the case of Homer's column operation technique, data input was permitted at intermediate system levels, but a great deal of information relative to the intermediate system relations was lost in the computation, and the computation was lengthy and not amenable to computer use.

The basic model developed in the current research is essentially a matrix inversion of a triangular matrix. It is patterned after a similar model first advanced by Vazsonyi (30) and later improved by Giffler (9) in the context of materials assembly systems. In this

model the simple system relations which Lieberman displayed in a series of matrices and which Homer displayed in a single non-square matrix are displayed in a square, triangularized matrix, S. It is then shown that a solution matrix S^* is the inverse of the matrix (I - S), and that this solution matrix S^* contains the solution obtained by both Lieberman and Homer. In addition, all of the intermediate results, some of which are never obtained by the Lieberman model, and all of which are lost by the Homer model, are retained in the solution matrix of this model. In addition, because of the triangularity of the S and S^* matrices, the inversion required is easier to compute than is either of the prior models.

The basic nature of information systems is then further explored. It is established that three basic types of data exist in such a system. These three types are distinguished as identification data, such as department numbers; quantitative data which arise basically from a count or measurement; and existence data which primarily indicate either that some activity has taken place or the status of some activity.

It is further established that the essential purpose of an information processing system is to change the form of the input data by changing it from quantitative to existence type data, by summarization, or by computation of some nature. Nine categories of such data transitions are recognized and discussed.

It is developed that a model for an information processing system must recognize these three data types and the transitions which the data undergo. In order to implement this recognition, three basic types of system relations are defined, instead of the single type of relation implied in the prior work and in the basic model.

These three systems relations are denoted as prime relations, concomitant relations, and deletion relations. A prime relation is that relation which exists when an element of a system is required for the preparation of a higher level element in a system. A concomitant relation arises in conjunction with the transition of data from quantitative to existence type, or in conjunction with a computation. A deletion relation exists when a data element is brought to some level of a system because it is a part of a record which contains other usable data, but where the data element itself is not required at that level.

Based on the recognition of the three types of system relation, two improved models are developed. In one of these, only prime and concomitant relations are used, while the other uses all three types of relation. The first results in a solution matrix wherein all of the lower level data elements which are required for the preparation of a higher level report are indicated, whether or not the lower level data actually appear on the higher level report. This matrix, S^*, is referred to as comprising a "composed of" type of analysis. The second model results in a solution matrix S'^*, which shows for a given higher level report only those lower level elements which actually appear on the report. This is referred to as a "contained in" type of analysis.

While the prime relation uses regular numerals as quantifiers in the S matrix representing the simple system relations, it was necessary to define special quantifiers for both concomitant and deletion relations. The concomitant quantifier gives rise to a special multiplication-like operator, while the deletion relation gives rise to a logical, rather than arithmetic, operator. The combination of prime and concomitant quantifiers in the matrices representing the information system complicated the inverse relationship shown for the basic model, and it was necessary to prove a theorem relative to the associativity of multiplication of matrices containing both prime and concomitant quantifiers. It was then possible to show that the S^* matrix of the first refined model was indeed the inverse of the $(I - S)$ matrix, as in the basic model.

The deletion relation quantifier is a logical operator, and it is shown that the inverse relationship no longer holds for the solution matrix S'^* and the matrix $(I - S')$ of the second refined model. It is shown, however, that they differ only by two logical operations occurring in the computation as a result of the deletion function.

Since both models depend on the triangular nature of the matrices for their computational base, a method of triangularization of the indices is developed. This technique permits the analyst to use any arbitrary designation for the systems elements, and does not require him to be concerned with system levels or the required triangularity.

In practice, the models presented would require the use of a digital computer and associated equipment for computation, so the algorithms have been programmed for a typical computer. In addition, a method has been developed and programmed to establish the amount of computer memory required for a given specific program and system.

CHAPTER I

INTRODUCTION

Purpose

The purpose of this research is to attempt to bring to the field of business information processing the concise notation, non-ambiguous description, and computational rigor of mathematics and mathematical models. It is hoped that the techniques of systems analysis and design developed during the course of this research will supplement, at least to an extent, the intuitional, non-quantitative approaches now used for information systems analysis and synthesis, and will result in economies both in systems development and in routine systems operation.

Background of the General Problem

The processing of information is a problem which is not new, but one which has become much more important within the past few years. The development of ancient languages, number systems, and rudimentary mathematics can be traced to the need of the ancients to compute and record taxes. The Bible abounds in references to taxation and census-taking, both of which imply an accompanying task of information processing. The making of decisions implies prior data processing of at least an elementary sort, as decisions are not possible without knowledge, however incomplete, of the possible outcomes of the alternative courses of action.

In modern context, information processing has come to imply mechanical, electro-mechanical and electronic processing of data. This concept has its beginning in the mid-1880's, when Dr. Herman Hollerith developed a mechanical method of processing census data, using the "unit card" principle now so familiar. The first set of equipment was installed in the Department of Health in Baltimore, Maryland for the processing of public health data in 1889. The Bureau of Census had equipment in time to process the data of the 1890 census. Dr. Hollerith later founded the Tabulating Machine Company, from which later emerged the International Business Machines Corporation (26, Schmidt, R. N., and Meyers, W. E.).

During the next several years, until the early 1950's, most of the development in the field was in refining and speeding up the equipment developed earlier. The basic concept remained unchanged: information relative to a single transaction was transcribed to a punched unit record card, and this card was then used for the updating of records and the preparation of reports. All processing was done in "batches," due to the limitations of flexibility of the equipment.

In the scientific community, strong interest was manifested in this equipment as a means for performing scientific calculations. However, the limitations of the equipment were such that many persons, starting in the 1920's, sought other, more sophisticated, equipment for performing scientific computation. Out of these early efforts emerged, in the late 1930's, some crude computers. Under the impetus of World War II, these crude computers were greatly refined, and then played a significant role in the development of the atomic bomb and other military developments.

One of the computers developed during the war was the Eniac, the development of Dr. John W. Mauchly and J. Presper Eckert at the Moore School of Engineering at the University of Pennsylvania. This computer was the world's first all-electronic, high-speed, large scale computer. Eckert and Mauchly later developed the Edvac, which was the first stored program computer, and which was the forerunner of Univac, the first commercial electronic computer. The first Univac I was delivered to the Census Bureau in 1951.

The introduction of the Univac I was followed quite rapidly by the entry into the field of several computers from many manufacturers, notably IBM. It is significant to note that virtually all of the computers introduced in the mid-1950's, including the popular IBM 650, were designed essentially for scientific computation, and were used for business information processing only by virtue of clever programming and sacrifice of efficiency of internal operation. Despite this handicap, the advent of stored program digital computers helped to cause what has come to be recognized as a revolution not only in the processing of information, but in the basic philosophy of operating a business enterprise as well.

Other factors beyond the development of the digital computer contributed to this revolution. One of the most important but perhaps the least recognized is the work done in the early 1950's by the United States Steel Company (1, American Management Association). This work was undertaken to establish the feasibility of a "Common Language" with a view toward extensive automation of clerical operations using the then existing punch card equipment. Common language was defined as (2, Moore Business Forms, Inc.):

> A recording medium that (a) comprehends the 26 letters of the alphabet, the 10 decimal digits, and a minimum of special characters and functions, and (b) has the ability to mechanize the transfer of raw data directly between the office and communications machines manufactured by different suppliers.

This group did not entirely succeed in their goal of establishing a single common language, but rather established two: punched cards and punched paper tape. To these have been added more recently magnetic tape and direct processor to processor communication. However, the group did accomplish two other things which have had a profound effect on the development of information processing. The first of these was that they did succeed in interesting manufacturers of all sorts of data processing equipment--adding machines, bookkeeping machines, typewriters, calculating machines, Addressograph machines, etc.--in adding either tape or card input and output devices to the equipment to permit the creation of data in mechanized form as a by-product of the necessary operation, and to permit the communication of data between equipment without human intervention. Second, the group established the basic principles of Integrated Data Processing.

The importance of the concepts of Integrated Data Processing to the revolution in information processing cannot be overstated (5, Ellis, H.).

> When I. D. P. was first introduced, many companies and governmental agencies began to exploit successfully this systems concept, in separate segments of their businesses. While this was going on, some of the same or different groups within companies, were doing considerable work, in getting on-stream their new high speed digital computers.

Many of these early starters realized that the integrated systems approach would be the most effective method to use, to get the most out of their computers--some did not. But most seemed to have the same goal in mind, and that was--that much mechanized data processing should be employed in order to get more action and value out of the results of data origination, transmission, processing, and data usage. In other words, improve the aids to decision making, by employing the least data manipulation, and retrieving the most value from existing and dynamic information.

There was, however, some unjustified optimism among those companies that did not use the integrated approach as a part of their data processing systems. Their systems were too machine oriented and not sufficiently systems concept oriented. What happened seemed to be this: That, as various mechanized systems (computerized or not) began to take hold, independently, in several segments of a business, some segments found their system to become stymied sooner or later and other segments discovered expensive problems to be overcome in data manipulation; also to find later, that their companies' combined systems were not sufficiently integrated and compatible, and would run somewhat short on expected accomplishment.

Another important contributing factor to the revolution in information processing has been, of course, the emergence of a discipline called variously systems engineering, operations research, applied mathematics and statistics, management science, etc., depending on the point of view of the individual describing the discipline. Whatever the title applied, this discipline has as its principle features the use of the scientific method and mathematical techniques for the purpose of optimizing the overall operation of the concern. The reason that this factor has been so important to information processing is that optimization in an enterprise concerns itself directly with the decision processes of the enterprise. In turn, the decision processes are entirely dependent on information processing, so that attention has been drawn automatically to this all-important facet of the enterprise. It might be said that information is the common thread running throughout the fabric of an organization.

Since the development of information processing technology has been so rapid in the past few years as to be termed "revolutionary," it is not surprising that one fails to find a unified body of either theory or practice in the field. Much of the development both in individual companies and at large has been accomplished by "trial-and-error" and "brute-strength-and-awkwardness" approaches. Many computer installations have been made without adequate prior justification. Individual companies have used specific equipment far beyond the design expectations of the manufacturer. Since the first computers were designed as scientific computers, all of the early installations for information processing were basically make-shift in nature, and only the great speed differential between computers and electro-mechanical equipment and the ingenuity of the programmers made them relatively successful.

In discussion of the present state of the art of information processing, it is helpful to distinguish between "exterior systems" and "interior systems." The exterior system embraces the entire operation under consideration--the set of decisions to be made routinely to control the operation, the set of input data to be processed in order that an optimal decision may ultimately be made, the required data origination, data transmission, data processing, report structure, and provision for data storage and subsequent retrieval, and the set of decision functions which will be used in conjunction with the processed data in order to arrive at specified decisions.

The interior system refers to the details of machine programming. It embraces such factors as card and record format, report format, operating rules, and most importantly, equipment programming itself. It concerns itself primarily with the data processing portion of the exterior system, but is also partially involved with data origination, data transmission, and data storage and retrieval insofar as the details of programming are involved.

Because of the complexity of the task of machine programming, the interior systems have had more attention in the literature and in practice. There are many books devoted to this aspect, as well as several periodicals, all of which are "hardware" or interior systems oriented. Relatively little has been written with regard to the design of the total, or exterior, system.

The general approach to exterior systems design at the present state of development is an intuitional one. Much of the information necessary for the design of the system is obtained by means of interviews with a large number of the working personnel of the organization, only a few of whom have any real qualification for giving pertinent answers. Too often, the systems engineer is forced to settle for a statement of what information management _wants_, rather than what management _needs_.

Almost the sole device for systems analysis and description is the graphical flow chart, accompanied by written or tabulated descriptive information. Many of these charting techniques have been dignified by a copyrighted name (19, National Cash Register Company)(6, Evans, O. Y.), but they are essentially similar and provide at best a systematic approach. There is no provision for quantifying data, nor for manipulation with a view to optimization.

In interior systems design, the flow chart is also used extensively, probably more so than for exterior systems. The complexity of computer programming has also forced the development, by both equipment manufacturers and users, of compilers--computer routines which will accept pseudo-English, problem-oriented instructions and convert them into machine language programs. Technical compilers were developed first, and have been available for some years; commercial or business compilers, due to greater differences in terminology, have only been made available quite recently.

The existence of compilers does not alter the basic fact, however, that business problems are largely uniquely programmed for each specific problem and each specific installation. The common payroll, which has been highly refined over the years, is seldom programmed identically, even in plants of the same division of a company. Similarly, "canned" or library programs, such as are widely available for technical problems, are not widely used in the information processing field, for the primary reason that the generality which exists between problems and installations has either not been perceived or has not been characterized. A few library programs are available; IBM for example offers MOS (Management Operating Systems) and PAL (Programmed Applications Library) programs to users of certain of their equipment, but these programs usually need extensive modification for use in a given installation.

Importance of the Problem

This lack of adequate means of systems analysis and design for both interior and exterior systems is a serious one. The penalty for this lack is essentially an economic one, but one that is difficult or impossible to assess. These economic penalties arise in the following basic ways:

1. The inability to rigorously characterize the exterior system results in a failure to optimize the total system design. At best, the design is sub-optimized by parts. At worst only a feasible system is designed, with little real thought given to optimality. To the extent that the system is less than optimally designed, the enterprise pays a corresponding economic penalty in terms of less than optimal return on investment.

2. Systems development using these inadequate techniques is costly. The interview technique is time consuming, both of systems engineering and operating personnel's time. The intuitional approach results in wasteful meetings, discussions, and inefficient use of facilities and manpower. Graphic representations of systems are tedious, time-consuming, and costly to prepare. Frequently several man-years of effort are devoted to charting the existing system before any creative work even begins.

3. In interior systems design, the two principal penalties are the cost of programming personnel, and the cost of operating either a larger computer than is needed or of using a smaller computer for more shifts than are really necessary. The philosophy of the development of compilers was to trade some of the speed of the computer for some of

the effort of programming. That is, it is much quicker and easier to program with the use of a compiler, so that fewer and perhaps less highly trained programmers can be used. However, machine time is used to compile the program, and, more importantly, the resulting program is not as efficient in the use of machine time as a carefully written program in machine language.

It should be noted that this latter factor is much less important in scientific computation than in information processing. In scientific programming, programs are usually used at most several times until the problem leading to the program is solved, and may never be used again by a given installation. The whole emphasis is in permitting the scientist or engineer to communicate with the computer in a language as near like the language normally used by the scientist as possible, and with minimum knowledge of computers and computer technology on the part of the scientist.

In information processing, the programs are used repetitively, sometimes several times a day, for at least several months and frequently several years. Any inefficiency in the program can raise the computation time, and therefore the cost, by a large amount through this frequent use. In the early days of programming when computers were severely limited in speed and capacity, a great deal of work was done in "optimizing" of programs for efficient running. With the advent of large memories and faster internal computing speeds, the immediate necessity of optimization of programs was lessened, and compilers became possible and popular. However, the fact that the penalty is less apparent has not diminished its cost; it has only permitted a trade-off for programming time. The penalty is a very real one, and is one which is too frequently overlooked.

In view of these economic penalties, it seems apparent that there is a real need for development of more rigorous and efficient means of systems analysis and design, for both internal and external systems.

The Specific Problem

The specific problem which will be attacked in this research relates to the methods used for description and analysis of existing or proposed information systems. As previously noted, the methods now in prevalent use are essentially qualitative. Description is usually done in either prose or graphic form, and any manipulation of the elements of the system is done on an essentially intuitive basis.

The models which will be presented will permit description on a quantitative basis, and will eliminate the necessity for prose description and the drawing of flow diagrams in most cases. Furthermore, the models will furnish information to the analyst relative to the areas within a given system which require study, and will furnish a basis for system design improvements by simulating the effect of any proposed changes.

CHAPTER II

LITERATURE SURVEY

Introduction

Despite the popularity of the subject of computers and data processing, the literature relative to quantitative or even systematic approaches to systems analysis and synthesis is remarkably sparse. Certainly much has been written in the broad field of information processing, but the vast majority of the material is either directed toward specific hardware or is narrative of a specific installed system for a specific enterprise.

Compilers

The greatest quantity of material to be found concerns itself with the subject of compilers. Under the stimulus of the U. S. Department of Defense, a group composed of representatives of the several computer manufacturers and governmental and industrial users was formed in 1959 for the purpose of producing "an English-like programming language which can be used with many different types of data processing systems" (8, International Business Machines Corp.). This group is known as the CODASYL group, where CODASYL is the acronym for Conference On DAta SYstems Languages. The principal output of this group to date is a language called COBOL (COmmon Business Oriented Language). The general requirement now is that any computer furnished for government use be accompanied by a COBOL compiler which will accept a certain required set of pseudo-English statements by means of which business computation programs can be written and presumably run on the computer of any manufacturer. In addition to the required set of language, the manufacturer is free to implement further elective language to exploit the unique features of his equipment.

On the surface, COBOL appears to be a simple language. For example, very complex computations can be written in English-like sentences such as the following:

```
SUBTRACT DEDUCTIONS FROM GROSS GIVING NET.

PERFORM TAX-CALCULATION.

IF STOCK IS LESS THAN ORDER-POINT PERFORM REORDER-ROUTINE.

WRITE MONTHLY-STATEMENT.
```

This simplicity, together with the promise that a program written for one machine can be used on another machine, even of another manufacture, would seem to be very desirable. However, COBOL is not as simple as it appears to be, and the literature indicates that COBOL leaves much to be desired. Indeed, some authorities have serious reservations about the entire compiler approach.

In a particularly critical article Grosch (12) states:

> One major agreement, for instance, is on the original concept of commonality: American and British, commercial and scientific, manufacturer and user, large machine owner and small, all agree that magic languages do not--repeat, <u>not</u>--make it possible to transfer work from one machine to another.
>
> On the question of economics many voices are raised: Patrick in the opening sentences at Rand; Gruenberger and Gordon later; Paine and Glennie in the Northampton College (BCS) discussions. All point to the hidden costs of the magic language: long compile times, repeated re-compilation during debugging, longer running times, and--surprise, surprise!--higher prices on the machines to cover the manufacturers' software investment.

Shaw (27) tends to agree with the foregoing comments pointing out the deficiencies of programming languages:

> Another reason for the existence of an overwhelming variety of programming languages is the fact that most of these languages are not machine independent. Even the ones that claim to be are not really. Thus, the motivation toward a standard notation for those areas that could be machine independent is considerably weakened by the argument, "If the language is going to be different anyway, why not make it lots different?"

Shaw also claims that the machine-independent languages are in fact harmful:

> This proliferation of "machine-independent" programming languages has two well-known and pernicious effects: it inhibits the communication of information processing procedures between people and computers both, and what's worse, it distracts attention and effort from that which is to be communicated--namely, procedures for solving real, worthwhile problems. Interestingly enough, these two effects are exactly those which each individual language is supposed to ameliorate. In conjunction, therefore, these languages tend to defeat their own purposes, with the result that the computing industry is, today, suffering from too much of a good thing.

In another article discussing both scientific and commercial compilers, Orchard-Hays (22) summarizes the feelings of several of the writers when he states:

> However, many feel COBOL is overly elaborate and does not deal with the real problems of data processing.

He also makes the point that COBOL, in spite of its supposed generality, has over 30 versions now published. In another analysis of COBOL (and ALGOL), Cantrell (4) indicates many deficiencies:

> The current COBOL and ALGOL families of languages are very elaborate and sophisticated languages. They are very general . . . This generality is accompanied by a profusion of rules and regulations. . . . As a result these languages are considerably harder to learn to use effectively than more simple languages.

. . . COBOL Procedures statements are relatively easy to read but the reader has to be well educated to understand the Data Description statements.

The English language design of COBOL makes it a verbose language. . . . The programmer has been relieved of machine details only to be faced with the problems of spelling words correctly and learning the exact meanings of some 256 "key words" in a new vocabulary. Often each word has several different meanings depending on its context. . . . This is not English but "pseudo-English" with its own specific grammar.

. . . But our newer languages, such as COBOL, have not added features to do things which we could not do in our older languages. Instead they have added more and fancier ways of doing things which we could do fairly well anyway. We do not appear to be developing better languages, only fancier ones. . . .

COBOL and ALGOL and their derivative languages appear to have been designed on the basis that anything can be implemented and that the consideration of compiler requirements in language design would lead to somewhat more machine oriented languages. We might say that this language design method is really no concern of the user. He doesn't have to use all of those fancy features if he doesn't want to. But when slow compilers, producing inefficient object programs come out a year or two late, the user begins to suspect that he would have been better off without all these frills. The user could never afford to buy hardware that was designed without any consideration of the cost of manufacturing it. It appears that he can hardly afford to wait for, and then waste computer time with, languages which were not designed for the best compromise between language utility and implementation requirements.

. . . Many compilers produce quite efficient object programs. Others are not so good. Horror stories of 10% to 20% efficient object programs are passed from user to user. Unfortunately many of these stories are true.

Despite criticisms of COBOL, it has become the established language for business data processing, albeit by Government fiat. The original CODASYL group, now expanded, is active, and with others is seeking for improved means of programming. One very promising adjunct to COBOL is the use of decision tables. Originally proposed by Kavanagh (15, 16) and Grad (10), under the acronym of TABSOL, the concept is currently under appraisal by the CODASYL group, where the technique is called DETAB-X (23, Pollack, S. L.).

Decision Tables

Decision tables are a systematic method for describing and documenting the flow of logic in a program. Like COBOL, certain descriptive material is written in pseudo-English, and in fact, in some versions COBOL statements per se are used. The general arrangement of the decision table format is as shown in Table 1.

Table 1. Decision Table Format

TABLE HEADER	
STUB HEADER	RULE HEADER ENTRY HEADER
CONDITION STUB	CONDITION ENTRY
ACTION STUB	ACTION ENTRY

The essence of the technique can perhaps best be gained from a small example adapted from a published procedures manual, and shown in Table 2 (6, op. cit.).

Table 2. Example of a Decision Table

TABLE: CREDIT CHECK								
					RULE 1	RULE 2	RULE 3	RULE 4
LINE NR.	VERB	OPERAND	OP	OPERAND	OPERAND	OPERAND	OPERAND	OPERAND
1		SPECIAL CLEAR	EQ		1	0	0	0
2		CREDIT LIMIT	GE	OK*		Y	N	N
3		PAY EXPER- IENCE	EQ				GOOD	BAD
4	MOVE	'APPRO- VED'	TO	ORDER STATUS	X	X	X	
5	MOVE	'REJECT'	TO	ORDER STATUS				X

OK* = In process amount + Accounts Receivable amount + Order amount.

In this decision table, the rules for performing a credit check are specified:

Rule 1 states: If special clearance is EQ (equal) to 1, then move the word 'approved' to order status.

Rule 2 states: If special clearance is equal to 0 and the credit limit is GE (greater than or equal to) OK then move the word 'approved' to order status.

Rule 3 states: If special clearance is equal to 0 and the credit limit is not greater than or not equal to OK and pay experience is equal to GOOD, then move the word 'approved' to order status.

Rule 4 states: (in conjunction with rule 5) If special clearance is equal to 0 and the credit limit is not greater than or not equal to OK, and pay experience is BAD, then move the word 'reject' to order status.

The advantages of the use of decision tables (with the current connotation that they will be used in conjunction with COBOL), are cited as (10, op. cit.):

Conciseness and clarity
Completeness
Meaningful relationships

Kavanagh is somewhat more enthusiastic (16, op. cit.):

> Structure tables force a logical, step-by-step analysis of the decision. . . . are easily understood by human beings regardless of their functional background. . . . form an excellent basis for communication between functional specialists and systems engineers. . . . go a long way toward solving the difficult systems documentation problem. . . . format is so simple. . . . that engineers, planners, and other functional specialists can write structure tables for their own decision-making problems with very little training and practically no knowledge of computers or programming. . . . errors are reported at the source language level, thus permitting the functional specialist to debug without a knowledge of computer coding. . . . offer levels of accuracy unequalled in manual systems. . . . are easy to maintain.

The literature does not disclose extensive application of the decision table technique, so that it is difficult to evaluate the practical merit of the method. It does appear that the concept is helpful to interior systems design; it also appears that some of the same criticisms which are attached to COBOL may also be applicable to TABSOL and DETAB-X. In particular, it seems evident that many rules and stipulations will accompany this language. The preliminary manual cited requires some 80 pages and is acknowledged to be incomplete. It is also felt that the same lack of general compatibility will become evident as these languages are implemented for the machines of various manufacturers. However, there is at least some semblance of a rigorous approach evident in this technique.

Graphic and Tabular Methods

Another area of occasional interest in recent writings has been that of graphic or tabular methods for either interior or exterior systems description. Grad and Canning (11) have proposed a graphical technique, while Evans (7) has proposed a tabular method. In addition, almost every computer manufacturer and almost every text relative to information processing proposes a graphic or tabular approach to systems design, under the generic title of "flow charting." None of the several graphical techniques investigated appeared to have any unique features; they will not, therefore, be discussed here.

Generalization Methods

Some attempts at generalization of interior systems design have been made, and have been relatively fruitful. The concept here, as outlined by McGee (20), and later by McGee and Tellier (21), is to write very general programs for operations such as sorting, report generation, and file maintenance, and to use these programs rather than writing specific programs for single-purpose operations. The concept has been quite widely accepted, and nearly every manufacturer of equipment has available generalized sorts and report generator programs. It should be noted that the principle advantage of these routines is the saving of programming time and effort, at the expense of running time on the computer. It should also be noted that McGee specifically does not attempt to generalize the calculations involved in an information processing system.

Quantitative Approaches

The body of literature relative to what might be called quantitative approaches to either interior or exterior systems design is woefully small, and much of the material which does appear addresses itself to particular aspects of systems design rather than to the general problem. For example, Wagner (31) writes of the use of decision theory to determine the amount of specific data necessary for making a given decision, while Rowe (25) discusses the use of simulation and display matrices in analysis of management control systems.

Two papers, by Lieberman (18) and Homer (14) are worthy of extensive note. Since they furnish a foundation for the present research, a separate section will be devoted to these two papers later in this chapter. Basically, these two papers used matrices for the description of systems and the manipulation of systems elements. With minor expansion, Kozmetsky and Kirschner (17) printed the Lieberman model as an appendix to their book. The principle expansion was in terms of a more lucid explanation of the use of the model, and more flexibility in its application.

A similar matrix approach, but a much more cumbersome one, was taken by Henderson (13). He assayed the construction of a five-dimensional model, his dimensions being "Functional distinction, natural structure, managerial prerogative, substantive content, and time." Even though he speaks of five dimensions, his matrices are two-dimensional arrays. One matrix is used for each report, so that a given system requires a large number of matrices. He further requires that a value be attached to each piece of data, so that an objective function can be written.

Henderson is able to cast his model in the form of an integer linear programming problem. The conclusion is reached, however, that the solution of the linear programming model cannot be found because of the very large number of constraints. It is interesting to note that 69 equations are used to merely describe the kinds of constraints.

In another approach, Young and Kent (32) propose the use of essentially a symbolic logic method for exterior systems design, with some connotation of interior systems design as well. Information sets, document sets, and set relationships are defined using established set and symbolic notation. While this approach is excellent from a descriptive point of view, great difficulties are encountered in manipulation of such a model. In the article, the authors resorted to a clever but rather complicated graphical technique which, while useful, does not offer the advantages of the Lieberman-Homer models.

In a graduate project at Purdue (29, Thomas, W. H.), a similar conclusion was reached; that symbolic logic is quite effective for describing systems, but that algorithms for manipulation of the model so obtained are not presently available, nor does it appear that they will become available for some time. Such a development must await a great deal of research in pure mathematics.

The Lieberman Technique

Lieberman's notation is as follows (18, op. cit.):

i = identification data; e.g., employee number.

q = quantitative data; e.g., hours worked.

Both i and q are subscripted by the same series (say k) for indexing of specific data elements.

B_r = the rth business function; e.g., production control.

S_m = the mth source data form; e.g., time card.

R_n = the nth report form; e.g., stock status report.

M = general notation for a matrix. A subscript s indicates a source data matrix, a numerical subscript indicates the level of an intermediate matrix, a subscript B indicates a matrix of the requirements of the business functions for given reports R.

The matrices may be called incidence matrices, since the rule for an entry in a given matrix is that a cell value of 1 is assigned if a given report requires a given piece of data, and is zero otherwise. The formation of the matrices is best seen by a simple example.

The first matrix, M_s, has a row for each i_k and q_k, and a column for each S_m. Table 3 shows an M_s matrix which presumes a system with six data pieces and four source data forms.

The interpretation given to this matrix is, for example, that identification data item 1 appears on source forms 1 and 4, while quantitative data item 5 appears on source forms 3 and 4.

Table 3. The M_s Matrix

	S_1	S_2	S_3	S_4
i_1	1	0	0	1
i_2	1	1	1	0
i_3	0	1	0	1
q_4	1	1	0	1
q_5	0	0	1	1
q_6	1	1	1	1

The first level report structure can now be shown in matrix form, as in Table 4, which assumes five first level reports, and uses a superscript on R to indicate the report level.

Table 4. The M_1 Matrix

	R^1_1	R^1_2	R^1_3	R^1_4	R^1_5
S_1	1	1	1	1	1
S_2	0	1	0	0	1
S_3	0	0	1	1	1
S_4	0	1	0	1	1

A similar interpretation is placed on the entries in this matrix. For example, source form number 2 is required for reports 2 and 5. Similarly, a matrix for second level reports can be constructed, as shown in

Table 5, where three second level reports are assumed.

Table 5. The M_2 Matrix

$$\begin{array}{c|ccc} & R_1^2 & R_2^2 & R_3^2 \\ \hline R_1^1 & 1 & 0 & 1 \\ R_2^1 & 1 & 1 & 1 \\ R_3^1 & 0 & 0 & 1 \\ R_4^1 & 0 & 0 & 1 \\ R_5^1 & 1 & 0 & 0 \end{array}$$

Finally, assuming a two level report structure, the M_B matrix can be constructed, showing the use of second level reports by the several business functions. In Table 6, two business functions are assumed.

Table 6. The M_B Matrix

$$\begin{array}{c|cc} & B_1 & B_2 \\ \hline R_1^2 & 1 & 1 \\ R_2^2 & 0 & 1 \\ R_3^2 & 1 & 0 \end{array}$$

At this point, certain information can be gained by examination of the matrices. For example, it is seen that S_4 contains all but one piece of the basic data in the system; one might investigate the feasibility of adding one piece of data to S_4 and eliminating the other source forms. It is also possible by adding rows and columns of the matrices to gain a measure of the use of a given piece of data or a given report.

Table 7. The $(M_s)(M_1)$ Matrix

	R_1^1	R_2^1	R_3^1	R_4^1	R_5^1
i_1	1	2	1	2	2
i_2	1	2	2	2	3
i_3	0	2	0	1	2
q_4	1	3	1	2	3
q_5	0	1	1	2	2
q_6	1	3	2	3	4

It is now noted that if the four matrices are taken in the sequence M_s, M_1, M_2, M_B, that they are comformable for matrix multiplication. Performing the first multiplication, the $(M_s)(M_1)$ matrix shown in Table 7 is obtained. This matrix shows the number of ways that each piece of source information is available to each of the first level reports. Thus q_6 is available to R_5^1 in four different ways, while i_3

is not available to R_1^1 or to R_3^1. This first product matrix can now be multiplied by M_2 to show the availability of source data to the second level reports. In turn, the product matrix so obtained can be multiplied by the M_B matrix, which will show the availability of source data to the business functions. Performing these two further multiplications results in the final, or solution matrix, for the entire system shown in Table 8.

Table 8. The Solution Matrix

	B_1	B_2
i_1	11	7
i_2	13	8
i_3	7	6
q_4	14	10
q_5	7	4
q_6	17	11

This resultant matrix presumes to show the redundancy of the system, and examination of the original and intermediate matrices is proposed to disclose ways in which the redundancy can be reduced. The interpretation of the solution matrix is that a given cell shows the number of ways which a given piece of source data is being made available to a given business function.

As presently constituted, Lieberman's model is best adapted to analysis of existing systems, or design of new systems where it is suitable to use the old system as a point of departure. Lieberman alludes to the fact that the model might be used for synthesis of new systems, by essentially constructing a final matrix of data sources to business functions, then using an 'inversion' function to work backward to the M_s matrix. He does not contemplate the usual matrix inversion, however, and discussion of this phase is deferred to a "later" paper which has not appeared.

The Homer Technique

The basic contribution of Lieberman was to furnish a concise basis for the description of exterior systems, and to gain some measure of the data redundancy of a system. Homer (14, op. cit.) pointed out some weaknesses of the Lieberman model and proposed techniques for overcoming these weaknesses and for improving the computational procedures.

Since the first phase of the current research will reproduce Homer's results with a considerable simplification of procedure and with more resultant information, it is appropriate, for purposes of comparison, to include Homer's notation, argument, and technique at this point. His mathematical development will not be included, since it justifies only his technique, and is not pertinent to the method to be developed in this current research.

Notation

Homer's notation is as follows (14, p. 501):

d_i = the ith item of data in a system: $i = 1, 2, \ldots, m$.

$R_{k(j)}$ = the kth report (or document) at the jth level; $j = 1, 2, \ldots, n$; $k = 1, 2, \ldots, p_j$. In particular, let:

$R_{k(1)}$ = the kth source document.

B_r = the rth business function; $r = 1, 2, \ldots, q$

M_{j-1} = the matrix which depicts the components required for the preparation of jth level reports. In particular, let:

M_0 = the matrix which depicts which items of data appear on which source documents.

M_n = the matrix which depicts which nth level reports are used in performing the various business functions. (This implies that business functions are carried out at the $n = 1$ level.)

Limitations of Lieberman Model

Homer cited the following conditions which "hinder the establishment" of the Lieberman model (14, p. 505):

1. Cases where it is extremely difficult to define the level of a report, because it is prepared not only from jth level reports, but also reports of level $j - 1$, $j - 2$, etc.

2. Similarly, cases where business functions are performed not only on the basis of nth level reports, but on lower level reports as well.

3. Cases where some business functions are performed at the $n + 1$ level, others at lower levels.

4. Cases where items of data enter a system at some level higher than $j = 1$.

5. Cases where reports are prepared outside the scope of the study, (outside the enterprise, often) and enter the system at some relatively high level.

6. Cases in which some $R_{k(j)}$ is a terminal report for some $j < n$; such as records primarily historical in nature, or summary reports prepared for use outside the scope of the study.

Such conditions result in a series of matrices which are incompatible for multiplication, for the row headings of M_j are not necessarily identical to the column headings of M_{j-1}.

Homer then develops an example, first using Lieberman's technique to demonstrate that the resultant matrices are indeed non-conformable (incompatible) for multiplication. He then develops two methods of resolving this difficulty.

The first of these techniques is that of augmenting the individual matrices (of Lieberman) with dummy vectors in order to force conformability for matrix multiplication. Then the series of matrix multiplications is performed exactly as contemplated by Lieberman in order to obtain the solution matrix. Since this method is rather lengthy, and is not as good a method as Homer's second technique, it will not be further described.

Homer's Second Method

In Homer's second method, a single matrix S is formed, using the elements of the M_{j-1} and M_n matrices. Then a solution, S^*, is obtained by performing a series of elementary column operations on S.

The rules for formation of the S matrix are (14, p. 508):

1. A row is established for each d_i and for each R_k in the system; i. e., for each component except the B_r's.

2. A column is established for each R_k and for each B_r in the system; i. e., for each component of the system except the d_j's.

3. As before, the number 1 will be inserted in each cell to represent an item of information appearing in a report, one report used to produce another, or a report used to perform a business function. In other words, those cells which would contain a 1 under the method of . . . (First method) . . . would contain a 1 under this method.

4. Each R_k will be represented by both a column and a row. Into each cell formed by the intersection of an identical row and column (r_{kk}), the value -1 will be inserted.

5. All other cells will be labelled zero.

6. The matrix can now be analyzed as follows:

   a. Should any column contain only zeros, the report or business function represented is outside the scope of the problem being investigated, and the column should be removed.

   b. Should any row contain only zeros, the report or the item of information represented is not a component of the system, and the row should be removed.

   c. Should any column contain -1 as the only non-zero entry, the component represented by that column is really an input to the system. The column should be removed.

   d. Should any row contain -1 as the only non-zero entry, the component represented by that row is really an output of the system. The row should be removed.

   e. In the event that the application of the above rules results in the deletion of both the row and column representing the same component, this is an indication that that component is not a member of the system under investigation.

The S matrix for Homer's example is shown as Figure 1. Homer indicates that the system inputs are data elements $d_1, d_2, \ldots,$ d_7, and reports $R_{4(1)}$ and $R_{6(2)}$, while the outputs are the report $R_{4(2)}$, and business functions $B_1, \ldots, B_4$.

An algorithm is then developed by means of which the solution matrix S^* in Figure 2 is developed (14, p. 509):

1. Identify the cells formed by the intersection of input rows and output columns; that is, the solution area.

2. Perform the necessary elementary column operations to reduce to zero each entry in an output column other than those in the solution area.

3. The resulting values in the solution area represent the solution of the algorithm; that is, they show the number of ways in which each input "reaches" each output.

4. Since each elementary column operation may not only reduce to zero the cell which is being operated upon, but may also introduce non-zero entries into other cells which are supposed to be reduced to zero, it is wise to proceed in a systematic fashion, from the bottom row of the matrix up.

The S^* matrix contains in its solution spaces (heavily outlined in Figure 2) the same data as would have been contained in the solution obtained by matrix multiplication, except that they are in different orders. Homer claims the following benefits for his second method, in addition to solving the problem of non-conformability (14, p. 511):

1. No attempt need be made by the analyst to force reports into classification by level.

2. Computation is greatly simplified in that only elementary column operations are required.

3. Inputs and outputs of the system are immediately identifiable, as are components outside the scope of the study.

4. The sequence in which components are listed is immaterial. No time or effort need be spent in ordering data.

5. Because of its simplicity, the method is well adapted to simulating the effect of changes in the system.

6. Any intermediate results (analogous to some $N_{a,b}$ where $a \geq 0$, $b \leq n$) can be obtained by defining "input rows" and "output rows" to be the rows and columns of the matrix of the desired result, and forming the solution area accordingly. Any $N_{a,b}$ so obtained can then be used to compute $N_{a,c}$ $(c > b)$.

	$R_{1(1)}$	$R_{2(1)}$	$R_{3(1)}$	$R_{1(2)}$	$R_{2(2)}$	$R_{3(2)}$	$R_{4(2)}$	$R_{5(2)}$	B_1	B_2	B_3	B_4
d_1	0	1	1	0	0	0	0	0	0	0	0	0
d_2	1	1	1	0	0	0	0	0	0	0	0	0
d_3	0	0	1	0	0	0	0	0	0	0	0	0
d_4	1	0	1	0	0	0	0	0	0	0	0	0
d_5	0	1	0	0	0	0	0	0	0	0	0	0
d_6	0	0	0	1	0	1	1	1	0	0	0	0
d_7	0	0	0	0	0	0	0	0	1	1	1	0
$R_{1(1)}$	-1	0	0	1	0	1	1	1	0	0	0	0
$R_{2(1)}$	0	-1	0	0	0	0	0	0	1	1	0	0
$R_{3(1)}$	0	0	-1	1	1	0	1	1	0	0	0	0
$R_{4(1)}$	0	0	0	0	1	1	0	0	0	0	0	0
$R_{1(2)}$	0	0	0	-1	0	0	0	0	0	1	1	1
$R_{2(2)}$	0	0	0	0	-1	0	0	0	1	1	0	1
$R_{3(2)}$	0	0	0	0	0	-1	0	0	1	0	0	1
$R_{5(2)}$	0	0	0	0	0	0	0	-1	0	1	1	0
$R_{6(2)}$	0	0	0	0	0	0	0	0	1	0	0	0

Figure 1. S Matrix from Homer's Example (14, p. 509)

	$R_{1(1)}$	$R_{2(1)}$	$R_{3(1)}$	$R_{1(2)}$	$R_{2(2)}$	$R_{3(2)}$	$R_{4(2)}$	$R_{5(2)}$	B_1	B_2	B_3	B_4
d_1	0	1	1	0	0	0	1	0	2	4	2	2
d_2	1	1	1	0	0	0	2	0	3	6	4	4
d_3	0	0	1	0	0	0	1	0	1	3	2	2
d_4	1	0	1	0	0	0	2	0	2	5	4	4
d_5	0	1	0	0	0	0	0	0	1	1	0	0
d_6	0	0	0	1	0	1	1	1	1	2	2	2
d_7	0	0	0	0	0	0	0	0	1	1	1	0
$R_{1(1)}$	-1	0	0	1	0	1	0	1	0	0	0	0
$R_{2(1)}$	0	-1	0	0	0	0	0	0	0	0	0	0
$R_{3(1)}$	0	0	-1	1	1	0	0	1	0	0	0	0
$R_{4(1)}$	0	0	0	0	1	1	0	0	2	1	0	2
$R_{1(2)}$	0	0	0	-1	0	0	0	0	0	0	0	0
$R_{2(2)}$	0	0	0	0	-1	0	0	0	0	0	0	0
$R_{3(2)}$	0	0	0	0	0	-1	0	0	0	0	0	0
$R_{5(2)}$	0	0	0	0	0	0	0	-1	0	0	0	0
$R_{6(2)}$	0	0	0	0	0	0	0	0	1	0	0	0

Figure 2. S^* Matrix from Homer's Example (14, p. 510)

Subsidiary Literature

During the course of later presentation of the research, an analogy will be drawn between an information system and a materials flow system. The research will also draw upon the work of two authors in this area. It is therefore pertinent to present the work of these two authors at this point, even though their work was done in a context quite different from that of information systems analysis.

Vazsonyi (30) considered the widespread problem of a materials system wherein raw materials and purchased parts are used to produce a hierarchy of sub-assemblies and assemblies and, finally, finished product. He developed a technique known as the "Gozinto theorem"* wherein the statements of relation between individual parts in the system are used to develop a matrix which represents the "total requirements factor," or the amount of each lower level component required for a given higher level assembly or sub-assembly.

Vazsonyi does not actually state and prove a theorem, but rather defines some relationships existing in the materials system and develops a method of computing a total requirement factor matrix. The main line of his reasoning and development starts with the portrayal of the materials system as a directed graph, which he calls a "Gozinto diagram."

In this diagram, as shown in Figure 3, the several parts and assemblies are represented by the nodes (circles) of the graph, and the

* Vazsonyi introduces a note of humor into his book by a footnote which attributes this theorem to "the celebrated Italian mathematician Zepartzat Gozinto," with reference to his "Collected Works, Vol. 3."

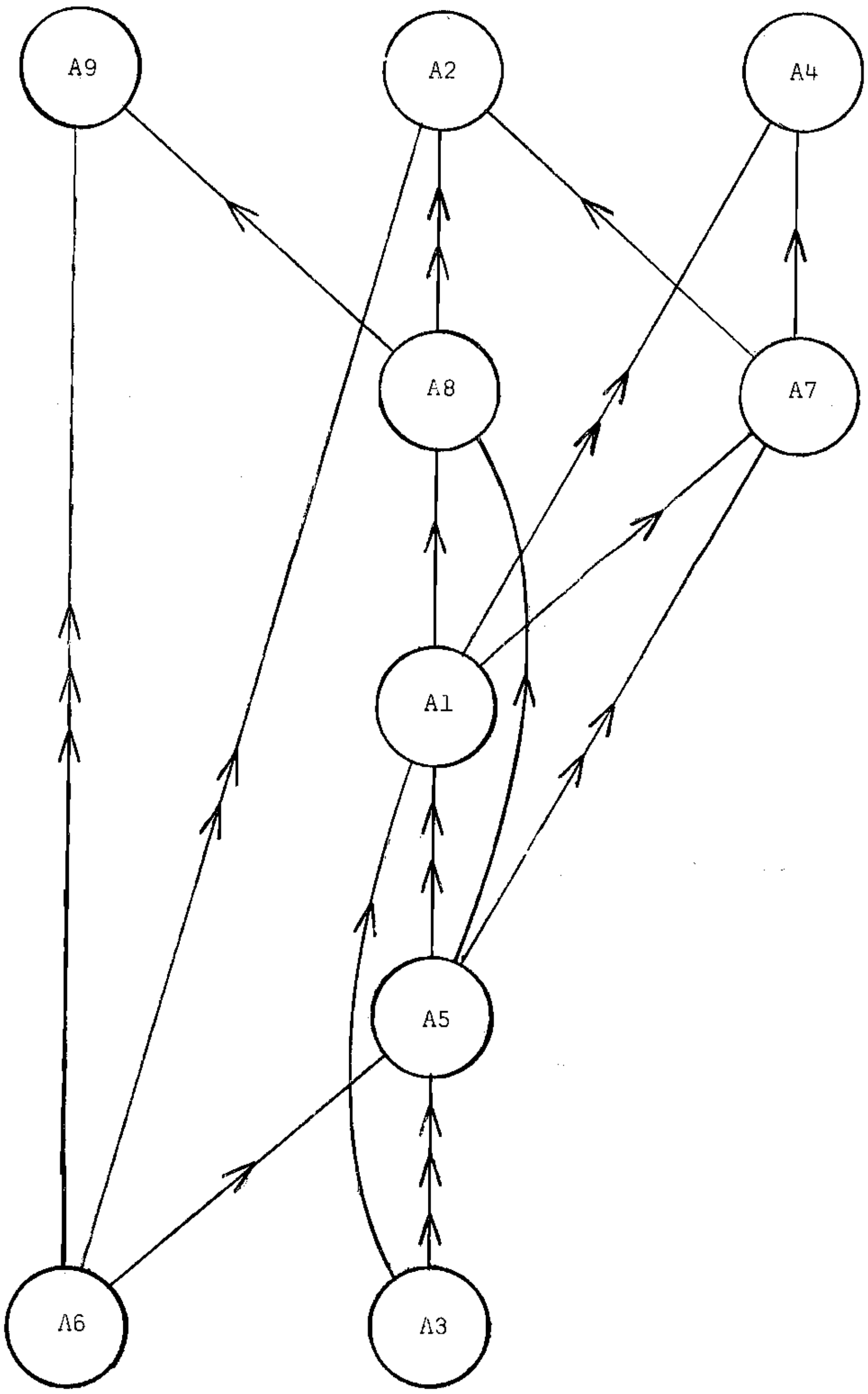

Figure 3. The Gozinto Graph (30, p. 430).

relationships which exist between parts and assemblies are denoted by lines connecting the nodes. The arrows on the line define both the direction of relationship and the number of a given part or sub-assembly at the beginning node which are required for fabrication of the sub-assembly or assembly represented by the node at the end of the line.

From this graphical conception of the materials system, the "next assembly quantity matrix" shown in Figure 4 can be developed by inspection. In this matrix, the row index i is the index of the node at which a line begins, the column index j is the index of the node at which a line ends, and the cell value is the number of arrows on the connecting line. The next step is to develop, again by inspection of the graph and simple calculation, a "total requirement factor matrix," depicted in Figure 5. Notation is then developed, wherein N denotes the next assembly quantity matrix, with n_{ij} the cells therein, and T is the total requirement factor matrix, with cells t_{ij}. Both N and T are of dimension n x n, where n is the number of parts and assemblies in the system. By appeal to the graphical representation and the two matrices, he is then able to show that the computation of the cells t_{ij} can be accomplished by use of the relationship:

$$t_{ij} = \sum_k n_{ik} t_{kj}, \; i \neq j, \qquad (1)$$

where k indexes over the columns of N and the rows of T.*

* While concise, this equation is not computationally efficient, requiring, as Vazsonyi notes, solution of large systems of simultaneous equations.

$$
\begin{array}{c|ccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
1 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 1 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \\
6 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 3 \\
7 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
$$

Figure 4. The Next Assembly Quantity Matrix (30, p. 431)

$$\begin{array}{c|ccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
1 & 1 & 3 & 0 & 3 & 0 & 0 & 1 & 1 & 1 \\
2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 7 & 33 & 1 & 27 & 3 & 0 & 13 & 10 & 10 \\
4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
5 & 2 & 10 & 0 & 8 & 1 & 0 & 4 & 3 & 3 \\
6 & 2 & 12 & 0 & 8 & 1 & 1 & 4 & 3 & 6 \\
7 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
8 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

Figure 5. The Total Requirement Factor Matrix
(30, p. 432)

Also, to make equation (1) hold, the relation

$$t_{ij} = 1, \; i = j \tag{2}$$

is defined, on an essentially arbitrary basis, but with the justification that this diagonal 1 reflects the inherent self-identity of a part.

It is next observed that equation (1) is a matrix multiplication, wherein N is post-multiplied by T, and that equation (2) represents the addition of the identity matrix (I) of rank n. Therefore, the whole relation can be represented by the matrix equation:

$$T = NT + I \tag{3}$$

It is then noted that this equation can be successively rewritten:

$$T - NT = I$$

$$(I - N)T = I$$

and finally:

$$T = (I - N)^{-1} \tag{4}$$

if the inverse exists.*

The T matrix is therefore the inverse of the (I - N) matrix, where I is the identity matrix and N has known values. Vazsonyi notes at this point that the matrices are triangular, and that for this reason the inversion is relatively easy to perform. He does not exploit this triangularity to any extent, however.

In a later work Giffler (9), in the same context of determination of total requirements factors in a materials assembly system, more fully explored the properties of the N, T, and (I - N) matrices. He defines three matrices, N, T, and C. N corresponds to a bill of materials, and the typical element n_{ij} is the quantity of "i" which is directly consumed per unit of "j." T corresponds to "total requirements," with elements t_{ij} being the quantity of "i" which must be manufactured or purchased per unit of "j." C is a "consumption matrix" where the c_{ij} represent the total quantity of "i" consumed, directly or indirectly, per unit of "j." Giffler's N and T matrices are precisely the same as those defined by Vazsonyi. All matrices are of dimension n x n.

It is first shown that the consumption matrix C is related to the bill of materials matrix N by:

* Vazsonyi does not point out that the relation between equations (3) and (4) depend on the existence of the inverse of (I - N), which in turn implies that this matrix must be non-singular. However, this is easily shown, since N is strictly triangular, and therefore (I - N) is triangular. In a triangular matrix, the determinant is equal to the product of the diagonal elements. In this case, (I - N) has 1's on the diagonal, the determinant is equal to 1, the matrix is therefore non-singular, and $(I - N)^{-1}$ exists.

$$c_{ij} = n_{ij} + \sum_k c_{ik} n_{kj}, \tag{5}$$

that is, that the consumption of i per unit of j is the sum of the direct consumption and the indirect consumption through the kth commodities (sub-assemblies) which in turn are consumed directly in i.

This relation is then expressed in matrix notation,

$$C = N + CN \tag{6}$$

which by appropriate manipulation becomes

$$C = N(I - N)^{-1} \tag{7}$$

where I is the n x n identity matrix.

It is then noted that if the matrix N is squared, the matrix N^2 will have elements

$$n_{ij}^2 = \sum_k n_{ik} n_{kj} \tag{8}$$

where n_{ij}^2 represents "second-level consumption" of i per unit of j. By extension, n_{ij}^m would represent mth level consumption of i per unit of j in the matrix N^m, and direct consumption would be equivalent to n_{ij}^1 in the original N matrix. By this argument, the consumption matrix C for an m - level system can be written:

$$C = N + N^2 + N^3 + \ldots + N^m. \tag{9}$$

It is then shown that since the consumption of commodities (parts and sub-assemblies) in other commodities is finite, the series on the right hand side of equation (9) terminates, that is, n^{m+1} is null for some finite m. It is then observed that the elements of T are related to those of C by:

$$t_{ij} = \begin{vmatrix} c_{ij}, & \text{if } i \neq j \\ 1 + c_{ij}, & \text{if } i = j \end{vmatrix} \tag{10}$$

which can be expressed by:

$$\begin{aligned} T &= I + C \\ &= I + N + N^2 + \ldots + N^m. \end{aligned} \tag{11}$$

Multiplying both sides by (I - N),

$$(I - N)T = (I - N)(I + N + N^2 + \ldots + N^m) \tag{12}$$

and performing the indicated operation on the right and cancelling yields

$$(I - N)T = I \tag{13}$$

and

$$T = (I - N)^{-1} \tag{14}$$

which is precisely the result obtained by Vazsonyi.

Giffler then exploits the triangularity of the N, (I - N), and T matrices, in particular the fact that (I - N) is the forward solution of its own inverse, to develop a pair of iterative equations for the computation of $(I - N)^{-1}$ when N is strictly triangular and non-negative. Assuming upper triangularity,

$$\begin{aligned} t_{ij} &= \sum_{k=1}^{j-1} t_{ik} n_{kj}, \qquad j > i \\ &= 1, \qquad\qquad\qquad j = i \end{aligned} \tag{15}$$

while in the case of lower triangularity,*

$$\begin{aligned} t_{ij} &= \sum_{k=j}^{i-1} n_{ik} t_{kj}, \qquad j < i \\ &= 1, \qquad\qquad\qquad j = i \end{aligned} \tag{16}$$

These equations require solving for rows from top to bottom in the case of equation (15) and for columns from left to right in the case of equation (16). As will be developed later, this is not the most

---

* Giffler stated this equation erroneously, in that his summation was from k = i to j - 1, which does not give the proper result.

desirable order of solution for computer operation.

In order to fully exploit the triangular nature of the N and T matrices, it is necessary to have a method of arranging the arbitrary designations of parts, sub-assemblies and assemblies in such a manner that the resulting matrices will be in the triangular form desired. Giffler (9, p. 18-34) provides the essential basis for triangularization. He first approaches the matter from the point of view of the cells within a matrix, and notes that triangularization can be done by a series of row and corresponding column permutations of a matrix that is triangular in nature, but not in form. He also defines pairs of permutation matrices which can be devised, and used to pre- and post-multiply a given matrix in order to triangularize it. Since this technique requires prior determination of the permutations to be made, it does not appear to be suitable for machine operation, and will therefore not be discussed further.

By looking at the essential nature of a triangularized N matrix (in the material assembly system) Giffler observed that triangularity could be achieved if the commodities and parts could be so ordered that "no commodity follows a commodity in which it is consumed and that assigning commodities to successive rows and columns in the same order must produce a (strict) triangular N." This definition will produce an upper triangular matrix. For lower triangularity, the word "follows" in the quotation above is replaced by "precedes."

Based on this definition, Giffler then develops the following algorithm:

1. For each $n_{ij} \neq 0$, list the pair (i,j). Assuming that

there are m such pairs, assign the successive position numbers 1, 2, . . ., m to each component index i. Assign the position numbers m + 1, m + 2, . . . , 2m to each assembly index j. Let (i, m + i) represent the position numbers of the ith pair.

2. Select the highest positioned component index. Assume that it occupies position i. (Actually at the outset it must occupy position 1.) Look for the first match of the index against an index in positions m + i, m + i + 1, . . . , 2m. If a match cannot be made repeat Step 2 with the next highest component index. If no match can be made with any component index, transfer to OUT.

3. Say that Step 2 produces a match with the index in position m + j. In this event exchange the indices occupying position i and j and the indices occupying positions m + i and m + j. Make a record of the exchanging indices.

4. Compare the exchanging indices in Step 3 with previously exchanged indices. If the indices have exchanged previously, transfer to TERMINATE. If the indices have not exchanged previously, repeat Step 2 with the next-highest component index.

TERMINATE. If the algorithm reaches this step, N is cyclic.

OUT. If the algorithm reaches this step, N is non-cyclic.

When the algorithm has run to OUT, all component indices will be listed in their (strict) triangularizing order. The indices of the products will be missing because they are not components. This deficiency can be corrected by adding a zero'th row and column to N and making all $n_{i0} = 1$ if i is the index of a product. This is the same as to assume a common fictitious commodity in which all products are consumed.

Giffler asserts that the above method is "readily mechanized by punch card collating equipment." The method produces an upper triangular matrix, but by reversing i and j indices, the algorithm will lower triangularize as well.

Summary and Conclusions

To summarize, relatively little has been done in a quantitative sense for either interior on exterior systems design. As currently practiced, information processing systems design is largely an art, rather than a science. The use of compilers, especially generalized compilers such as COBOL, is extensive, but these compilers are widely criticized. Decision tables seem to be a desirable adjunct to the compilers for internal systems description and analysis, but many of the criticisms directed toward the compilers is also directed to decision tables. Generalized routines, such as sort routines and report generators, are used extensively, but also have the limitation of requiring valuable computer time in order to save programming effort.

Some advance has been made by the Lieberman and Homer models, but the techniques are not widely used. It is believed that additional work in the direction indicated by these models is justified. Symbolic logic has been suggested as a means of analysis but has the limitation of difficulty of manipulation which renders it relatively weak as either an exterior or interior systems design tool.

In conclusion it is appropriate to quote from two rather similar articles which point out the need for investigation of the sort contemplated, without, however, indicating the direction which the investigation should take! Postley (24) comments:

> But while new triumphs in scientific computing have been derived from a sound base of mathematical theory and research, no such base exists in the business field. The principles of accounting and record keeping operations are reasonably well-defined for non-computer-oriented systems. But these principles must undergo substantial revisions when we consider . . . applying them to data processing equipment, . . . Thus, a second dimension in the classification of scientific or business data processing is the presence of well developed mathematical and engineering theory for scientific information systems and the almost complete lack of a fundamental theory of business information systems.

About a year later Steel (28) stated:

. . . data processing is not really <u>clearly understood</u> by anyone. . . . Any attempt to bring some order into the present chaos must face the unpleasant fact that no fundamental theory exists for immediate application. This is the key difference between commercial data processing and scientific computing. . . . The absence of a corresponding fundamental discipline underlying data processing requirements has prevented the parallel development of technique for those problems whose solution now seems to occupy the bulk of computer time in the world.

CHAPTER III

SCOPE OF STUDY AND BASIS FOR IMPROVEMENT

Introduction

The purpose of this chapter is to present a brief overview of the work to be shown in detail in following chapters, and to discuss certain matters pertaining to prior work in the field and certain characteristics of information processing systems which will furnish a basis for improvement of techniques of systems analysis.

The research to be presented has three basic phases. In phase I, the work of Homer and Lieberman will be duplicated with improvement in terms of the amount of information made available about the system under study and in terms of ease of computation. Chapter IV will present the mathematical development, an algorithm, and a numerical example illustrating this phase.

Later in this chapter it will be shown that while this first phase represents an improvement, the model does not adequately represent an information system in that it does not embrace consideration of types of data and data transitions. Phase II therefore presents a mathematical development, algorithms, and numerical examples for two refined techniques. Chapter V will contain the mathematical development, while Chapter VI will present the algorithms and numerical examples.

Since the study of any real system would result in such a number of systems relations that the algorithms could not be used on a manual basis, phase III of the research will be devoted to a discussion and development of the use of mechanical and electronic equipment for obtaining numerical solutions using the refined algorithms. The algorithms were, in fact, developed from the point of view that computer methods rather than manual methods would be used in practice. Chapter VII will contain a general discussion of computer considerations, while details of machine operation will appear in the Appendices.

Considerations for Refinement

While Chapter IV will present a method which will reproduce the results of Homer and Lieberman with improvements of computability and usability, it will be pertinent to examine critically both the prior work and the phase I work of this writer with a view to substantial refinements which will permit a more widespread use of the techniques in analysis of real systems. As a first step in this direction, it is pertinent to develop and discuss the types of data which may be found in an information processing system, and to develop the types of transitions through which these data may pass between the input and output stages. Based on this, the limitations of the prior work and the phase I work may be seen, and refinements made.

Data Types

The data which arise in an information processing system may be classified into three basic types, based entirely upon the essential nature of the data rather than its source or intended usage. The three types may be designated as identification data, quantitative data, and status or existence data.

Type one, identification data, is represented by such items of data as payroll numbers, stock numbers, machine numbers, dates, times, locations, and the like.

Type two, quantitative data, is usually originated as a result of a count, measurement, or computation. Included are such data as the amount of product produced on a machine, the tensile strength of a sample subjected to a quality test, the number of pounds shipped on a shipment, the number of employees absent from a given shift, the amount of payment made to an employee or a vendor. Type two information is used largely, but not exclusively, at the lower echelons of the enterprise, for the purpose of making immediate decisions, conveying instructions and short term results, and preparing working documents such as invoices and vouchers.

Type three data are characterized as those data which indicate status or existence, e. g., a specific commodity is assigned to a machine, a failure has occurred in a given process, a given quality test was performed, a specific shipment has been made. Frequently this is the type of information which the control groups and administrative functions of an enterprise are concerned with, since the mere existence of an event is often sufficient for some action to be taken. For example the fact that a shipment has been made is an indication to the order-invoice section that the customer should be billed.

It should be noted that data may be, and frequently are, converted from type two to type three data merely by taking the presence of some quantitative data as indication that some action has taken place. For example, the existence of a non-zero entry in the hours worked column of an employment record is construed as evidence that the employee was present. This conversion of type two to type three data is not in general reversible, nor can the original type two data be regained unless specific provisions are made for its storage and subsequent retrieval.

It is also important to note that neither type two nor type three data are meaningful unless accompanied by the proper type one identification data. Frequently a single item of identification data suffices for several pieces of type two and/or type three data, but if some of the type two or type three data are removed by some processing step, or if type two data are converted to type three data, or if some other type of data conversion is made, the type one data must still be reproduced in order to identify the surviving data. One implication of this fact is that apparent redundancy of type one data in an information system cannot be viewed in the same light as redundancy of type two or type three data, since the identification data <u>must</u> carry through the system in substantially unchanged form on every document and report.

Furthermore, it is not uncommon for a single item of type one or type two data to carry with it several items of type one data. For example, the type two data "pounds shipped" will very likely carry a date, invoice number, customer number, product number, bill of lading number, packing slip number, etc. In a payroll sub-system, the gross wage may be identified by employee number, department number, social security number, date, shift, and other information required to reflect the fact of paying the employee in the several pertinent accounts.

A consequence of this concept of data types is that it may not always be wise, in practice, to analyze the type one data separately from the type two or type three data which it is forced to accompany. It may be better from the point of view of both analysis and synthesis of the information system to view the type one data as either coupled to some corresponding type two or type three data, or to view the type one data as a subset of the appropriate type two or type three data, or perhaps merely to view the type one data as a constant running throughout the given sub-system.

One further concept is of interest in trying to reduce the overall amount of data in the system. Since in many cases type three information can be generated or deduced from appropriate type two data, it will be worthwhile to examine instances of creation or maintaining type three data with a view to obtaining the same result as a by-product of the necessary creation or maintaining of the corresponding type two information. Any means of analysis should furnish a means of examining type three data items, particularly when both types of data pertaining to a given event are carried on the same document or report, or are used by the same business function.

Data Transitions

One of the primary functions of an information processing system is to perform prescribed transitions on the input data in order to make it more meaningful to the intended user. Information has been defined as "processed and organized data." It is therefore appropriate to examine the essential nature of the data transitions which take place in a system, and to characterize them in a manner to make them amenable to description and analysis. The following transitions can take place:

1. Data can change types. As discussed in the preceding section, the presence of quantitative data (Type two) may be interpreted as indicating status or existence (Type three). The important fact here is that the data are not the same, and should not bear the same identification for analysis.

2. Data may terminate. This usually arises when a report is extracted for a higher level report, leaving behind unwanted data.

3. Data are summarized, with only the total going on to a higher level. A common example is that the number of hours worked by an individual employee is summarized by department, and only department totals are carried forward in the system.

4. Data are computed, with only the results being preserved in upper levels of the system. An example is that pounds shipped times price per pound may be computed, and only the extension used in some higher level documents.

5. Data are generated in the processor or process, and used either as an intermediate step in a computation or as a basis for comparison. In some cases these data never appear on a report as such, but are lost after the processing purpose for which they were created has been fulfilled. A prime example, but not the only one, is the generation of "proof totals" in a computer run to assure that all data were included in the run.

6. Data, either input or processed, are stored and later retrieved. These delayed data are usually either in original form, such as "last months sales," or in cumulative form, such as "sales year-to-date."

7. Data may be generated on the passage of time. For example in a hospital billing system, the room charges are created by the fact of a room being occupied at midnight. In industrial context, burden charges are (artificially) presumed to be generated by the passage of a calendar month. This transition is, in effect, a special case of data origination, but since it usually arises without overt action having taken place, it can also be considered as a data transition.

8. Occurrence-generated data are those data which arise based on the operation of some decision rule. For example, if three successive absences of an employee occur, this may occasion the preparation of a report to the plant medical section for the purpose of checking whether such absence is due to medical reasons. This data transition almost always gives rise to origination of type three data, and frequently calls for the recovery of stored data as well.

9. System constants are those data, usually of type two, which are used repetitively and largely automatically in computations. For example, in quality control computations, the figure "3" is frequently used as a constant by which the computed standard deviation is multiplied in order to establish control limits. In payroll computation, the number "40" is frequently used when it has been established that an employee has worked the standard work week. This type of transition may also be viewed as a special kind of data origination.

The principle effect of these data transitions from the point of view of the analysis of a system is that, in general, input data do <u>not</u> survive in pure form through the system and into the higher level reports, except for the type one identification data. Rather, the higher level reports consist largely of data which have been summarized, computed, and otherwise derived through the transitions of input data. Therefore, the analysis technique must be such that these transitions are explicitly recognized and properly treated.

A Materials Flow System Analogy

Since phase I of the current research is based on a prior work done in the area of materials flow systems, it is fruitful to briefly discuss these systems, and to draw an analogy between a materials control system and an information processing system. It will later be shown that the analogy fails at critical points, which furnish a basis for further refinement.

The materials flow system is characterized by input to the system in the medium of raw materials and purchased parts, intermediate stages of sub-assembly and assembly, commonly denoted by the term "in-process inventory," and final stages of finished product. In addition, some of the intermediate assemblies may be sold as such for replacement purposes, or for use by other manufacturers for purposes of further assembly.

The computation usually associated with this system is called the "materials requirement determination." Given a production forecast or schedule in terms of finished products and saleable intermediates, the task is to compute the amount of raw material and purchased parts, and the number of assemblies, sub-assemblies, sub-sub-assemblies, etc., which will be required in order to support the production commitments.

As a manual task, this computation is one which is tedious, error-prone, and expensive. It was therefore one of the first, after the payroll, to be considered as a punch-card application when such equipment became available. The IBM Corporation, realizing the broad requirement, several years ago developed the so-called "explosion" technique, which used the abilities of conventional card equipment to do this task. While this card "explosion" was a major improvement over manual methods, and made effective use of then available equipment, the technique had one inherent flaw. This was that each level of assembly required a separate, sequential "pass" through the processing equipment, starting with the finished product and ending with the raw materials and purchased parts. This also implied that the order and structure of the materials system be carefully determined, and that the level of entry of each component be carefully specified in advance of the computation. This determination was usually made by tedious manual methods, and usually by engineering personnel who could understand drawings and bills of material.

With the advent of stored program digital computers, it was natural to utilize the higher processing speeds and improved machine logic to improve the method. However, the same inherent problem of multiple passes, one for each level, remained, due to the inability to logically relate the several levels of assembly. At the present time almost all processing is done by this technique, however, despite the limitations involved.

As cited in Chapter II, Vazsonyi (30) developed and reported the so-called "Gozinto" method of materials requirements determination, while Giffler (9) improved the technique somewhat and furnished certain essential proofs. This writer (3) modified and refined the technique for the purpose of materials requirements determinations, and showed the feasibility of the technique under certain circumstances of assembly hierarchy and computer memory configuration.

The analogy between the assembly of products and the structure of a business information system is readily drawn. The origination of data at some place where an event takes place corresponds to "raw material." Data arising from other systems or exterior sources can be considered as "purchased parts." Sub-assemblies in the plant correspond to work sheets, intermediate reports, and working decks, tapes, and records. Prepared reports compare to finished product, and it can be noted that within the report structure there are levels in the same sense that there are levels of assembly. It should be noted that the analogy is not perfect, and that many circumstances arising in a data system do not find a counterpart in the materials system. However, these difficulties will be discussed and treated in a later section. In this section, it will only be noted that the basic "Gozinto" technique, with some modifications of computational procedure discussed in Chapter IV, can reproduce the results of Homer and Lieberman with full preservation of system relationships and improvement of computational nicety.

Criticism of Prior Work and of Current Phase I Work

The Homer-Lieberman Models

In the Literature Survey, certain criticisms of the Lieberman model were made by Homer, and Homer proposed two methods to alleviate these difficulties. These methods represent a vast improvement over the basic technique originated by Lieberman, by eliminating the problem of non-conformability of matrices, and indeed, by the elimination of the necessity for performing a whole series of matrix multiplications. In addition, the benefits claimed are for the most part desirable ones. However, some of Homer's cited benefits* warrant some discussion.

In benefits one and four, the point is made that the analyst need not spend time in ordering data, nor in classifying the reports into levels in order that the computation technique will function. This is quite true. On the other hand, in a real situation, it is desirable to know the report levels, and an ordering of the data is very helpful in understanding the system under study. It is significant to note that Homer, in his example, did order the sequence of data and reports with respect to their levels, even while pointing out that it was not necessary to the processing of his model.

In benefit six, a procedure is outlined by means of which intermediate results can be obtained. This is not seen to be a benefit, but rather a loss of information over the basic Lieberman technique. While it is true that the information can be recovered, it requires additional computation, which tends to reduce the effectiveness of the model.

In benefit five, and in benefit two, the simplicity and ease of computation are cited. It is true that the method is an improvement over the matrix multiplication method. However, the performance of column operations, while not difficult, is time consuming, and would be tedious for other than a trivial problem. While it is true that the

---

* See Chapter II, page 31.

technique could be programmed for a digital computer, it would require extensive memory search techniques which would be difficult to program and time-consuming to operate.

Perhaps the greatest criticism of the technique is that much of the data is lost in the computation, and because of the lack of ordering (unless particular effort is made to do so), the results are difficult to interpret. In addition, after the S matrix is formed, a search of the matrix is desirable, if not required, in order to eliminate non-applicable or erroneously included data from the matrix.

In view of the foregoing criticisms, it is then desirable to develop a technique and computational scheme which will meet Homer's original objections to Lieberman's model, to achieve the benefits inherent in Homer's model, and to gain such other benefits as may be possible. The model developed in Chapter IV is one such technique.

The Phase I Model

While the Phase I Model to be presented in Chapter IV will alleviate many of the short-comings of the Homer-Lieberman models, several other criticisms pertain uniformly to both the prior models and the Phase I model. Since the refinements in later chapters will be attached to the model proposed by this writer, however, the criticisms will be discussed in the light of this latter model.

The two essential criticisms of the basic algorithm is that no distinction is made of data type, and that no provision is made for recognition of the data transitions which arise in the processing. These shortcomings result in a final solution matrix, S^*, which does not give an adequate representation of the system under study, and which in fact may be misleading.

A principal effect on the S^* matrix is that redundancy tends to be overstated for all three types of data. The inability to recognize type one data also leads to attaching undue importance to necessary duplication of identification data. The inflated indication of redundancy results from the fact that the model as constituted carries through the system data which in fact have been lost due to one of the transitions having taken place.

The second effect on the S^* matrix brought on by this same failure to recognize data transitions is that upper levels indicate the presence of information which in fact is not available from the report in question. It is true that the data may have been required in the report preparation, but they do <u>not</u> appear in the report as such. In this sense, the S^* matrix is misleading, and can lead to erroneous conclusions in the systems design.

The failure to recognize data transitions has also resulted in a model which considers only the external reports of the system, and which has neglected entirely the more voluminous internal reports, working papers, summary decks, tape records, etc. The volume and number of these latter is frequently much greater than the published reports, and are extremely important in designing a well functioning system. Some of this prior omission can be rectified by merely making a more detailed analysis, but recognition of data transition type is necessary for full inclusion of these important considerations in the systems analysis.

Some of the types of transitions lead to specific difficulties of computation or interpretation of the S^* matrix. These are detailed below:

1. As previously indicated, several of the transition types lead to overstatements of redundancy. This occurs on changing of data from type two to type three, termination, summarization, and computation. Redundancy from the first two transitions is of a fairly simple nature in that an indication of the changed or terminated data is carried through into the higher level report structure as shown in the S^* matrix. Redundancy indications from summarization and computation is more malignant, in that it compounds much more rapidly. For example, if five daily reports are summarized into a weekly total, the S^* matrix will give the indication that the final report includes five daily figures; it may contain none whatsoever.

2. Stored data creates the problem of non-triangularity, or cycling, in the S matrix. For example, presume that a report R_1, containing weekly payroll totals, is stored for a week, then retrieved in order to use the prior week's total as a comparison for the current week's payroll total on the current week's R_1 report. This would result in the incompatible description that report R_1 is required for the preparation of report R_1. This is true in a sense, but the analysis technique must recognize that last week's report is not the same entity as the current report. They are alike in data format, but entirely different in data content.

3. There is danger that time-generated, occurrence-generated, and constant data may not be properly recognized, and that their effect on the system may be lost. This difficulty is largely one of care in examination and description of the system rather than inadequacy of the

mathematical model. It should be noted, however, that the examination of occurrences to determine whether the criteria for generation of data has been met may appear to cause redundant use of the information when such is not really the case.

In addition to the difficulties caused by data types and data transitions, there are further difficulties arising from the nature of analysis desired. The S* matrix shows essentially for each report the lower level reports and input data which are prerequisite for the preparation of the given report, or, to state it in another way, the lower level reports and data which the upper level report is "composed of." While this is important, it is also important for the analyst to have a concise statement, preferably in the same format, of the actual <u>data content</u> of each report. In the present form, the S* matrix not only does not contain this information, but is misleading in this respect in that it indicates "composed of" information which may be erroneously interpreted as being "contained in" information.

A very frequent data processing operation is that of "match-merging" wherein two reports (usually in card or tape form) are collated, or merged into physical or conceptual juxtaposition by the device of matching type one data between the two reports. For example, one input card may contain a payroll number and the number of hours worked. An internal master record is maintained which contains payroll number, rate of pay, and department number. A match-merge is performed, comparing the two files on payroll number, and physically placing together each master card with the associated transaction card(s) containing the same payroll number. This operation produces a very great indication of redundancy of the identification number, when in fact only a single statement of the identification is carried forward through the system. This fact lends further credence to the previous statement that apparent redundancy of type one data is not necessarily a cause for concern.

A further cause of difficulty in using the existing algorithms lies in the fact that in defining the relationships which exist between data elements, there is no single, unique way in which a given information system must be viewed. Depending upon the circumstances, and, unfortunately, the point of view of the analyst, there are several viewpoints which might be taken. For example, one might choose to look at a given system as a hierarchy of reports, with only secondary consideration given to the data and computations involved. Again, the same system might be viewed with primary emphasis placed on the data content, and secondary consideration of the specific documents and reports upon which the data appears. A third viewpoint might emphasize the organizational aspects of the system, with principal attention given to the organizational level responsible for report preparation. In practice, all of these points of view are used, frequently on the same study.

Unfortunately, the difference in viewpoint can be reflected in a difference of definition of the relationships between systems elements. Since the S and S^* matrices are functions of the defining relationships, this means that the difference in viewpoint can be reflected by real difference in the final S^* matrix. In other words, there is no assurance that S^* offers a unique representation of the system under study.

Based on the foregoing discussion, the refinement of the model for systems analysis must then permit and indeed demand that proper recognition be given to data types and data transitions. In addition, this refined technique must also provide means of eliminating or minimizing the other noted criticisms. Refinements will fall, in general, into two related categories:

1. Improved definition of relationships between data systems elements, by recognition of the types of data and of data transitions. These definitions will be made in the next section of this chapter.

2. Development of algorithms which will permit the inclusion of these new definitions, and which also furnish a means for systems analysis on a basis of the content of reports in addition to the former sole basis of composition of reports. This development will be the subject of Chapters V and VI.

Relationships Between Systems Elements

Consideration of the basic data types, data transitions, and the essential nature of information systems indicates that the relationships which exist between systems elements may be placed into three categories, rather than the single category assumed in the prior work and in the current phase I work. Before describing these three categories it is desirable, however, to define the term "system element" more rigorously.

Definition of System Element

1. By "simple element" will be meant any piece of recorded data, numerical, alphabetical, or symbolic, which exists in the system as an input, output, or intermediate piece of data, regardless of the form of

recording.

2. By "compound element" will be meant any report, form, document, card, tape, or disk record of any nature containing one or more pieces of recorded data (simple elements), and from which the simple elements can be recovered and made meaningful to humans without arithmetic computation.

3. Where it is not necessary to distinguish between simple and compound elements, the term "system element" or simply "element" will be used.

By the above, a record kept on magnetic tape of each employee would be a compound element, and each individual piece of data within this record, such as payroll number, name, address, wage rate, etc., would be a simple element. However, the figure "Average weekly pay for the past six weeks," if it required a computation based on simple elements stored within this record as opposed to being already stored, would not be a system element.

Definitions of Systems Relations

The three categories of relationship between systems elements may now be described and defined. The relationship implied in the Homer-Lieberman models and in the Phase I model is essentially a relationship of composition. That is, the relationship merely indicates that a system element of lower order is required for the preparation of an element of higher order. There is no implication of computation, nor of change of data type from type two to type three. This relationship will be defined, for the refined algorithms, as a "prime relation."

Definition of Prime Relation. By prime relation will be understood that relationship which indicates that an element at some given level of a data system is merely required for the preparation of some higher level element. By this definition, all data input is prime, as well as some elements at higher levels.

Definition of Concomitant Relation. A second category of relationship involves computed data, and the transition of type two data to type three data. In this relationship, one or more simple elements undergo a transition, either from type two to type three data, or through the operation of some arithmetic process or logic process. In this process, a new piece of data is created, and some or all of the lower level data may be lost. An important factor in a relation of this sort is that it always occurs in conjunction with the preparation of a compound element (record or report) of some sort. It has therefore been defined as a "concomitant relation."

By concomitant relation will be understood that relationship which occurs during the preparation of a record or report wherein an arithmetic operation involving simple elements of an order lower than the level of the element being prepared, or wherein a transition from type two to type three data takes place. The constituent elements of the computation, or the type two data from which the type three data were inferred may or may not also appear in the new compound element being derived. If they do appear, they will also bear prime relationships.

Definition of Deletion Relation. The situation frequently arises in an information system that a record, document, or report which is used for the preparation of a higher level element contains more basic elements than are required. In this case, the necessary information is extracted from the lower level report, and the unwanted data are terminated, or from the point of view of the balance of the system, are deleted. Therefore a third category of relationship is defined, called a "deletion relation."

By deletion relation will be understood that relationship which exists when an element from a lower level is brought to a higher level by virtue of being included within a compound element which is required at a higher level, but which in itself does not become an integral part of the higher level.

A deletion relation is a negative relationship, and even more strongly, is a blocking relationship, in that it is used to indicate that an element is terminated, absolutely and finally, on a given route through the system. This relationship will prevent a false indication of a given piece of information being available on a report when in fact it was only used at a lower level.

As will be shown in Chapters V and VI, the proper use of the prime and concomitant relations will eliminate much of the false redundancy indications in the S^* matrix, and will produce a matrix that represents the system under study on a "composed of" basis. The use of all three types of realtions will result in a final matrix, S'^*, which will represent the system on a "contained in" basis.

Rules for Treatment of Data Types and Transitions

With the relations defined, rules can now be given for treatment of the three data types and the several types of data transitions.

1. It is not necessary to make formal distinction between the

three types of data. The defined relationships will enable adequate analysis to be made. Type one data will give false indications of redundancy, but since this factor can be readily checked, it is not felt desirable to take the step to eliminate it.*

2. The use of prime relations is straightforward, and can be applied according to the definition, noting that all input has a prime relation to the compound element on which it is recorded.

3. The following transitions can be classed as concomitant:

   a. Changes from type two to type three data.

   b. Summarizations.

   c. Computations.

   d. Within-processor generation.

   e. Occurrence-generated data.

4. The following transitions may be treated as deletion relations if the definition is met:

   a. The type two data when a transition to type three has been made.

   b. Data terminations.

   c. Summarizations. (The undesired factors.)

   d. Computations. (The undesired factors.)

   e. Within-processor generation.

   f. Occurrence-generated data.

---

* The redundancy of type one data is caused primarily by the fact of merge operations, as previously discussed. If one desired to eliminate this redundancy, the judicious use of concomitant relations would do so. However, this researcher feels that the indication of number of merges is desirable as a measure of system complexity.

5. Time-generated data should be treated as input data, but may then be subjected to the deletion relation if the definition is satisfied.

6. System constants should be treated as input data, and may then be subjected to the concomitant and deletion relations if the definitions are satisfied.

7. Data which are stored, then retrieved after a fixed time to act as comparative data should be treated as new input, and _should therefore be given a different designation_ in the analysis from that which obtained when the element was stored.

Use of the New Techniques

It is desirable at this juncture to give a general overview of the phases of a typical analysis using the algorithms to be developed, so that these algorithms may be viewed in the broader context of systems analysis. It should be noted that the algorithm presented in Chapter IV is not intended for application, but is only a developmental step leading to the refined algorithms presented in later chapters.

1. The first analysis step is that of identification of the systems elements and definition of the system relations, together with their quantifying symbols which will be introduced in Chapter V. The sources of information are the usual ones encountered in data systems study--the reports, documents, and records themselves, procedures manuals, existing flow charts, and the like. Note that it is _not_ necessary to prepare flow charts of the system, nor is it necessary to display the relationships in matrix form in actual practice. In some

cases, such visual displays may be useful conceptual aids, however.

2. Card decks containing the specification of the system relations are prepared, so that the algorithms can be used for mechanical and electronic computation of the system matrix. This is a simple clerical task, best done by someone other than the analyst.

3. The third step is to perform the necessary computations of the algorithms to be presented in Chapter VI. In practice, this computation will be entirely by mechanical and electronic equipment. The output will be the S^* and S'^* solution matrices, representing respectively a "composed of" and a "contained in" analysis of the system.

4. These output matrices will then be scrutinized by the analyst. Further comment relative to the interpretation to be placed on these matrices will be made in conjunction with the later presentation of a numerical example. At any rate, the analysis of the matrices will suggest certain actions to the analyst. The possible results of these actions can be simulated by making appropriate changes in the definitions of the system relations, and recomputation of the solution matrices. In a complex system, this repetition may take place several times during the design and development stages, as new ideas and concepts are introduced.

Finally, a firm design is established, and the final solution matrices and the cards representing the system will find a new role as a convenient and concise means of documentation of the system as designed. Furthermore, any required redesign may be accomplished with a minimum amount of restudy and reprocessing.

CHAPTER IV

THE BASIC MODEL

Introduction

In this chapter, use will be made of the analogy drawn in Chapter III between a materials flow system and an information processing system in order to modify certain work done in the field of materials flow system analysis for use in the present context. The work of Vazsonyi (30) and Giffler (9) will be drawn upon and modified in order to present a mathematical basis for the basic model. An algorithm will be developed on this mathematical basis, and Homer's (14) example will be used to demonstrate that the basic model developed preserves the solution obtained by Homer. Finally, the basic model will be discussed briefly, pointing out its benefits over the Homer technique.

The Mathematical Basis

As was discussed in Chapter III, a strong, but not completely valid, analogy can be drawn between a material assembly system and an information processing system. Indeed, it is common practice to represent an information processing system graphically as a flow diagram or flow chart. These flow charts are subject to exactly the same interpretation and conversion to matrix form as Vazsonyi's "Gozinto diagram" shown in Figure 3, and the flow diagram of Giffler's presentation (9, p. 4). The symbols used in the flow chart represent the systems elements, while the connecting lines represent the relations which exist between the elements. It is not usual in flow charts to express the number of lower level elements required for the next higher level, but this information could readily be added if desired.

An S matrix may then be defined as a matrix representation of the flow chart of the information system under study. The cells s_{ij} may be defined as the number of times a given system element i is used directly in preparation of system element j, which conforms exactly to the definition of n_{ij} in the materials assembly system.* In this basic algorithm, in order to conform to the practice of Lieberman and Homer, a cell s_{ij} will be equal either to one, if a relationship exists, or to zero if no relationship exists. This is an unneeded restraint to the basic algorithm, however.

Similarly, the S^* matrix of the basic algorithm corresponds with the T matrix of the materials assembly system. The cells s^*_{ij} of S^* may be defined as the quantity of i which is accumulated in the processing per unit of j.** It is also possible to define a consumption relation and corresponding matrix C, where the cells c_{ij} represent the total quantity of element i "consumed" in the preparation of element j, either

* The reader is referred to the summary of Vazsonyi's and Giffler's work presented in Chapter II in comparing the S, S*, and C matrices to the N, T, and C matrices, and to the development in general.

** It will be shown in Chapter V that the definition of the S* and C matrices in this chapter is inaccurate in an information processing system, and that observation of this inaccuracy is the basis for refinement of the technique. However, these definitions must be assumed in the basic algorithm in order to show correspondence to the techniques of Homer and Lieberman.

directly or indirectly. This matrix corresponds directly to the C matrix of Giffler's development.

In justification of the C matrix, Lieberman's basic notion can be examined.* His concept was to form a series of matrices, one for each level of the system under study. The first of this series, matrix M_s, represents as rows the items of data and as columns the source documents upon which they appear. The second matrix, M_1, showed the source documents as rows, the first level of reports as columns. Subsequent matrices M_2, M_3, . . . , M_B, used the column designation of the preceding matrix as row designation, and the reports to be produced at that level as column designations. Lieberman's technique then consists of successive multiplication of the matrices to obtain a solution matrix,

$$M = ((\ldots((M_s M_1)M_2)M_3)\ldots)M_B) \qquad (1)$$

Using this Lieberman technique as a point of departure, let there be defined a series of matrices to be multiplied together as above in order to obtain a final system matrix M. However, let the cells of the matrices be any arbitrary integer quantity, rather than being confined to zero or one as in the Lieberman model.

Assuming a three-level system, the matrices E, F, and G, shown in Table 9, can be defined, with the small letters in the cells denoting any arbitrary integers, and the numerals denoting system elements as rows and columns also being arbitrarily assigned. It would now be an

* See the summary of Lieberman's work in Chapter II.

easy matter to obtain the product matrix M for the three matrices, where M = (AB)C = A(BC) = ABC because of associativity.

Leaving Lieberman's model, let now the same data be displayed in a single S matrix, according to the prior definition of S. Also, let S be partitioned as shown in Table 10.

Table 9. Matrices E, F, and G

$$E = \begin{array}{c|ccc} & 3 & 4 & 5 \\ \hline 1 & a & b & c \\ 2 & d & e & f \end{array}$$

$$F = \begin{array}{c|ccc} & 6 & 7 & 8 \\ \hline 3 & g & h & i \\ 4 & j & k & \ell \\ 5 & m & n & o \end{array}$$

$$G = \begin{array}{c|ccc} & 9 & 10 & 11 \\ \hline 6 & p & q & r \\ 7 & s & t & u \\ 8 & v & w & x \end{array}$$

Table 10. S Matrix (Partitioned)

	11	10	9	8	7	6	5	4	3	2	1
11	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
8	x	w	v	0	0	0	0	0	0	0	0
7	u	t	s	0	0	0	0	0	0	0	0
6	r	q	p	0	0	0	0	0	0	0	0
5	0	0	0	o	n	m	0	0	0	0	0
4	0	0	0	ℓ	k	j	0	0	0	0	0
3	0	0	0	i	h	g	0	0	0	0	0
2	0	0	0	0	0	0	f	e	d	0	0
1	0	0	0	0	0	0	c	b	a	0	0

It can be immediately recognized that the partitioning of S in Table 10 results in a 4 x 4 partitioned matrix S_p, where the original E, F, and G matrices are contained within the partitioned matrix. This can be displayed as:

$$S_p = \left\{ \begin{matrix} 0 & 0 & 0 & 0 \\ G & 0 & 0 & 0 \\ 0 & F & 0 & 0 \\ 0 & 0 & E & 0 \end{matrix} \right\} \tag{2}$$

Now, by the ordinary rules of matrix multiplication and addition, the following matrices can be obtained:

$$S_p^2 = \begin{Bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ FG & 0 & 0 & 0 \\ 0 & EF & 0 & 0 \end{Bmatrix} \qquad (3)$$

$$S_p^3 = \begin{Bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ EFG & 0 & 0 & 0 \end{Bmatrix} \qquad (4)$$

$$S_p^4 = \quad (0) \quad (\text{Null}) \qquad (5)$$

$$S_p + S_p^2 + S_p^3 = \begin{Bmatrix} 0 & 0 & 0 & 0 \\ G & 0 & 0 & 0 \\ FG & F & 0 & 0 \\ EFG & EF & E & 0 \end{Bmatrix} \qquad (6)$$

It is now noted that the product EFG found in S_p^3 is precisely the objective of the multiplication in the Lieberman model, and is therefore also the solution space in Homer's model. In addition, the products EF and FG in S_p^2 are seen to be the two-level relationships of the system, while the single level relationships are preserved in the S_p matrix. If the indicated multiplications of E, F, and G were performed, and the results inserted in the right-hand side of Equation 6, the result would then meet precisely the definition given for the C matrix.

Therefore, by extension, the relationship

$$C = S + S^2 + S^3 + . . . + S^m \tag{7}$$

must hold, where m is the number of levels in the system under study.*

The definition of the matrix S^* now permits the observation that:

$$s^*_{ij} = \begin{vmatrix} c_{ij}, & \text{if } i \neq j \\ 1 + c_{ij}, & \text{if } i = j \end{vmatrix} \tag{8}$$

from which

$$\begin{aligned} S^* &= I + C \\ S^* &= I + S + S^2 + . . . + S^m \\ (I - S)S^* &= (I - S)(I + S + S^2 + . . . + S^m) \end{aligned} \tag{9}$$

and continuing by steps identical to those in the material systems case, the conclusion may be reached that

$$S^* = (I - S)^{-1} \tag{10}$$

* The foregoing argument helps to clarify the statement made by Giffler relative to the C and N matrices, Equation 9, Chapter II.

if the inverse exists. Since S is strictly triangular, (I - S) must be triangular. In a triangular matrix, the determinant is equal to the product of the diagonal elements, therefore, since the determinant of (I - S) = 1, the matrix is non-singular, and $(I - S)^{-1}$ exists.

In summary, it is seen that the solution of Lieberman and therefore of Homer is imbedded within the more general solution of the $S - S^*$ matrix relationship. Furthermore, the S, C, and S^* matrices represent the information system in the same sense as the series of matrices of Lieberman, and the two modifications proposed by Homer. That the representation is not entirely adequate has already been commented upon, and later chapters will develop a more adequate representation.

Computational Considerations

It was previously noted (Chapter II, Equations (15) and (16)) that Giffler had developed iterative equations for the computation of the T matrix, given N. With appropriate changes of notation, these equations could form the basis for a computational algorithm for the computation of the S^* matrix. These equations, however, would require the storage in computer memory of the entire S matrix prior to the start of computation and would offer little, if any, possibility for data compression in memory. In any matrix operation using a digital computer, the problem of exceeding available memory capacity exists, and in this case will act as an upper bound on the number of systems elements which can be accommodated in the computation. Therefore, in order to permit overlap of input and computation, and to provide a basis for compression of data in computer memory, a modified algorithm has been developed which permits computation without storing the S matrix, which permits both compression of data and discarding of data as the computation progresses, and furthermore, will accommodate within its structure the more complex relationships to be developed in connection with the refined algorithms.

In the development to follow, it is assumed that S and S^* are lower triangular as a matter of expository convenience. With suitable changes, these matrices could be assumed to be upper triangular, with no loss of validity.

If S and S^* matrices are examined it is noted that the diagonal of S^* consists of cells each with the value of 1, placed there by the definition of S^* and the operation of Equation 8. Several of the columns on the right hand end of S^* have the diagonal 1 as the only non-zero entry in the column, and there are several other such columns scattered through the matrix. This condition will always exist, since these columns represent input elements.

It is now noted that every column, except for the diagonal 1, is a linear combination of the columns to the right of it, with the coefficients of the combination being specified by the corresponding column of the S matrix.[*]

It is also apparent that a given cell of the S matrix is required in the computation only once. Furthermore, the result would be the same, for a given column, if the order of multiplication and addition of col-

[*] It is for this reason that Homer was able to obtain his S^* matrix by a series of column operations.

umns were permuted, since the order of linear combination does not affect the value of the column vector.

This immediately suggests that a computer program could be written which would make use of the relationships shown above, by starting computation on the right hand end of the S^* matrix, and reading s_{ij} values from the bottom of the column up. This would also permit reading the s_{ij} from input, rather than storing them, since they will be used only once, in a systematic manner.

Based on the above, the following expression may be written, by means of which the jth column of S^* may be computed.

$$S_j^* = \sum_{k=j+1}^{n} s_{kj} S_k^* \quad \text{if } j < i \tag{11}$$

$$s_{ij}^* = 1 \qquad \text{if } j = i \tag{12}$$

$$s_{ij}^* = 0 \qquad \text{if } j > i \tag{13}$$

In these expressions, S_j^* is the jth column vector of S^* for cells below the diagonal, S_k^* is the kth column vector of S^* for $k > j$, and n is the total number of systems elements in the information processing system. It is assumed that computation will start with the rightmost column of S^* and proceed column by column to the left.

The Basic Algorithm

The basic algorithm for formation and solution of a system matrix is as follows: (The notation is the same as Homer's, for comparative purposes.)

1. Form the system matrix S according to the following rules:
   a. Denote each item of information as d_i, each report or document by R_k, and each business function by B_r. The report level need not be specified. (In the example which follows they have been specified in order to conform to Lieberman's technique and Homer's notation for comparative purposes.)
   b. Establish both a row and a column for each B_r, R_k, and d_i. The sequence of the rows and columns must be such that the business functions, B_k, are listed first, followed by the R_k reports, and finally the input data d_i. Within the R_k, the rule must be followed that no report precedes a report in which the first report is needed for the preparation of the second. The order of rows and columns must be identical.*
   c. Define each cell as 1 if a data item d_i is used in a given report or directly by a business function, a report R_k is used in preparation of another report, or a report R_k is used by a business function B_r.

* An algorithm to be presented in conjunction with the refined model will furnish a simple means for assuring that the row and column designations of the S and S* matrices are properly ordered for triangularity.

Define all other cells as zero. The matrix so defined should be strictly lower triangular.

2. Prepare a format for the solution matrix S^*, with the same row and column designations in the same order as for the matrix S.

3. Place 1's on the diagonal of the solution matrix.

4. Start computation at the lower right hand corner cell. The computation will proceed column by column, from right to left until the left-most column has been completed.

5. Establish a column vector V, initially zeros, outside the S^* matrix. This is a working vector, and should contain as many cells as there are cells below and including the diagonal cell in the column of S^* which is currently being computed. The top cell of V corresponds to the diagonal element of the column being computed, while the bottom cell of V corresponds to the bottom cell of the column being computed. (Note: In the first iteration there is but one cell, the diagonal itself. Computation of all the columns representing data items d_i is trivial, but necessary in computer operation from a conceptual point of view.)

6. Look at the corresponding column of the S matrix, starting at the bottom of the column. If this cell is non-zero, determine the row designation of the cell in question, and find the column with the same designation. If the cell is zero, repeat step 6 for the cell next above.

7. Multiply the column so obtained by the value of the non-zero cell, and add the result to V. Repeat steps 6 and 7, working up the column, until the diagonal is reached. When the diagonal is

reached, add the vector V to the column of the S^* matrix being computed. (The only non-zero entry in this column up to this point will have been the 1 on the diagonal.) Choose the column of the S^* matrix immediately to the left, and repeat from step 5. When all columns have been computed, a triangular matrix will have been generated, with 1's on the principal diagonal, and zeros above this diagonal.

A Comparative Example

In order to illustrate the proposed technique and in order to furnish a basis for comparison with Homer's method, the data from Homer's example have been used, and are shown in Figure 6, while the resultant S^* matrix is shown in Figure 7.

It will perhaps assist in understanding the foregoing algorithm if the computation of one of the columns is shown in detail. Since the first several columns are either trivial or easy to compute, the computation of the column labeled R5(2) will be treated. Assume that all columns of the S^* matrix to the right of R5(2) have been computed and entered as shown, and that R5(2) and all columns to the left have a 1 on the diagonal as the only non-zero entry. (Columns to the left of the column being computed, in both the S and S^* matrices cannot enter into the computation.) The computation starts with step 5 of the algorithm, and a column vector V, of 13 cells, initially zero, is established. In step 6, it is seen that the bottom cell of the R5(2) column of the S matrix is zero. Therefore, step 6 is repeated, and the second from the bottom cell is found to contain a 1. The row designation of this cell is d6. The multiplication of the correspondingly labeled column by the cell value, and subsequent addition to the V vector, is shown in Table 11. Note that only column values below the diagonal are of interest, because of triangularity.

Step 6 is then repeated, finding zero values on the next six iterations. On the seventh iteration, a 1 is found, with the row designated as R3(1). The multiplication of the correspondingly labeled column by the cell value, and the addition to the previous V vector is shown in Table 12.

Table 11. Computation of Column R5(2)--First Step

Cell Value		Column d6			
(From S)		(From S^*)		Product	Vector V
					0
					0
					0
					0
					0
					0
					0
					0
					0
					0
					0
1	X	1	=	1	1
		0		0	0

	B1	B2	B3	B4	R1(2)	R2(2)	R3(2)	R4(2)	R5(2)	R6(2)	R1(1)	R2(1)	R3(1)	R4(1)	d1	d2	d3	d4	d5	d6	d7
B1																					
B2																					
B3																					
B4																					
R1(2)		1	1	1																	
R2(2)	1	1		1																	
R3(2)	1			1																	
R4(2)																					
R5(2)		1	1																		
R6(2)	1																				
R1(1)					1		1	1	1												
R2(1)	1	1																			
R3(1)					1	1		1	1												
R4(1)						1	1														
d1												1	1								
d2											1	1	1								
d3													1								
d4											1		1								
d5												1									
d6					1		1	1	1												
d7	1	1	1																		

Blank cell denotes a zero entry.
Data is same as in Homer's example (Figure 1).

Figure 6. S Matrix--Improved Technique

	B1	B2	B3	B4	R1(2)	R2(2)	R3(2)	R4(2)	R5(2)	R6(2)	R1(1)	R2(1)	R3(1)	R4(1)	d1	d2	d3	d4	d5	d6	d7
B1	1																				
B2		1																			
B3			1																		
B4				1																	
R1(2)		1			1																
R2(2)	1	1				1															
R3(2)	1						1														
R4(2)								1													
R5(2)		1	1						1												
R6(2)	1									1											
R1(1)	1	2	2	2	1		1	1	1		1										
R2(1)	1	1										1									
R3(1)	1	3	2	2	1	1		1	1				1								
R4(1)	2	1		2		1	1							1							
d1	2	4	2	2	1	1		1	1			1	1		1						
d2	3	6	4	4	2	1	1	2	2		1	1	1			1					
d3	1	3	2	2	1	1		1	1				1				1				
d4	2	5	4	4	2	1	1	2	2		1		1					1			
d5	1	1										1							1		
d6	1	2	2	2	1		1	1	1											1	
d7	1	1	1																		1

Blank cell denotes a zero entry.
Cells in Homer's "Solution Spaces" are outlined.

Figure 7. S^* Matrix--Improved Technique

Table 12. Computation of Column R5(2)--Second Step

Cell Value		Column R3(1)		Product	Vector V
					0
					0
					0
					0
1	X	1	=	1	1
		0		0	0
		1		1	1
		1		1	1
		1		1	1
		1		1	1
		0		0	0
		0		0	1
		0		0	0

Step 6 is again repeated, and a zero cell encountered. The next repetition of step 6 finds a 1 with a row designation of R1(1). Table 13 details the computation for this cell.

Table 13. Computation of Column R5(2)--Third Step

Cell Value		Column R1(1)		Product	Vector V
					0
					0
1	X	1	=	1	1
		0		0	0
		0		0	1
		0		0	0
		0		0	1
		1		1	2
		0		0	1
		1		1	2
		0		0	0
		0		0	1
		0		0	0

Another repetition of step 6 again finds a zero value in the S matrix, while yet another encounters the diagonal cell. At this point the vector V is added to the R5(2) column of the S^* matrix, which was previously zero except for the diagonal element. This gives the result shown in this column in Figure 7.

Discussion

The cells in Figure 7 corresponding to the cells in the "solution space" of Homer's S^* matrix in Figure 2 have been outlined. A comparison of the two figures shows that the cell values of corresponding cells are identical in value, although the arrangement of cells is somewhat different. It is thus seen that the proposed technique reproduces Homer's basic results, as was also shown in the mathematical development.

In addition, it will be noted that the S^* matrix of the proposed technique preserves all original and intermediate information system relationships. For example, several of the reports are intermediate reports, and their data content was lost in the Homer model unless auxilliary computations were made. The presented technique clearly defines the composition of all of these intermediate reports, which might in a real system be important working documents. Furthermore, if redundancy does indeed exist, a basis is furnished for tracing this redundancy through the information system.

In summary, it appears that this model is an improvement to the existing models, and resolves Homer's objections to the Lieberman model. Except for benefits one and four, the benefits of Homer's model are met or exceeded. As will be seen in subsequent chapters, a triangularization procedure will be presented which would, in conjunction with this basic model, also result in meeting benefits one and four.

Further discussion of the interpretation and manipulation of the S and S^* matrices, and their use in systems design could be made at this time. However, both the Homer model and the one just presented have serious limitations in practice, due to failure to consider certain differences in data types and in presenting spurious indications of redundancy. These limitations have been discussed in Chapter III and techniques will be developed to eliminate or minimize them in later chapters.

CHAPTER V

THE REFINED MODELS--MATHEMATICAL DEVELOPMENT

Introduction

This chapter will treat the mathematical considerations pertaining to the refined models. Since efficient computer manipulation of the refined models depends on the triangular nature of the S and S' matrices to be defined, the first portion of the chapter will deal with the logic of triangularization. Following this portion, a mathematical treatment of the refinements will be developed, wherein the several objections inherent to the basic model will be resolved.

Notation

The three categories of systems relations defined and discussed in Chapter III cause very different action in the refined algorithms. It is therefore necessary to define the notation to be used, and the rules of computation within and between the three categories of relation prior to pursuing the mathematical development.

The three categories of relation will carry notation and quantification as follows:

1. Prime relations will be denoted by a pair (i, j) where j is the identification of the element which requires the ith type of element in its preparation. The number of i elements required will be denoted by a non-negative integer, k.*

For example, the designation D1 W1 1 denotes a prime relation where element W1 requires one of element D1 in its preparation. Al input elements will result in an arbitrary prime relation, and an arbitrary designation pair (0,j) 1. As will be seen, this designation is necessary in order that triangularization of the S matrix may be accomplished.**

2. Deletion relations will be denoted by a pair (i,j) where j is the identification of the element from which the ith element is to be deleted. The deletion itself is denoted by the negative integer -1. The character 1 is symbolic, rather than numeric, and any negative integer will have the same effect of deleting all reference to the ith element from the jth element, and from higher level elements which in turn use the jth element.

For example, the designation C1 R1 -1 denotes a deletion relation where reference to element C1 is to be deleted from element R1 and from all subsequent elements which use R1.

For brevity, the terminology "pairs (i,j)" or simply "pairs" will be used in future discussion to refer to the above notations when such use is not confusing.

3. Concomitant relations will be denoted by an n-tuple (i,j, k,...,n) of element identifications where j is the identification of the

---

* In practical operation, if no relation exists, a zero will be assumed.

** The designations used as examples, may be seen in graphic form in Figure 9, Chapter VI.

compound element wherein the result of the computation or transition will reside, i is the identification of the simple element being computed, and k,...,n are identifications of the factors (simple elements) entering the computation or transition. The relation is quantified by the symbol (c), where c is an integer. The symbol (c) has special multiplication-like properties in the operation of the algorithms.

As an example of the notation of concomitant relations, the designation C1 W3 D3 S1 (1) denotes the fact that simple elements D3 and S1 are used in the computation of simple element C1, this computation arising in the preparation of compound element W3, and that simple element C1 appears in compound element W3. The designation C4 R2 C2 (1) indicates that simple element C2 makes a transition to become simple element C4, this transition arising in the preparation of compound element R2, and C4 appears in R2.

Note that a concomitant relation has at least three, and up to (finite) n, element designations. In the formation of the S matrix, the quantifier (c) pertains to the relation (i,j) and to the relations (i,k), (i,ℓ), ..., (i,n), but does not pertain to the relations between j and subsequent indices nor k and subsequent indices.

In the computation of the algorithms to be presented, the symbol (c) has special properties which are defined as follows, where (c) is the quantifier for a concomitant relation, $\star$ is the operator in the multiplication-like operation being defined, k is a non-zero integer, and 0 is an ordinary numeric zero:

$$(c) \times k = (ck) = k \times (c)$$

$$(c) \times 0 = 0 = 0 \times (c)$$

$$(c_1) \times (c_2) = 0$$

The symbol (c) behaves as in ordinary addition for addition with zero; that is, $(c) + 0 = 0 + (c) = (c)$. It will later be shown that addition of (c) with values other than zero never arises.

Triangularization

Triangularization is an essential part of the computational technique leading to the solution matrices of the refined models. In order to obviate tedious labor on the part of the systems analyst, it is desirable that this triangularization be done by mechanical means, and that the computational system accept designations of systems elements which are entirely arbitrary. It is necessary that the designations of the systems elements be consistent and unique. That is, each system element must be designated by one and only one set of characters, and different elements must not be given the same designation.

The arrangement of a matrix which is triangular in nature into triangular form is essentially a matter of interchanging rows and columns until the desired arrangement is reached, and is therefore a logical rather than a mathematical problem. There are many ways of accomplishing this task, most of which are not adaptable for machine operation. Giffler (9, p. 18-34) describes one such method for triangularization. This method, however, has several limitations which will be discussed in Chapter VII. An algorithm will also be presented in Chapter VI which is readily adaptable to mechanization, and which overcomes the objections to the Giffler method.

The essential basis for the new algorithm rests on the definition of triangularity derived from observation of the triangularized matrix. Restated in terms of an information processing system and of lower triangularity, the S matrix (and therefore the S^* matrix) will be lower triangular if the row and column indices of S are both assigned in the same order, and in such a manner that the index of a given system element precedes the indices of every system element which is required in its preparation.

In order to take arbitrarily identified and arbitrarily ordered pairs of elements between which some relationship exists and place these pairs in a relationship such that the preceding definition holds, the following can be noted:

1. In any information system, there are two subsets of elements which are unique. One of these subsets is the subset of elements comprising the input elements, and which is unique in the fact that no other elements are required in their preparation. The other is the subset of elements comprising the output from the system, which is unique in the fact that these elements are never required in the preparation of other system elements.

2. If one or the other of these two subsets of elements can be identified, it can be removed from the set of all elements. Assume that the output subset is identified, and that all elements of the output subset are removed from the system. There will then exist another subset of systems elements which were formerly required only for the

preparation of the output subset, and which now has the same property as the removed subset; that is, that these elements are not required in the preparation of other (reduced) system elements.

3. If this subset can be identified and removed, then the argument of 2, above, can be repeated for the further reduced system. The argument can indeed be repeated, removing successively lower levels of subsets, until the lowest level (input) only remains, and this level has already been shown to be unique. The same argument would apply if the input subset were assumed to be the first identified and removed. In this case, successively higher levels of subsets would be removed, until only the output subset remained.

Consider now the set of all pairs (i,j) where j is the arbitrary designation of a system element which requires the ith system element in its preparation, and where each input element causes the formation of a pair (0,j) reflecting the fact that no element is required for the preparation of an input element. Clearly, the input subset can be identified by recognition of the i = 0 in the pair (i,j). The output subset can be recognized as the subset of pairs (i,j) which have an identification j which does not appear in any pair (i,j) as an i index. Now assume that the output subset is removed from the set, leaving a reduced set. The removal of the pairs (i,j) of the output subset removed, in addition to the unique j's, the identification i of all elements required for preparation of the elements j. These i identifications still exist, however, as j identification on some elements remaining in the reduced set. At least some of the remaining elements in the set were required only for preparation of the removed j elements.

These elements can now be recognized by the same method used to recognize the original output subset; that is, as a subset of pairs (i,j) which have an identification j which does not appear in any pair (i,j) as an i index. By repetition, the entire set can be exhausted, one subset at a time. This will produce the desired triangular condition.

Basis for Refinement

Having established the basis for the basic model and for triangularization, it is now pertinent to turn attention to the additional considerations leading to the refined algorithms. The primary considerations involve the recognition of the existence of three data types, the several types of transitions, and the three categories of system relations defined in Chapter III. Other substantiating information can be gathered from the section "The Mathematical Basis" in Chapter IV. In this latter, it was noted that both the S^* and C matrix definitions, while accurate for the basic algorithm in order to show correspondence to the techniques of Lieberman and Homer, were not accurate in their description of an information processing system.

Consider first the S^* matrix. It was defined, in analogy to the T matrix of the material assembly system, as the matrix wherein the cells s^*_{ij} are defined as the quantity of i which is accumulated in the processing per unit of j. The inaccuracy of this definition lies in the fact that data in a data system do not "accumulate" in the sense implied, and that therefore the analogy to the material system breaks down. Rather than "accumulate" in a physical sense, data are only reproduced, or copies, in their movement from level to level of a system.

Rather than being used up, in the sense of materials, they are in essence only borrowed, for the purpose of copying, and their basic form is not changed through the pure act of copying or transcribing from one form to another. It is this false accumulation which gives false indications of redundancy in the basic S^* matrix.

By the same argument, the C matrix is also ill-defined and erroneous in its representation of the true system. This matrix, where c_{ij} represents the total quantity of element i consumed in the preparation of element j, also carries the connotation that data are physically "used up" in preparation of a higher level element, and are no longer available for use elsewhere in the system. In both of these matrices, the connotation is carried that if five reports require a given piece of data, then these data must be supplied as five separate pieces.

These two matrices also do not recognize the fact that unlike a materials system, in which all the matter which enters the system subsequently leaves the system imbedded within some finished product (neglecting waste), in a data system some of the elements are lost irretrievably during the course of processing. For example, type two data make an irreversible transition to type three data, and the type two data progress no further or are even destroyed.

Furthermore, the two matrices do not recognize the fact that elements are created within a system (e. g., a multiplication of two numbers) in which lower level system elements are used, but do not appear as such within the element created, and in which the substance of the lower level elements is completely lost. In a material assembly system, the fitting of two parts together physically results in a new element, but generally the two contributing elements are still physically present and frequently physically recognizable. Such is not always true when information system elements are combined to form new elements.

The recognition of data types and data transitions, together with the recognition of the shortcomings of the existing S^* and C matrices, furnish a basis for reformulation of the systems relations and redefinition of the matrices, so that refined algorithms can be developed.

The reformulation of the system relations takes the form of defining three categories of system relations--prime, concomitant, and deletion. These have been defined in Chapter III, and notation assigned in a prior section of this chapter. Recognition of these three categories also permits development of two separate, but similar, means of analysis, one on the basis of the composition of reports, the other on the content of reports. The composition type analysis will be developed first.

Development of Composition Type Analysis

The composition type of analysis will depend on the two categories of relations denoted as prime and concomitant. Using these two relations, the S matrix need not be redefined. The cells s_{ij} have been defined as the number of times a given system element i is used directly in the preparation of system element j. Both prime and concomitant relations meet this definition, prime in the sense of use by being transcribed, concomitant in the sense of elements being used in computation or in a transition from type two to type three data.

S^* must be redefined to include both types of system relations. The cells s^*_{ij} of S^* are now defined as the number of times the data element i is used or is available for use in the preparation of data element j. The "or is available for use" provides for indication of true redundancy, while the concurrent use of both prime and concomitant relations prevents indication of false redundancy, as will be shown later. The unit diagonal cells are not so explicitly contained in this definition, but should be construed to be included, if only by the plausible argument that some form is required upon which to record the contributing elements.

The C matrix, used only in the development, must also be redefined. Due to its identity to the S^* matrix, except for the unit diagonal elements, it should be similarly defined. The cells c_{ij} of C are now defined as the number of times the data element i is used or is available for use in the preparation of data element j, either directly or indirectly.

The essential difference between the C and S^* matrix, other than the evident one that S^* has unit cells on the diagonal, is that the elements of C contain the information desired in the analysis, but is difficult to compute, while S^* contains the same information, but because of the diagonal 1's is easy to compute. Conversely, it is difficult to develop the connection between the S and S^* matrix without using the concept of the C matrix. Therefore, the C matrix, with its similarity to the S^* matrix is convenient to use in developing the mathematical background, but can be dispensed with in numerical computation.

The development follows the development of the basic model to some extent, except that the matter is more complex, due to the introduction of the concomitant relation and its quantifier (c). As mentioned in a previous section, this quantifier leads to some multiplication-like operations. These operations were developed in consideration of the method to be used in computing the S^* matrix and may be justified on that basis. If the symbol (c) appears to the left of the operator $\times$, it will be used in computation as a scalar multiplying a vector on the right of the operator. If it appears on the right of the operator, it will be in the role of a cell in a vector.

Properties of Matrices

In the proof of the basic model, the common property of associativity of matrices was assumed in order to show that $C = S + S^2 + \ldots + S^m$. In order to follow a similar proof for the refined model, it is now desirable to examine the system of matrices which might be formed representing the relationships between two adjacent system levels. These matrices are of the Lieberman-Homer type, except that both regular scalars and concomitant relation quantifiers are present in the matrix. These matrices are analogous to the matrices E, F, and G in the proof of the basic algorithm.

In consideration of the definition of prime and concomitant relations, several observations can immediately be made relative to the structure of these matrices, and may be stated as axioms. Prior to stating these axioms, however, some terms will be defined in order to shorten the discussion to follow.

Definitions.

Concomitant index is a row or column index in a matrix wherein the row or column index is the identification of a system element which is formed as a result of a concomitant relation.

Regular index is a row or column index in a matrix wherein the row or column index is the identification of a system element which is formed as a result of a prime relation.

A concomitant cell in a matrix is a cell wherein the value is a concomitant relation quantifier, (c).

A regular cell in a matrix is a cell wherein the value is a regular scalar, n.

System Matrix Axioms.

Axiom 1. The lowest level system matrix will never have a concomitant index as a row index, since all row indices refer to input elements which have been defined to have a prime relation.

Axiom 2. A given concomitant index will appear first as a column index in the matrix representing the level at which the element is formed, and as a row index in the matrix representing the next higher system level. This follows directly from the definition and notation for concomitant relation.

Axiom 3. A concomitant index may appear as both a row and column index in the case where an element created by a concomitant relation is subsequently used to create another concomitant relation.

Axiom 4. If a concomitant index is a column index, then the cells of that column will have only the values zero or (c); that is, concomitant cells, because the entire column is defined by the i, k, ..., n indices of the concomitant n-tuple (i, j, k, ..., n) with a single quantifier.

Axiom 5. If a concomitant index is a row index, then the cells of that row will be concomitant cells, with value zero or (c). In this case the row is defined either by the i,j indices of a concomitant n-tuple, or by the creation of a new concomitant element using a previously created concomitant element.

Matrix Multiplication. Consider two matrices, A and B, conformable for multiplication in the usual sense. Under the usual rules for multiplication of matrices, the element in the (i,k) cell of AB = (d_{ik}) is:

$$d_{ik} = \sum_j a_{ij} b_{jk} \qquad (1)$$

In matrix multiplication where the matrix consists of both regular and concomitant cells, three cases can be distinguished. Since in all cases the index j may be either a regular or concomitant index, the set of indices j may be divided into two subsets:

$$j_1 = [j\text{: } j \text{ is a regular index}],$$

$$j_2 = [j\text{: } j \text{ is a concomitant index}].$$

Then the matrix multiplication may be written as:

$$d_{ik} = \sum_{j\varepsilon j_1} a_{ij}b_{jk} + \sum_{j\varepsilon j_2} a_{ij}b_{jk} \tag{2}$$

Case 1. In this case, both i and k are regular indices. Then:

$$d_{ik} = \sum_{j} a_{ij}b_{jk} = \sum_{j\varepsilon j_1} a_{ij}b_{jk} + \sum_{j\varepsilon j_2} a_{ij}b_{jk} = n \text{ (or zero)}. \tag{3}$$

The summation over $j\varepsilon j_1$ in this case consists of multiplication of regular cells. The summation over $j\varepsilon j_2 = 0$, since by Axioms 4 and 5, both a_{ij} and b_{jk} are either zero or (c), and by prior definition, 0 $*$ (c) = 0 = (c) $*$ 0, and $(c_1) * (c_2) = 0$. Therefore, d_{ik} is either zero or a regular cell, n.

Case 2A. In this case, i is a regular index and k is a concomitant index. Then:

$$d_{ik} = \sum_{j\varepsilon j_1} a_{ij}b_{jk} + \sum_{j\varepsilon j_2} a_{ij}b_{jk} = (c) \text{ (or zero)} \tag{4}$$

In summation over $j\varepsilon j_1$, a_{ij} is a regular cell, b_{jk} is a concomitant cell or zero by Axiom 4. Therefore the result of this portion of the summation is a concomitant cell (c). In summation over $j\varepsilon j_2$, a_{ij} is either zero or (c) by Axiom 4, and b_{jk} is either zero or (c) by Axiom 3 or 4. This summation is therefore zero, and d_{ik} is either zero or a concomitant cell (c).

Case 2B. If i is a concomitant index and k is a regular index, then

$$d_{ik} = \sum_{j\varepsilon j_1} a_{ij}b_{jk} + \sum_{j\varepsilon j_2} a_{ij}b_{jk} = (c) \text{ (or zero)} \tag{5}$$

In summation over $j\varepsilon j_1$, a_{ij} is either zero or (c) by Axiom 4 and b_{jk} is a regular cell. As in Case 2A, the result is a concomitant cell (c) or zero. The summation $j\varepsilon j_2 = 0$, as in prior cases, since both a_{ij} and b_{jk} must be either zero or (c). Therefore, d_{ik} is either zero or a concomitant cell (c).

Case 3. Both i and k are concomitant indices, and

$$d_{ik} = \sum_{j\varepsilon j_1} a_{ij}b_{jk} + \sum_{j\varepsilon j_2} a_{ij}b_{jk} = 0 \tag{6}$$

In this case, the summation over $j\varepsilon j_1 = 0$ and the summation over $j\varepsilon j_2 = 0$, since all a_{ij} and b_{jk} are either zero or (c). Therefore, $d_{ik} = 0$.

Based on the above cases, the following Lemmas can be stated:

<u>Lemma 1</u>. In matrix multiplication where both concomitant and regular indices and cells are present, if indices i and k are both regular indices, then the product cell d_{ik} is a regular cell, n, or zero.

<u>Lemma 2</u>. If either i or k is a concomitant index, and the other index is a regular index, then d_{ik} is a concomitant cell, (c), or zero.

<u>Lemma 3.</u> If both i and k are concomitant indices, then the product cell $d_{ik} = 0$.

<u>Lemma 4.</u> $\sum_{j \varepsilon j_2} a_{ij} b_{jk} = 0$, regardless of the i and k indices.

<u>Lemma 5.</u> The addition (c) + n, for $n \neq 0$, never occurs in the matrix multiplication, so it is therefore unnecessary to define addition involving concomitant relation quantifiers other than (c) + 0 = 0 + (c) = (c).

<u>Theorem 1. (Associativity)</u> In series of matrix multiplications involving matrices with both regular and concomitant indices and cells, multiplication is associative.

The proof is as follows. Consider three matrices P, Q, and R, conformable for multiplication in the usual sense, and composed of both regular and concomitant cells, and indexed by both regular and concomitant indices. From the immediately prior results, the element in the (i,k) place in PQ, d_{ik}, is

$$d_{ik} = \sum_{j \varepsilon j_1} p_{ij} q_{jk} \tag{7}$$

and the element in the (i,ℓ) place in (PQ)R is

$$b_{i\ell} = \sum_{k \varepsilon k_1} d_{ik} r_{k\ell} = \sum_{k \varepsilon k_1} \sum_{j \varepsilon j_1} p_{ij} q_{jk} r_{k\ell} \tag{8}$$

where summations over $j \varepsilon j_2$ and $k \varepsilon k_2$ are omitted because of Lemma 4.

The (i,ℓ) place in $P(QR)$, $b'_{i\ell}$, may be similarly computed. The (j,ℓ) place of QR has the element $c_{j\ell}$,

$$c_{j\ell} = \sum_{k\varepsilon k_1} q_{jk} r_{k\ell}, \tag{9}$$

and the (i,ℓ) place of $P(QR)$ is

$$b'_{i\ell} = \sum_{j\varepsilon j_1} p_{ij} c_{j\ell} = \sum_{j\varepsilon j_1} \sum_{k\varepsilon k_1} p_{ij} q_{jk} r_{k\ell} \; . \tag{10}$$

In view of the mixing of regular and concomitant indices and cells, it is now pertinent to show that the (i,ℓ) place of PQR is identical regardless of the order of multiplication and summation. There are 16 combinations of the four indices i, j, k, and ℓ. These combinations are shown in Figure 8, together with the d_{ik}, $c_{j\ell}$, $\sum_{k\varepsilon k_1} d_{ik} r_{k\ell}$, and $\sum_{j\varepsilon j_1} p_{ij} c_{j\ell}$ established through the application of Lemmas 1 through 5. It is seen that in all 16 combinations, identical results are obtained regardless of order of summation, so that the conclusion may be reached that $d_{i\ell} = d'_{i\ell}$.

In combination 1, all indices are regular, and the resulting cell is regular or zero by Lemma 1.

In combination 2, i is regular, k is regular, and $j\varepsilon j_1$, so d_{ik} is a regular cell or zero. However, ℓ is a concomitant index, so $b_{i\ell}$ is concomitant or zero by Lemma 2. Now, j is regular, ℓ is concomitant, and k is regular, so that $c_{j\ell}$ is a concomitant cell or zero

Index Combination	Index				d_{ik}	$c_{j\ell}$	$b_{i\ell}$	$b'_{i\ell}$
	i	j	k	ℓ				
1	R	R	R	R	n	n	n	n
2	R	R	R	C	n	(c)	(c)	(c)
3	R	R	C	R	(c)	0	0	0
4	R	C	R	R	0	(c)	0	0
5	C	R	R	R	(c)	n	(c)	(c)
6	R	R	C	C	(c)	0	0	0
7	R	C	R	C	0	0	0	0
8	C	R	R	C	(c)	(c)	0	0
9	R	C	C	R	0	0	0	0
10	C	R	C	R	0	0	0	0
11	C	C	R	R	0	(c)	0	0
12	C	C	C	R	0	0	0	0
13	C	C	R	C	0	0	0	0
14	C	R	C	C	0	0	0	0
15	R	C	C	C	0	0	0	0
16	C	C	C	C	0	0	0	0

R = Regular Index. n = Regular Cell.

C = Concomitant Index. (c) = Concomitant Cell.

$b_{i\ell} = \sum_{k\epsilon k_1} d_{ik}r_{k\ell}$. $b'_{i\ell} = \sum_{j\epsilon j_1} p_{ij}c_{j\ell}$.

Figure 8. Index Combinations and Resultant Cell Values in Successive Matrix Multiplication.

by Lemma 2. But again, since i is regular, Lemma 2 indicates that $b'_{i\ell}$ is a concomitant cell or zero. Combination 5 results in a concomitant or zero cell by a similar argument.

All other combinations result in $b_{i\ell} = b'_{i\ell} = 0$. In the combinations 7, 9, 10, 12, 13, 14, 15, and 16, both d_{ik} and $c_{j\ell}$ are zero by either Lemma 3 or Lemma 4. In combination 3, $c_{j\ell} = 0$ by Lemma 4, so $b'_{i\ell} = 0$. The cell d_{ik} is concomitant by Lemma 2, then $b_{i\ell} = 0$ by Lemma 4. A similar argument holds for combinations 4, 6, and 11. In combination 8 both $b_{i\ell}$ and $b'_{i\ell}$ and zero by Lemma 3, as both i and ℓ are concomitant indices. This Lemma would also apply to combinations 10, 13, 14, and 16.

It will also be useful to prove the following theorem:

Theorem 2. (Existence of Identity) IA = A = AI, where A is a matrix with both regular and concomitant cells, and I is an identity matrix of proper size and in the usual sense.

In proof, let $a'_{j\ell}$ be the (j,ℓ) cell of IA. Then

$$a'_{j\ell} = \sum_k \delta_{jk} a_{k\ell} , \qquad (11)$$

where δ_{jk} is the Kronecker delta. Then since $1 \times (c) = (c)$ by definition, it is evident that both regular and concomitant cells are preserved under this multiplication, and IA = A. In the same manner, if a''_{km} is the (k,m) cell of AI,

$$a''_{km} = \sum_\ell a_{k\ell}\, \delta_{\ell m} , \qquad (12)$$

and AI = A.

Development of the "Composed of" Model

As in the case of the basic algorithm, the system relations expressed in the S matrix may also be expressed in a series of matrices, where each matrix represents the relations, both prime and concomitant, which exist between two adjacent levels of the information system under study. Because of the nature of concomitant relations, and to permit the entry and exit of systems elements at any level, a technique similar to Homer's first method must be used in order to force conformability of these matrices for multiplication. The following rules will assure such conformability:

1. Assume an n-level information system, with between level matrices A, B, C, ..., N. Start with the lowest level matrix, A, and write as row indices the identification of all input elements.

2. Write as column indices of A the identification of all elements which use, on either a prime or concomitant basis, the input elements listed in the rows.

3. Enter in the cells of A the appropriate quantifier, either prime or concomitant.

4. Write as row indices for the next higher system matrix B the column indices of matrix A.

5. Write as column indices of B the identification of all elements which use, on either a prime or concomitant basis, the elements listed in the rows.

6. Enter in the cells of B the appropriate quantifier, either prime or concomitant.

7. Continue in this manner for matrices C, D, ..., N. The column indices of N will all be the identification of output reports, but some output reports may not be present due to leaving the system at a lower level. Some matrices will show a given identification as both a row and a column index, due to higher level elements being composed of elements from more than one lower level.

Consider now a three level system, with matrices P, Q, and R, defined and established as above. The letters in the cells of the matrices in Table 14 are arbitrary quantifiers, either prime or concomitant, subject to the definition of these relations, and to the previously shown Lemmas relative to formation of matrices consisting of both prime and concomitant elements.

Let the matrices P, Q, and R be as shown in Table 14. Note that the matrix R has element 8 as both a row and column index, which is interpreted as the fact that elements enter element 8 at two levels. The cell R_{88} is of course empty.

In a manner corresponding direct to the argument advanced in developing the basic algorithm, these three matrices might now be multiplied together in the manner of the Lieberman model, and the products PQ, QR, and PQR would contain the information relative to indirect system relations.

Also in a manner corresponding to the development of the basic algorithm, let the same data be displayed in a single matrix S_a, where S_a differs from S only in that element 8 appears twice as both a row and column index, but without duplication of non-zero information. Also, let S_a be partitioned into the 4 x 4 matrix shown in Table 15.

Table 14. Matrices P, Q, and R

$$\begin{array}{c|cc} & 4 & 5 \\ \hline 1 & a & b \\ 2 & c & d \\ 3 & e & f \end{array}$$

$$\begin{array}{c|cccc} & 6 & 7 & 8 & 9 \\ \hline 4 & g & h & i & j \\ 5 & k & \ell & m & n \end{array}$$

$$\begin{array}{c|cc} & 8 & 10 \\ \hline 6 & o & p \\ 7 & q & r \\ 8 & - & s \\ 9 & t & u \end{array}$$

This partitioned matrix can be represented by the following matrix, where the original matrices P, Q, and R are contained within the partitioned S_a matrix:

$$s_a = \left\{ \begin{array}{cccc} 0 & 0 & 0 & 0 \\ R & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & P & 0 \end{array} \right\} \qquad (13)$$

Table 15. The S_a Matrix

$$S_a = \begin{array}{c|cc|cccc|cc|ccc|} & 8 & 10 & 6 & 7 & 8 & 9 & 4 & 5 & 1 & 2 & 3 \\ \hline 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 6 & o & p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & q & r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & t & u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 4 & 0 & 0 & g & h & i & j & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & k & \ell & m & n & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & a & b & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & c & d & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & e & f & 0 & 0 & 0 \\ \hline \end{array}$$

Now, since Theorem 1 assures associativity of matrix multiplication, and since addition of concomitant relation quantifiers with zero is defined, the following matrices can be obtained:

$$S_a^2 = \begin{Bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ QR & 0 & 0 & 0 \\ 0 & PQ & 0 & 0 \end{Bmatrix} \tag{14}$$

$$S_a^3 = \begin{Bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ PQR & 0 & 0 & 0 \end{Bmatrix} \qquad (15)$$

S_a^4 is null.

$$S_a + S_a^2 + S_a^3 = \begin{Bmatrix} 0 & 0 & 0 & 0 \\ R & 0 & 0 & 0 \\ QR & Q & 0 & 0 \\ PQR & PQ & 0 & 0 \end{Bmatrix} \qquad (16)$$

It is now noted that in S_a^2 the intermediate level products PQ and QR are found, and that in S_a^3 the product PQR is found. The summation of the three matrices, when expanded to the original dimension of S_a, meets the definition of C_a, where C_a is the same as C except for the fact that element 8 appears twice as a row and a column index. Since this is purely a matter of notation, S_a is equivalent to S, and C_a is equivalent to C. C_a can be converted to C, while still preserving triangularity, by adding the right-most column 8 to the other column 8, then removing any duplication between the two rows with index 8, and deleting the lower of the two rows.

It has therefore been shown that

$$C_a = S_a + S_a^2 + S_a^3 + \ldots + S_a^m$$

and because of the equivalence of C_a and C, S_a and S,

$$C = S + S^2 + S^3 + \dots + S^m . \qquad (17)$$

By definition of the matrix S^*,

$$S^* = I + C \qquad (18)$$

$$S^* = I + S + S^2 + S^3 + \dots + S^m . \qquad (19)$$

Then, multiplying both sides by (I - S) and carrying out the multiplication,

$$(I - S)S^* = (I - S)(I + S + S^2 + \dots + S^m)$$

$$(I = S)S^* = I^2 - S + S - S^2 + S^2 - \dots -$$

$$S^m + S^m - S^{m+1} .$$

But S^{m+1} is null, and $I^2 = I$, so

$$(I - S)S^* = I \qquad (20)$$

and multiplying through by $(I - S)^{-1}$,

$$S^* = (I - S)^{-1}. \qquad (21)$$

That this inverse exists is shown by observing that (I - S) is triangular by definition, with determinant $|(I - S)| = 1$. (I - S) is therefore non-singular, and the inverse exists.

It has been shown then that S and S^* have the same relation when concomitant indices and quantifiers are employed as in the basic algorithm when all systems relations were presumed to be prime. The computation also proceeds in the same general manner, except that special rules pertaining to "multiplication" obtain when concomitant relation quantifiers are encountered. An example will be presented in Chapter VI.

Development of Containment Type Analysis

The containment type of analysis depends on all of the three categories of relations, defined as prime, concomitant, and deletion. The addition of the deletion relation requires a redefinition of both the S matrix and the S^* matrix. These redefined matrices will be denoted S' and S'^*, respectively.

As noted in the earlier section of this chapter entitled "Notation," the deletion relation quantifier -1 is symbolic, rather than numeric. Furthermore, the deletion relation leads to an operation in the computation of the S'^* matrix which is logical rather than arithmetic in nature. For this reason, it is not possible to offer a development of the computational procedure on a mathematical basis, but only a development on the logical basis. In fact, it can be shown, by attempting the multiplication, that

$$S'^{*}(I - S') \neq I \tag{22}$$

and that therefore the relationship between the S and S^{*} matrices shown in the previous section is not true for S' and S'^{*}.

<u>Definition of S' and S'^{*} Matrices</u>

In the definition of the S matrix, the cells s_{ij} were defined as the number of times a given system element i is used directly in the preparation of system element j, where both prime and concomitant relations met the definition. In the case of prime relations, however, where the i index of the relationship is the identification of a compound element, the compound element itself (a report, tape record, or the like) will not physically appear, or be "contained in" the compound element denoted by the j index of the prime relationship. Furthermore, some of the simple elements included within the ith compound element may neither be used nor contained in the jth element.

In order to exclude the ith compound element from appearing in the jth compound element of S'^{*}, a change in definition of the solution matrix and a change of computational method is required. The exclusion of the unused simple elements within the ith compound element leads to the deletion relation and a corresponding change in definition of the matrix S'.

The matrix S' is now defined as that matrix where the cells s'_{ij} represent the number of times a given system element i is used directly in the preparation of system element j in the case of prime and concomitant relations, <u>and</u> as the simple system element k which is contained in compound element i, but which will not physically be included in compound element j in the case of a deletion relation.

The matrix S'^{*} is defined as that matrix where the cells s'_{ij} represent the number of times the <u>simple</u> data element i which actually appears in <u>compound</u> element j is used or is available for use in preparation of data element j.

It should be noted that the definition of S' represents the inclusion of the deletion relation in the previously defined S matrix, while the definition of S'^{*} represents the blocking out of any element, simple or compound, which does not actually appear in a given compound element.

<u>Effect on Computation</u>

The effect on computation, in comparison to computation of the "Composed of" matrix S^{*}, is that of adding two logical operations to the mathematical operation of inverting the (I - S') matrix. These two logical operations prevent the realization of a true inverse, but result in the definition of S'^{*} being met in the solution.

The first logical operation is the prevention of any compound element from appearing in another compound element in S'^{*}. This is done by recognizing that any column in S'^{*} which contains a non-zero entry other than on the diagonal is compound; and that the diagonal element, which represents the compound element itself, must be ignored in the computation of an element at a higher system level.

The second logical operation is the preventing of any reference to a simple element from appearing in a compound element where a deletion relation is indicated in the S' matrix. This is done by the simple method of merely viewing the presence of a deletion relation in s'_{pq} as a block to permitting any entry, prime or concomitant, from appearing in the corresponding cell, s'^*_{pq} of S'^*. This in turn will prevent the carrying of the deleted element to any higher level compound element.

The algorithm for computation of the S'^* matrix to be presented in the next chapter will therefore bear a strong resemblance to the algorithm for computation of S^*, except for provision for carrying out the two logical operations indicated above in addition to the mathematical operations leading to what might be called a pseudo-inverse of $(I - S')$.

CHAPTER VI

THE REFINED MODELS--ALGORITHMS AND EXAMPLES

Introduction

Based on the development of Chapter V, this chapter will present algorithms by means of which the specifications of systems relations prepared by the systems analyst may be arranged in the required triangular form and the computation of the system solution matrices S^* and S'^* performed. At this time, the algorithms are presented for expository purposes as algorithms for manual computation. They were developed specifically, however, for machine computation, and Chapter VII and the Appendices contain further discussion of the implications of machine computation and restatements of the algorithms in language more appropriate for such use. In order to illustrate the triangularization and computation of the solution matrices, a very small numerical example is presented, and this example is also used as a basis for discussion of interpretation of the solution matrices.

Triangularization Algorithm

In the previous chapter, it was established that the desired triangularity of the S and S' matrices could be realized by recognizing and removing first the highest level of systems element designations from the set of all designations, then the next highest, and so on until all designations had been removed. The algorithm to be presented merely represents a systematic means of such recognition and removal. On a manual basis, a single list of the pairs (i,j) could be used rather than a double list, but for subsequent mechanized operation, a double list is desirable.

Because of the way that the triangularization algorithm operates, the method of identification of system elements is not critical, and any element can be designated by any arbitrary numerical or alphamerical designation, with the only restraint being that the identification be consistent and unique--that is, a given identification pertain to only one element, and that a given item have only one identification.

In many real systems, some sort of forms control is exercised, wherein every report is given a distinctive number. If such is the case, this number is perfectly useful for identification purposes and may be so used. Business function (if designated as "elements" for purposes of analysis) might well be designated by the section number of the organization manual which describes the function. The identification of data has usually not been accomplished uniquely and consistently prior to the start of a systems study. A useful technique in this case is merely to assign a serial number to each piece of data as it is discovered in the system, without regard to the system level at which it is encountered.

In the example in a later section, an identification which is partly mnemonic is used. This has been done purely for expository purposes, and is not recommended in practice. From the point of view of computer operation, a purely digital identification is preferred.

In the algorithm to follow, it is presumed that the elements of the system have been given the required unique and consistent designations, and that the definition of prime relations (pairs) and concomitant relations (n-tuples) has been done. Deletion relations are not a factor in triangularization, and the pairs defining deletion relations should be set aside until the formation of the S' matrix takes place.

1. Prepare two identical lists, each containing all pairs (i,j), together with the quantifier, if desired.* Designate these two lists as list A and list B. Order list A on the j index, with minor ordering within the j index on the i index. Order list B on the i index, with minor ordering within the i index on the j index.

2. Compare each j, in turn, in list A to the i in list B. Remove from list A all pairs (i,j) for which a j in list A does not have an equal i in list B.

B. Remove from list B all pairs (i,j) which were removed from list A in step 2. Record the order of removal of j indices from list A. The j indices of the pairs removed are the designations of the highest order elements of the information processing system.

4. If the lists A and B are not exhausted, repeat steps 2 and 3 on the reduced lists, which will remove successively lower level system elements. If the lists are exhausted, go to step 5.

5. When the lists are exhausted, the record of order of removal of pairs represents the order of listing of row and column elements in

---

* The quantifiers are not required in triangularization, but it is usually convenient for the quantifier to accompany the pair or n-tuple, particularly in machine versions of the algorithm.

the S and S' matrices for the prime relations. Now arrange a list of concomitant relations (n-tuples) in the same order based on the second index of the n-tuple, j', and within a given j', if an element designation appears both as an i index (first position in n-tuple) and as an index k, ℓ, ..., n, then the n-tuple containing the element as an i index follows all n-tuples where it appears as a k, ℓ, ...,n index. Then for every j in the record of removal of pairs from list A for which there is a j' = j in the list of concomitant relations, insert after j in the record of removal the first index, i, of the concomitant n-tuple.

The triangularization of the system elements is complete, and for manual determination of the S, S', S^* and S'^* matrices this condition is adequate. For determination by the use of a computer, it is necessary to re-index the elements. This technique will be presented in Appendix B, since it is not of interest in the general development of the algorithms.

Formation of S and S' Matrices

In computation by means of a computer, it is not necessary to form the S and S' matrices. It is, however, necessary in the manual computation. Furthermore, even though the two matrices are not formed in the computer solution, the S and S' matrices exist in conceptual form, in the medium of card or tape records. The formation of the matrices in conceptual form or for manual computation is governed by the following rules:

1. For both the S and S' matrix, label both rows and columns in the same order as the record of removal of pairs established in

step 3 of the triangularization algorithm, as augmented by the inclusion of indices from the n-tuples as specified in step 5. This labeling should proceed to the right of the columns and downward for the rows, starting at the upper left hand corner of the matrix.

2. For the S matrix, record all prime relations by entering the quantifying numeral in the cell denoted by the pair (i,j). Record all concomitant relations with the concomitant quantifier (c) for each cell, where the cell indices are determined by considering the first element designation i of the n-tuple as a row index, and the second element designation j as a column index; and then by considering the first element designation i as a column index and each of the remaining element designations k, ℓ, ..., n, in turn, as row indices, until the n-tuple is exhausted. Note that an n-tuple of s element designations will produce s - 1 cells in the S matrix.

3. For the S' matrix, follow steps 1 and 2, above. In addition, record all deletion relations as -1 in the matrix, using the same rules for determining the proper cell as for the prime relations.

The S—S^* Algorithm

Having formed the S matrix, the S^* solution matrix may be computed using the following algorithm:

1. Prepare a format for the solution matrix S^*, with the same row and column designations in the same order as in the S matrix. Place 1's on the diagonal of the solution matrix.

2. Start computation at the lower right hand cell. The computation will proceed column by column, from right to left until the left-

most column has been computed.

3. Establish a column vector V, initially zeros, outside the S^* matrix. This is a working vector, and should contain as many cells as there are cells below and including the diagonal cell in the column of S^* which is currently being computed. The top cell of V corresponds to the diagonal element of the column being computed, while the bottom cell of V corresponds to the bottom cell of the column being computed.

4. Examine the bottom cell of the column in S with the same column designation as the column being computed in S^*. If this cell is non-zero, determine the row index of the cell in the S matrix, and find the column of S^* with this same index. If the cell is zero, determine whether this cell is on the prime diagonal of the matrix, indicated by the fact that the row index i = the column index j. If it is the diagonal cell, go to step 6. If the cell is not the diagonal cell (and is zero), repeat step 4 for the cell in the S matrix next above this cell in the same column.

5. Multiply the non-zero value of the cell established in step 4 by the column of S^* found as a result of step 4 and add the product to V. In this multiplication the usual rules of multiplication obtain for the ordinary numerals quantifying prime relations, but when a concomitant quantifier is involved, either in the cell value or in the column vector, the relations specified in the "Notation" section, Chapter V must be used.

Choose the cell in the S matrix next above the current cell in the same column, and repeat step 4 and step 5.

6. When a diagonal cell is reached, as indicated by the operation of step 4, add the vector V to the column of the S^* matrix under consideration. Then choose the column of the S^* matrix immediately to the left of the one just computed, and repeat the algorithm from step 3. When all columns have been computed, a triangular solution matrix S^* will have been generated.

The S'—S'^* Algorithm

The S'^* matrix is computed from the S' matrix using an algorithm which is a modification of the S-S^* algorithm. Steps 1, 2, and 3 are identical, with appropriate change of notation from S to S' and S^* to S'^*. The balance of the steps are as follows:

4. Examine the bottom cell of the column in S' with the same column designation as the column being computed in S'^*. If this cell is non-zero, determine which of the relation types is represented by the quantifier in the cell. If the cell entry is a deletion relation, -1, insert an X in the corresponding cell of Vector V, then repeat step 4 for the cell in the S matrix next above this cell in the same column of S'. The purpose of this X is to prevent any further entries from any source being placed in that particular cell.

If the cell entry is the quantifier for either a prime or concomitant relation, determine the row index of the cell in the S' matrix, and find the column of S'^* with the same index. If the cell is zero, determine whether this cell is on the prime diagonal of the matrix, indicated by the fact that the row index i = the column index j. If it is the diagonal cell, go to step 6. If the cell is not the diagonal cell (and is zero), repeat step 4 for the cell in the S' matrix next above this cell in the same column.

5. If the cell is a concomitant relation quantifier, multiply the cell by the column of S'^* found as a result of step 4, observing the special rules for such multiplication previously specified. If the cell value is a regular numeral, determine whether the column of S'^* found in step 4 has any non-zero cells other than the diagonal cell. If it has not, multiply the cell value by the column vector (which consists of zeros except for the diagonal element) and add the product to the vector V. If the column has non-zero, non-diagonal values, multiply the cell value by the column vector ignoring the diagonal cell (or treating it as zero), and add the product to the vector V. In both of these multiplications, observe the previously defined rules for multiplication involving concomitant quantifiers. Also observe the presence of an X in the vector V as blocking any further entry.

Choose the cell in the S' matrix next above the current cell in the same column, and repeat step 4 and step 5.

6. When a diagonal cell is reached, as indicated by the operation of step 4, add the vector V to the column of the S'^* matrix under consideration, now treating an X in vector V as a zero. Then choose the column of the S'^* matrix immediately to the left of the one just computed, and repeat the algorithm from step 3. When all columns have been computed, a triangular solution matrix S'^* will have been generated.

A Numerical Example

In order to illustrate the operation of the refined algorithms, and to furnish a basis for discussion of the interpretation of the S^* and S'^* solution matrices, a small numerical example will be presented. In this example, the input to the system is a daily time report, listing payroll number, department number, and hours worked, recorded on a punched card. Processing, by unit record equipment, is to compute daily labor dollars by department and total, and weekly department total labor dollars, compared to the prior week. In addition to these two reports, cards containing payroll number, department number, hours worked and labor dollars are to be created for subsequent use in the payroll operation. Mnemonic designations for the several systems elements have been used for ease in discussion; as previously emphasized, element designations can be purely arbitrary for these algorithms. Also, for purely expository purposes, a flow chart of the system is shown in Figure 9.

The following list designates the systems elements:

D1 - Payroll number.

D2 - Department number.

D3 - Hours worked.

S1 - Hourly rate. (This is a constant for a given payroll number, and is stored, with the payroll number, in a master deck.)

C1 - Individual cost. (C1 = D3 x S1)

C2 - Department total cost, daily. (C2 = C1 summed over D2)

C3 - Grand total, daily. (C3 = sum of all C2)

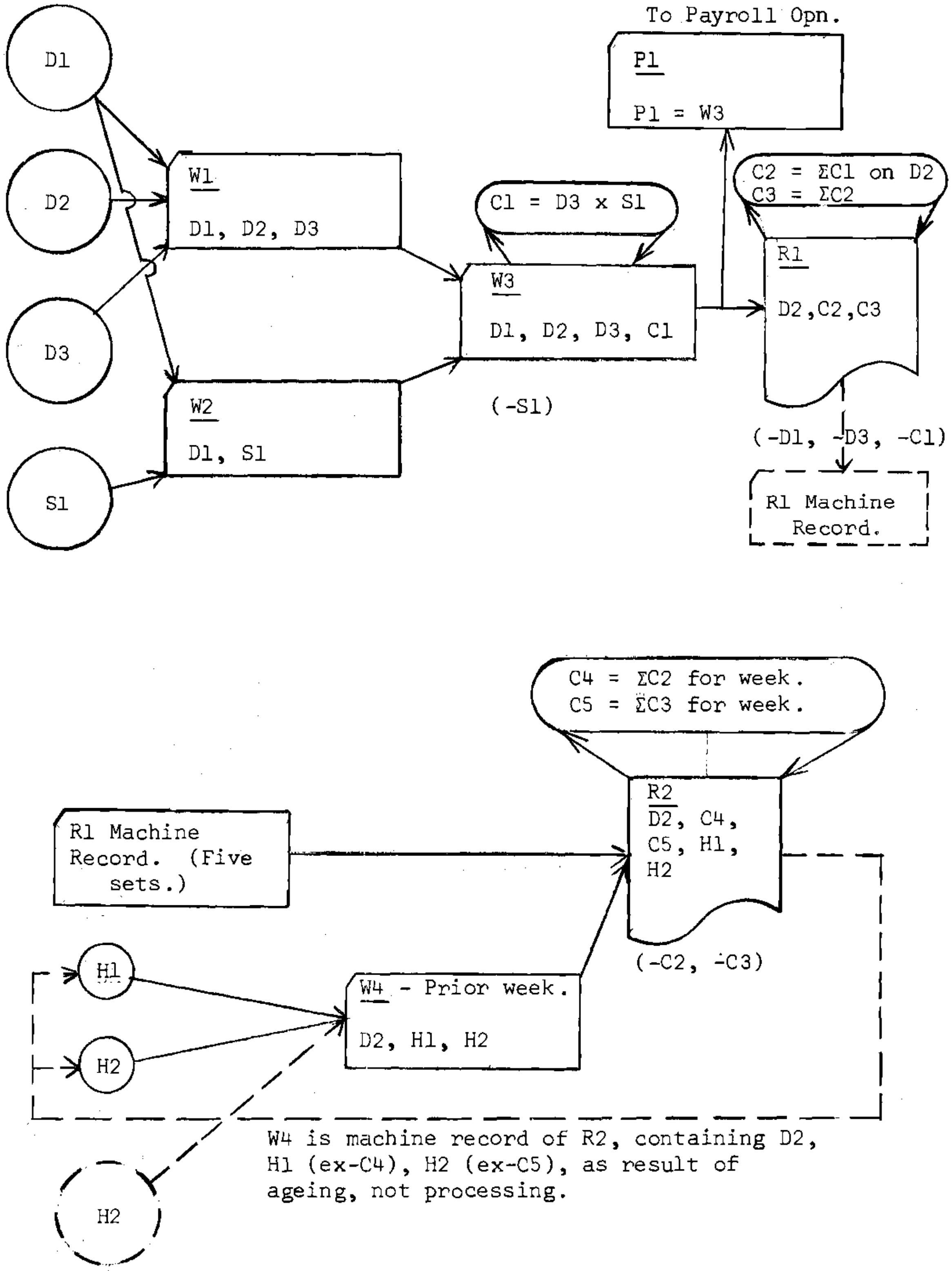

Figure 9. Flow Chart for System of Example.

C4 - Department total, weekly. (C4 = C2 summed over 5 days)

C5 - Grand total, weekly. (C5 = C3 summed over 5 days)

H1 - C4 value for previous week.

H2 - C5 value for previous week.

R1 - Daily report.

R2 - Weekly report.

W1 - Input record. (Working paper, contains D1, D2, D3)

W2 - Master card. (Contains S1 and D1)

W3 - Card containing C1, D1, D2, D3.

W4 - Card containing same contents as R1, but aged one week. (Contains D2, H1, H2)

P1 - Card to payroll operation. Has same content as W3.

From the verbal description of the system (and in this case from the flow diagram, Figure 9), the following prime relations can be established. The first six are input data, which are arbitrarily designated (0,j), with a quantifier of 1.

0	D1	1	
0	D2	1	
0	D3	1	
0	H1	1	} These two elements follow the rule stating that "aged" data be treated as new input.
0	H2	1	
0	S1	1	
D1	W1	1	
D2	W1	1	
D3	W1	1	
D1	W2	1	
S1	W2	1	
W1	W3	1	
W2	W3	1	
W3	P1	1	
W3	R1	1	
H1	W4	1	
H2	W4	1	
R1	R2	5	Five daily reports required for weekly report.

W4	R2	1
D2	W4	1

The concomitant relations are those pertaining to data transitions, and for this system are as follows:

C1	W3	D3	S1	(1)
C2	R1	C1		(1)
C3	R1	C2		(1)
C4	R2	C2		(1)
C5	R2	C3		(1)

The deletion relations are:

C1	R1	-1
D1	R1	-1
D3	R1	-1
S1	W3	-1
C2	R2	-1
C3	R2	-1

The systems relations having been established, the next step is that of triangularization. The deletion relations are not used in this process, and the concomitant relations are not used until step 5. The first step is the preparation of ordered lists A and B, which appear as Table 16.

In performing step 2 of the algorithm, using the lists A and B, one notes that the first j in list A is D1. Looking down the list B, it is noted that there is an i = D1 in the list, the seventh pair down. Therefore, the pair (or pairs) with j = D1 in list A is retained. Similarly, for j = D2, D3, H1, and H2 in list A a corresponding i in list B is found. Finally, j = P1 in list A is encountered, with no corresponding i = P1 in list B. Therefore the pair W3 P1 is removed from list A, as indicated in Table 16 by underscoring.

Table 16. Lists A and B, Ordered for Triangularization*

LIST A		LIST B	
i	j	i	j
0	D1	0	D1
0	D2	0	D2
0	D3	0	D3
0	H1	0	H1
0	H2	0	H2
W3	P1	0	S1
W3	R1	D1	W1
R1	R2	D1	W2
W4	R2	D2	W1
0	S1	D3	W1
D1	W1	H1	W4
D2	W1	H2	W4
D3	W1	R1	R2
D1	W2	S1	W2
S1	W2	W1	W3
W1	W3	W2	W3
W2	W3	W3	P1
H1	W4	W3	R1
H2	W4	W4	R2

* The underscored entries in the table represent the pairs removed in the first iteration of the algorithm.

The next entry, j = R1 finds i = R1 in list B, so the pair is retained in list A. However, the next entry with j = R2 does not have a corresponding i = R2 in list B, so both of the pairs in list A with j = R2 are removed, as indicated by the underscored pairs. The balance of the j's in list A find corresponding i's in list B, and so remain in the list.

In step 3, all pairs (i,j) removed from list A in step 2 are now also removed from list B. This operation is again indicated in Table 16 by the underscored entries in list B. Also, the order of removal of pairs from list A is recorded at this time. Since lists A and B are not exhausted, step 4 now indicates that reduced lists A and B be formed, and steps 2 and 3 be repeated on these reduced lists. These reduced lists and the removals due to the second iteration of steps 2 and 3 are shown in Table 17.

Step 2 this time resulted in the removal of pairs from list A where j = R1 and j = W4, since corresponding i's in list B could not be found. Step 3 resulted in the removal of the same pairs from list B, and the listing of the order of removal of pairs from list A. Since the lists A and B were not exhausted, step 4 now calls for preparation of further reduced lists A and B, shown in Table 18, together with the pairs removed as a result of the next, or third, iteration.

On this third iteration, the pairs in list A involving j = H1, H2, and W3 are removed from list A, and subsequently from list B. The reduced lists A and B shown in Table 19 remain, and a fourth iteration removed pairs involving j = W1 and W2. Further reduced lists A and B (not shown) would be subjected to a fifth iteration which would remove the balance of the pairs from both lists.

Since both lists are now exhausted, the algorithm proceeds to step 5. At this point, the record of order of removal of pairs from list A, prepared in step 3 of the algorithm is used. In addition, the list of concomitant n-tuples is ordered as indicated in step 5 of the algorithm is now used in conjunction with the record of order of removal in order to establish an augmented order of removal which will give the proper order of indices for the S, S', S^* and S'^* matrices by including those elements formed by a concomitant relation. Table 20 shows these three lists.

Table 17. First Reduction of Lists A and B*

LIST A		LIST B	
i	j	i	j
0	D1	0	D1
0	D2	0	D2
0	D3	0	D3
0	H1	0	H1
0	H2	0	H2
W3	R1	0	S1
0	S1	D1	W1
D1	W1	D1	W2
D2	W1	D2	W1
D3	W1	D3	W1
D1	W2	H1	W4
S1	W2	H2	W4
W1	W3	S1	W2
W2	W3	W1	W3
H1	W4	W2	W3
H2	W4	W3	R1

* The underscored entries in the table represent the pairs removed in the second iteration of the algorithm.

The S and S' matrices may now be formed, using the augmented list (Table 20) for the row and column indices. The prime and concomitant relations are shown in the S matrix, while the S' matrix shows all three categories of relations. Figures 10 and 12 show the S and S' matrices, respectively, which are obtained from the data of the example by application of the rules of S and S' matrix formation.

Table 18. Second Reduction of Lists A and B*

LIST A i	LIST A j	LIST B i	LIST B j
0	D1	0	D1
0	D2	0	D2
0	D3	0	D3
0	H1	0	H1
0	H2	0	H2
0	S1	0	S1
D1	W1	D1	W1
D2	W1	D1	W2
D3	W1	D2	W1
D1	W2	D3	W1
S1	W2	S1	W2
W1	W3	W1	W3
W2	W3	W2	W3

* The underscored entries in the table represent the pairs removed in the third iteration of the algorithm.

Table 19. Third Reduction of Lists A and B*

LIST A i	LIST A j	LIST B i	LIST B j
0	D1	0	D1
0	D2	0	D2
0	D3	0	D3
0	S1	0	S1
D1	W1	D1	W1
D2	W1	D1	W2
D3	W1	D2	W1
D1	W2	D3	W1
S1	W2	S1	W2

* The underscored entries in the table represent the pairs removed in the fourth iteration of the algorithm.

The S^* and S'^* matrices, shown in Figures 11 and 13, are then obtained by the operation of the two refined algorithms presented. In order to show more clearly the operation of the algorithms, the details of computation of the R2 column of each output matrix will be shown. In both cases, presume that the computation has been completed for all columns to the right of the R2 column. The algorithm will then be entered at step 3. The $S - S^*$ algorithm will be demonstrated first.

1. The vector V is established, with 17 cells being required, all zero initially.

Table 20. Record of Order of Removal, Ordered Concomitant N-Tuples, and Augmented Record of Order of Removal.

Order of Removal	Ordered N-Tuples	Augmented Order of Removal
P1	C5 R2 C3	P1
R2	C4 R2 C2	R2
R1	C3 R1 C2	C5
W4	C2 R1 C1	C4
H1	C1 W3 D3 S1	R1
H2		C3
W3		C2
W1		W4
W2		H1
D1		H2
D2		W3
D3		C1
S1		W1
		W2
		D1
		D2
		D3
		S1

2. In step 4, the bottom cell of the R2 column of S would be examined, and found to be zero. Step 4 would therefore be repeated for the next to bottom cell, which is also zero. Step 4 would then be repeated ten times, each time encountering a zero value, and thus passing to the next higher cell in the column.

3. On the 11th iteration, the cell value 1 would be encountered. Since the cell is non-zero, the row index W4 would be established, and the W4 Column of S^* would be obtained. Step 5 of the algorithm then calls for the multiplication of the cell value 1 by the column vector

	P1	R2	C5	C4	R1	C3	C2	W4	H1	H2	W3	C1	W1	W2	D1	D2	D3	S1
P1																		
R2																		
C5		(1)																
C4		(1)																
R1		5																
C3			(1)		(1)													
C2				(1)	(1)	(1)												
W4		1																
H1								1										
H2								1										
W3	1				1													
C1							(1)				(1)							
W1											1							
W2											1							
D1													1	1				
D2								1					1					
D3												(1)	1					
S1												(1)		1				

Blank cell denotes zero entry.

Figure 10. S Matrix--Refined Algorithm.

	P1	R2	C5	C4	R1	C3	C2	W4	H1	H2	W3	C1	W1	W2	D1	D2	D3	S1
P1	1																	
R2		1																
C5		(1)	1															
C4		(1)		1														
R1		5			1													
C3		(5)	(1)		(1)	1												
C2		(5)		(1)	(1)	(1)	1											
W4		1						1										
H1		1						1	1									
H2		1						1		1								
W3	1	5			1						1							
C1	(1)	(5)			(1)		(1)				(1)	1						
W1	1	5			1						1		1					
W2	1	5			1						1			1				
D1	2	10			2						2		1	1	1			
D2	1	6			1			1			1		1			1		
D3	1	5			1						1	(1)	1				1	
S1	1	5			1						1	(1)		1				1

Blank cell denotes zero entry.

Figure 11. S^* Matrix--Refined Algorithm.

	P1	R2	C5	C4	R1	C3	C2	W4	H1	H2	W3	C1	W1	W2	D1	D2	D3	S1
P1																		
R2																		
C5		(1)																
C4		(1)																
R1		5																
C3		-1	(1)		(1)													
C2		-1		(1)	(1)	(1)												
W4		1																
H1								1										
H2								1										
W3	1				1													
C1					-1		(1)				(1)							
W1											1							
W2											1							
D1					-1								1	1				
D2								1					1					
D3					-1							(1)	1					
S1											-1	(1)		1				

Blank cell denotes zero entry.

Figure 12. S' Matrix--Refined Algorithm.

	P1	R2	C5	C4	R1	C3	C2	W4	H1	H2	W3	C1	W1	W2	D1	D2	D3	S1
P1	1																	
R2		1																
C5		(1)	1															
C4		(1)		1														
R1					1													
C3			(1)		(1)	1												
C2				(1)	(1)	(1)	1											
W4								1										
H1		1						1	1									
H2		1						1		1								
W3											1							
C1	(1)						(1)				(1)	1						
W1													1					
W2														1				
D1	2										2		1	1	1			
D2	1	6						1			1		1			1		
D3	1										1	(1)	1				1	
S1												(1)		1				1

Blank cell denotes zero entry.

Figure 13. S'* Matrix--Refined Algorithm.

W4, and the addition of the product to Vector V. This operation is shown in Table 21.

Table 21. Computation of Column R2 of S^*--First Step

Cell	X	Col W4	=	Product	+	Vector V Before	=	Vector V After	Row Index
						0		0	R2
						0		0	C5
						0		0	C4
						0		0	R1
						0		0	C3
						0		0	C2
1		1		1		0		1	W4
		1		1		0		1	H1
		1		1		0		1	H2
		0		0		0		0	W3
		0		0		0		0	C1
		0		0		0		0	W1
		0		0		0		0	W2
		0		0		0		0	D1
		1		1		0		1	D2
		0		0		0		0	D3
		0		0		0		0	S1

4. Step 4 would then be repeated for the cell next above in the column of S, but again a zero would be encountered. This would cause another repetition of step 4, another encountering of a zero value, then the next iteration of step 4 would encounter the value 5. The row index is R1, so the R1 column of S^* would be obtained. This time, the multi-plication--addition process would be as shown in Table 22. Note that the concomitant relation is involved, invoking the use of the multiplication-

like operation previously described for some cells.

5. The next repetition of step 4 encounters the concomitant quantifier (1), with row index C4, which requires column C4, of S^*, for computation. The operation is shown in Table 23.

Table 22. Computation of Column R2 of S^*--Second Step

Cell	X	Col R1	=	Product	+	Vector V Before	=	Vector V After	Row Index
						0		0	R2
						0		0	C5
						0		0	C4
5		1		5		0		5	R1
		(1)		(5)		0		(5)	C3
		(1)		(5)		0		(5)	C2
		0		0		1		1	W4
		0		0		1		1	H1
		0		0		1		1	H2
		1		5		0		5	W3
		(1)		(5)		0		(5)	C1
		1		5		0		5	W1
		1		5		0		5	W2
		2		10		0		10	D1
		1		5		1		6	D2
		1		5		0		5	D3
		1		5		0		5	S1

6. On the next iteration, the value (1) is found with row index C5. The computation is shown in Table 24.

7. The next iteration of step 4 encounters a zero cell which is the diagonal cell. Step 6 then requires the addition of the vector V to the R2 column of S^*. This column has only a 1 in the diagonal cell,

so this operation amounts to transcribing vector V into the S^* matrix, with a 1 placed in the R2,R2 cell. This completes the computation of this column.

Table 23. Computation of Column R2 of S^*--Third Step

Cell	X	Col C4	=	Product	+	Vector V Before	=	Vector V After	Row Index
						0		0	R2
						0		0	C5
(1)		1		(1)		0		(1)	C4
		0		0		5		5	R1
		0		0		(5)		(5)	C3
		(1)		0		(5)		(5)	C2
		0		0		1		1	W4
		0		0		1		1	H1
		0		0		1		1	H2
		0		0		5		5	W3
		0		0		(5)		(5)	C1
		0		0		5		5	W1
		0		0		5		5	W2
		0		0		10		10	D1
		0		0		6		6	D2
		0		0		5		5	D3
		0		0		5		5	S1

The computation of a column of S'^* proceeds in a similar manner. Entering the algorithm at step 3, for column R2:

1. The vector V is established, with 17 cells being needed, all zero initially.

2. Step 4 would encounter zeros on the first 10 iterations, until a 1 is finally encountered with W4 as row index and W4 as the column of

S'^{*} to be used in multiplication. Step 5 would then result in the computation detailed in Table 25.

Table 24. Computation of Column R2 of S^{*}--Fourth Step

Cell	X	Col C5	=	Product	+	Vector V Before	=	Vector V After	Row Index
						0		0	R2
(1)		1		(1)		0		(1)	C5
		0		0		(1)		(1)	C4
		0		0		5		5	R1
		(1)		0		(5)		(5)	C3
		0		0		(5)		(5)	C2
		0		0		1		1	W4
		0		0		1		1	H1
		0		0		1		1	H2
		0		0		5		5	W3
		0		0		(5)		(5)	C1
		0		0		5		5	W1
		0		0		5		5	W2
		0		0		10		10	D1
		0		0		6		6	D2
		0		0		5		5	D3
		0		0		5		5	S1

The critical point of this computation is the examination of column W4 for structure, and the ignoring of the diagonal cell because of the presence of other non-zero entries in the column.

3. The next two iterations of step 4 encounter deletion relations which result in the placing of an X in vector V in the C2 and C3 positions. These X's will prevent any further entries in these cells. These two iterations will not be shown, but the result will be reflected

in subsequent detailed computations.

Table 25. Computation of Column R2 of S'^{*}--First Step

Cell	X	Col W4	=	Product	+	Vector V Before	=	Vector V After	Row Index
						0		0	R2
						0		0	C5
						0		0	C4
						0		0	R1
						0		0	C3
						0		0	C2
1		1 (ignored)		0		0		0	W4
		1		1		0		1	H1
		1		1		0		1	H2
		0		0		0		0	W3
		0		0		0		0	C1
		0		0		0		0	W1
		0		0		0		0	W2
		0		0		0		0	D1
		1		1		0		1	D2
		0		0		0		0	D3
		0		0		0		0	S1

4. The value 5, corresponding to row R1 is next found, and column R1 is obtained. Column R1 has non-zero, non-diagonal values, so the diagonal cell is treated as zero in the computation shown in Table 26.

It should be noted that the X's placed in vector V by action of the deletion relations have blocked out the 5's obtained in the product vector for the C3 and C2 cells.

5. The next encountered values in the R2 column of S' are both concomitant relations, and the computation would be exactly the same as

for the corresponding cells of the earlier example, with, of course, a different vector V due to different prior operations. The net result would be the insertion of (1) in the vector V in the C5 and C4 positions. Finally, addition of the vector V to the column R2 of S'^* would give the result shown in Figure 13. Note that the deletion symbol X in the vector V is treated as a zero in this final addition.

Table 26. Computation of Column R2 of S'^*--Fourth Step

Cell	X	Col R1	=	Product	+	Vector V Before	=	Vector V After	Row Index
						0		0	R2
						0		0	C5
						0		0	C4
5		1 (ignored)		0		0		0	R1
		1		5		X		X	C3
		1		5		X		X	C2
		0		0		0		0	W4
		0		0		1		1	H1
		0		0		1		1	H2
		0		0		0		0	W3
		0		0		0		0	C1
		0		0		0		0	W1
		0		0		0		0	W2
		0		0		0		0	D1
		1		5		1		6	D2
		0		0		0		0	D3
		0		0		0		0	S1

Discussion of the Example

In order to discuss the example, it is informative to introduce an S^* matrix, shown in Figure 14, produced by the use of the unrefined algorithm presented in Chapter IV, with no consideration given to concomitant and deletion relations, and with the prime relations established from a report-oriented point of view.

It is immediately evident from the figure that the S^* matrix produced with the refined algorithm has eliminated the false indications of redundancy. The entries with value 5 in the S^* matrix of the refined algorithm are valid representations of the fact that five reports R1, one for each of the five days of the week, were used in preparation of R2. The D2 entry of 6 indicates the merging of the five R1 reports on department number, then the merging of W4, containing the historical data, with the five daily reports. The entry 10 for D1 might be viewed as false redundancy, but again, it represents the merging of a W1 with a W2 for the initial computation of C1, recorded on W3, then the subsequent merging of five R1's for the R2 report. C4 and C5 are noted as values which were prepared as a computation concomitant to the preparation of R2, which conforms to the actual system concept. Similar comments could be made relative to R1, W3, and P1, and it is seen that the refined S^* matrix is a true representation of the system on a "composed of" basis.

Turning attention to the S'^* matrix, Figure 13, it is seen that it is an accurate representation of the system on a "contains" basis, with the possible exception of values greater than one for identification type data in P1, R2, and W3. Again, these values represent the merging of compound elements on the basis of the identification data, and is thought to represent useful information.

It might be argued that the S'^* matrix represents information

	P1	R2	C4	C5	W4	H1	H2	R1	C2	C3	W3	C1	W1	W2	D1	D2	D3	S1
P1	1																	
R2		1																
C4		1	1															
C5		1		1														
W4		1			1													
H1		1			1	1												
H2		1			1		1											
R1		10	5	5				1										
C2		10	5	5				1	1									
C3		10	5	5				1		1								
W3	1	20	10	10				2	1	1	1							
C1	1	20	10	10				2	1	1	1	1						
W1	1	20	10	10				2	1	1	1	1	1					
W2	1	20	10	10				2	1	1	1	1		1				
D1	2	40	20	20				4	2	2	2	2	1	1	1			
D2	1	21	10	10	1			2	1	1	1	1	1			1		
D3	1	20	10	10				2	1	1	1	1	1				1	
S1	1	20	10	10				2	1	1	1	1		1				1

Blank cell denotes zero entry.
System is same as for Figures 11 and 13, but computation is by unrefined algorithm.

Figure 14. S^* Matrix--Unrefined Algorithm.

which is known prior to computation, from flow charts of the system and other basic data. To an extent, this is true. However, part of the aim of this research is to eliminate the necessity for flow charting, and the S'^{*} matrix is an effective substitute.

It may be observed that both S^{*} and S'^{*} show that P1 is identical in data composition and content to W3. Furthermore, both of these documents are in card form, so that P1 is not needed except perhaps as a matter of processing convenience. In a real system analysis such an indicated duplication would be investigated with a view to possible elimination.

CHAPTER VII

CONSIDERATIONS OF MECHANIZATION AND COMPUTER OPERATION

Introduction

In current day practice, it is usual and expected that numerical values arising from the use of a given mathematical technique will be obtained by means of a digital computer. In the early days of computer operations, algorithms which were designed for manual operation required substantial modification in many cases before they could successfully be used for computer operation, and a whole new subject, that of numerical analysis, has sprung up as a branch of the mathematics.

A more recent trend has been to write algorithms specifically from the point of view of the essential computer logic necessary for numerical calculation. Giffler's whole work (9) has adopted this style, with real advantage to those who seek to employ his algorithms. It is also now commonly seen in the literature that an appendix of an article contain at least the flow diagram, and perhaps the program listing in FORTRAN or ALGOL, in order to facilitate both understanding and possible use by the reader.

In the work under discussion, it seems even more appropriate that the algorithms be stated in language appropriate to computer usage, and that the computational portion of the techniques be specified in terms meaningful to those engaged in computer operations. The research is, in effect, an attempt to permit the central processor of the information system under study to become introspective relative to the system it serves. The algorithms in preceding chapters were developed with the view in mind that their computation under real world considerations would be done by means of digital computers and associated equipment, even though it was necessary to present the algorithms as if they were to be manually computed, for expository purposes.

Sections of the Appendix will be devoted to restatement of the algorithms presented in terms of detailed instructions for triangularizing indices by means of punched card equipment, and flow charts and program listings for the computation of the S^* and S'^* solution matrices. Every attempt has been made to preserve generality from the point of view of equipment, especially in the flow diagrams. However, since programs must be written for real machines, some generality has been lost in the fact that the procedures and program listings apply only to the specific machine for which they are written, except that in some cases, as specifically noted, the program would run on equipment with fewer special features.

For the triangularization process, only a card sorter, collator, and reproducing punch are required. Any of the models currently on the market are suitable.

The two algorithms for producing the S^* and S'^* solution matrices have been written for an IBM 1401 computer, with a minimum of 8K memory, and 4 tape drives. Advanced programming features are also required. The equipment was chosen specifically because it is representative of the type and size of equipment which might be found in a typical manufacturing, commercial, or institutional situation. It is felt that installations using equipment of similar or greater capabilities could readily adapt or re-write the programs to suit their own capabilities. For other IBM equipment in the 1400 series, this adaptation would of course be very simple.

In order to make the maximum use of the logic capabilities and of available memory, the programming has been done in 1401 Autocoder language. This language necessitates the use of 4 tape drives for compilation. The algorithms could also be programmed in ALGOL or FORTRAN, but with substantial loss of operating speed and memory capacity.

Triangularization and Reindexing

An essential part of the computational technique leading to the S^* and S'^* solution matrices is the triangularization of the S and S' matrices. In order to obviate tedious labor, it is desirable that the mechanized portion of the computational system accept designations of systems elements which are entirely arbitrary; that is, that the analyst need not be concerned with the mechanics of triangularization nor the subsequent reindexing of elements for computer operation.

Giffler (9, p. 18-34) presents an algorithm which can be used for triangularization, and he claims that it can be accomplished on punch card equipment. Trials made using this algorithm disclose several important limitations, however.

1. The algorithm is not in fact amenable to operation on punch card equipment. The collator is not capable of interchanging cards as the algorithm requires, it can only select and segregate matches or mis-matches. Neither is the collator capable of recording the exchanges

and comparing the current exchange with all prior ones.

2. The algorithm can be programmed for computer operation, but in actual trials it proved to be an inefficient operation, in that the matter of recording and comparing index changes required repeated table look-up operations.

3. The i indices are left scattered in the final ordering obtained by operation of the algorithm. That is, if there are several pairs (i,j) in the system with the same i value, and only some of the i find matches on the following j's, then the pairs with the same i will not be contiguous in the resulting ordering. This fact makes the required reindexing (for computer solution of the ensuing computations) very difficult, or else requires a further step to consolidate the indices.

4. Since the list is not reduced in size as the algorithm progresses, it is necessary to store, and operate with, the entire list during the entire computation. Furthermore, no output can be realized from the computer until the entire triangularization is complete, which results in long overall computation times.

For these reasons, a new algorithm was developed, which has been stated both in manual form (p. 121) and mechanized (Appendix B) form. Both forms also contain provision for handling concomitant and deletion relations as well as prime relations.

In the new algorithm, the use of two decks of cards (lists A and B in the manual version) permits the use of a collator to both recognize and remove the successive subsets, while careful ordering of the initial set and of the order of removal prevents the scattering of indices noted in Giffler's method. Reindexing for computer operation is readily done, and consists simply of assigning sequential numbers to the elements of the triangularized lists or decks in a systematic manner.

Computation of the Solution Matrices

The principle concern in development of computational methods for mechanization of the algorithms for computation of S^* and S'^* was that of limitations of memory, particularly if a computer of modest size is to be used. It is therefore necessary to give every consideration to computational techniques in order to conserve this scarce memory space.

Again, Giffler (Chapter II, p. 40) established computational expressions for the inversion of the (I - N) matrix which could be easily translated to computer operation. However, his computational method would require that the S matrix be stored in memory in its entirety before computation could start, and that the S^* matrix be overlayed on the S matrix as computation proceeded. This was due to the fact that the computational equations required solution of columns from left to right in the case of a lower triangular matrix. Reconsideration of the computation, which is reflected in the statement of the algorithms in Chapter VII, shows that if the computation proceeds from right to left (in a lower triangular matrix) it is not necessary to store the S matrix at all. It is only necessary to read from a card a single cell s_{ij}, and then to perform all the computation required by the entry s_{ij} at one time, storing the results in memory in the S^* matrix.

The recognition of the ability to eliminate storage of the S matrix permits consideration of methods of compressing the S^* matrix in memory as it is generated, and other means of conserving memory space. It is readily noted that an uncompressed S^* matrix, due to triangularity, will have as an upper bound,

$$C = \frac{n^2 + n}{2} \tag{1}$$

cells, including the diagonal cells, where n is the number of systems elements under consideration. In addition, some space would be required for identification, work areas, and, of course, the computer program.

A first saving in memory allocation can be made by observing that the diagonal 1's are arbitrarily present in every column of S^*, and that they could always be supplied, if needed, by the program on recognition that the cell in question has index $i = j$. In the program, the cells so conserved will be used for column and row identification and other information.

A substantial saving in memory space can be realized by observing that S^* has many zero cells. By further consideration of the property discussed in Chapter V relative to formation of columns as linear combinations of columns to the right, it can be noted that a given column of the S^* matrix will never have non-zero elements with row index i smaller than the smallest non-zero cell row index of the same column of the S matrix, except for the diagonal element itself, which will not be stored as such. That is, if there are zero cells between the diagonal and the first (highest in column) non-zero cell in a given column of the S matrix, then the S^* matrix will have at least these same zero cells. This observation permits two things to be done:

1. These specific cells need not be stored in the matrix, thereby conserving storage. The detailed method of accomplishing this will be discussed in the Appendix dealing with the computer program.

2. A means is provided to compute the number of zero cells of this type, so that the prior statement of the upper bound of memory required can be improved. The computation is based on the fact that if the triangularized list of systems relations is re-ordered with major ordering on the column index j and minor ordering on the row index i, then the first pair (i,j) within a group of identical j indicates the cell which will occupy the highest non-zero position of the jth column of S, and therefore of S^*. Then the number of intervening zero cells between the diagonal cell and this highest cell is $i - (j + 1)$. If a j index is not present in the list, a null column of S is indicated, and the column in S^* will have only the diagonal cell. In this case, the number of zero cells is $n - j$, where n is the number of system elements in the system under study.

Define a list R of all pairs (i,j) in a system. Define a sublist L as the pairs $(i,j) \varepsilon R$ such that i is the smallest index associated with a given column index j. Note that for a given system, there will be some columns j with no pairs (i,j). Then the number Z of zero cells of the type under discussion is:

$$Z = \sum_{\substack{j=1 \\ j \in L}}^{n} i_j - (j + 1) + \sum_{\substack{j=1 \\ j \notin L}}^{n} (n - j) \qquad (2)$$

Then the upper limit U on storage required for a given matrix will be:

$$U = \frac{n^2 + n}{2} - Z \qquad (3)$$

A simple program to compute Z and U from the input cards will be found in Appendix D.

One further possibility for conservation of storage space derives from the computation of S^* from the point of view of linear combinations of columns. This depends on the property that after a column computation involving a given s_{ij}, if there are no further non-zero values to the left of s_{ij} in the same (ith) row, there will be no further need in the calculation for the corresponding column of S^* with index $j = i$. Presuming that this column has been presented in some form of output, this column can then be discarded in its entirety, making room for other column vectors, and therefore permitting a larger S^* matrix to be computed and stored.

In order to use this property, it is necessary to locate and designate the cells s_{ij} having this property. This is easily done by ordering the triangularized list (sorting the cards) with major sequence on i, minor sequence on j. Then the first occurrence of each i in the list will identify the proper cells. These cells can be identified by tagging; that is, punching some designator in the card in an appropriate field.

This last method of compression is valuable under circumstances when a problem could not otherwise be run on a given computer. Its use, however, complicates the programming because of the fact that the discarding of a vector is accompanied by the need to completely rearrange the remaining portion of the S^* matrix in memory, and changing the internal indexing. This also increases the running time, substantially in some cases. It is therefore recommended that the method be used only when absolutely necessary. The programs in the Appendix do not incorporate this feature.

CHAPTER VIII

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

The development of mathematical models for the description and manipulation of information system relationships represents an attempt to quantify a process which has been approached in the past on an essentially qualitative and intuitional basis. Only very limited attempts in this direction are shown in the literature.

The basic model developed in Chapter IV has no real meaning from the point of view of application, but serves as an essential bridge between the prior work of Lieberman (18) and Homer (14) and the refined models presented in Chapters V and VI. The basic model did, however, succeed in reproducing the results of the prior investigators with some computational and conceptual improvement, but it failed to adequately represent system under study.

The recognition of differences between types of data and of the existence of transitions of data within the system led to the definition of three types of system relations instead of the single type presumed in the prior work and in the basic model. This in turn permitted extension of the basic model to the two refined models.

From an operation point of view, examination of the S^* and S'^* matrices by the analyst will disclose possible areas of system improvement. The existence of a number larger than expected[†] in a given cell will indicate the possibility of redundancy in the number of times a given element is made available for preparation of a given report. The partial or complete duplication of a column will suggest the possibility of eliminating or combining two or more reports. Comparison of columns of S^* and S'^* will give an indication of data elements from lower levels which might easily be added to higher level reports in order to make them more useful.

The fact that the matrices may represent a proposed system as well as an existing system permits comparison and evaluation of relative redundancy in two systems. Furthermore, the fact that the algorithms are mechanized allows the use of the technique for simulation of proposed system changes, and can in this manner be used for synthesis as well as analysis of information systems.

In summary, the benefits of the refined techniques are that the shortcomings noted in prior work have largely been overcome. In detail, it is felt that the refined techniques have achieved the following benefits:

1. The three basic types of data, and the several transitions to which data may be subjected, have been explicitly recognized, and notational and computational techniques have been devised to adequately deal with them.

2. The overstatement of redundancy inherent in prior techniques

---

† The analyst would expect the number five, for example, if five daily reports were summarized to produce a weekly report.

has been eliminated for type two and type three data, and reduced to a legitimate indication of frequency of merging for type one data. A major concern in the analysis of systems is to assure that adequate information reaches the decision maker with a minimal amount of duplicate data origination and processing. The refined models can furnish a basis for elimination of duplication of origination, and can do much in disclosing duplication of processing.

3. By definition of the categories of system relations, and specification of rules for establishing these relations, the analyst is forced to carefully define each observed relation, rather than merely indicating that some relation exists. Further, he is forced to look at the system from a data element point of view, as opposed to the organizational or report hierarchy point of view, which assures that a more detailed analysis will be made. This results in removing much of the distinction between the interior and exterior system, which should be of advantage in realizing an effective final design.

4. The analyst is freed from a great deal of tedious description of a prose or graphic nature. The proposed technique does not require the preparation of flow charts or block diagrams. Furthermore, the analyst need only be concerned with defining the relations, and need not be concerned with determining the level at which they occur, nor with arranging them in any particular order. The mechanized algorithms perform this function for him as a necessary step in the overall analysis. The time of the analyst is freed for examination of the solution matrices which are a true and adequate representation of the system from both a "composed of" and "contains" point of view.

5. A very great part of the entire analysis technique can be programmed for a digital computer, greatly reducing the clerical activity required of the analyst, and also greatly reducing the volume of data which he must examine. The two solution matrices contain much of the data needed for analysis of the system.

6. Since the computation is carried out in the computer, it becomes feasible to introduce changes of definitions of relations on a trial basis. In this way, the technique can be used for simulation of proposed system changes, and can in this manner be used for synthesis as well as analysis of data systems.

7. An adequate solution for the problem of cycling caused by ageing of data has been advanced by treating stored, then subsequently retrieved data, as a new data origination.

8. All of the benefits cited by Homer (Chapter II, p. 31) have been met, and in most cases, exceeded.

In general it is believed that the models presented will furnish an adequate representation of information processing systems of the type encountered in practice. Like all models, they will not cover all conceivable situations in complete detail, but they will adequately represent the principal structure of a system. It would be difficult, for example, to represent the situation where a randomly selected portion of the data is used in the preparation of a report, such as is sometimes done in more sophisticated systems. Furthermore, the model represents only the structure of the system, and does not consider the volume of a given data element to be processed.

It can be fairly stated, then, that while this research is believed to be a step, hopefully a major one, in the direction of quantitative analysis of information processing systems, that much remains to be done before information systems analysis can be viewed as a science, rather than an art.

Recommendations

Since in consideration of the total field of information processing so much remains to be done, the recommendations for further study must be considered as indicative rather than inclusive.

Of a practical and applicatory nature, the models of the current research should be applied to a number of real applications, to determine the range of the models' applicability. Care should be taken to include project-oriented organizations as well as those engaged in more repetitive activities.

In a more theoretical vein, the following studies are suggested:

1. Investigation of other models for the structural examination of information processing systems.

2. Investigation of means for characterizing information processing systems in terms of the volume of processing.

3. Investigation of the use of random variables as quantifiers for the system relations.

4. Investigation of the essential similarities between computational subsystems (e.g., payroll and inventory accounting) with a view to developing general methods of computation.

5. Investigation of the basic compatibility of programming languages (e.g., COBOL) and information processing systems requirements.

APPENDIX A

GLOSSARY

The following Glossary of terms unique to this research is presented:

Composed of - A kind of information systems analysis where, for a given report, all of the lower level data which were required for the preparation of the report are indicated, whether or not they actually appear on the report as prepared. Used in opposition to the term "Contained in."

Compound Element - Any report, form, document, card, tape, or disk record of any nature containing one or more pieces of recorded data (simple elements), and from which the simple elements can be recovered and made meaningful to humans without arithmetic computation.

Concomitant Relation - A relationship within an information system which occurs in conjunction with the preparation of a compound element wherein one or more simple elements undergo a transition.

Concomitant Cell - A cell in a matrix wherein the value is a concomitant relation quantifier, (c).

Concomitant Index - A row or column index of a matrix where the index is the identification of a system element formed as a result of a concomitant relation.

Contained in - A kind of information system analysis where, for a given report, only the lower level system elements which actually appear on the report are indicated. Used in opposition to the term "Composed of."

Data Transition - An operation within an information system wherein the input data are transformed in some manner to make them more usable to the decision maker. Nine forms of transition are identified and discussed (p. 52).

Deletion Relation - The system relation which exists when an element from a lower level is brought to a higher level by virtue of being included within a compound element which is required at a higher level, but which in itself does not become an integral part of the higher level. It is used the indicate that an element is terminated on a given route through the system.

Exterior System - The entire operation under consideration, including the set of decisions made routinely to control the operation, the set of input data to be processed in order that a decision may ultimately be made, the required data origination, data transmission, data processing, report structure, provision for data storage and subsequent retrieval, and the set of decision function which will be used in conjunction with the processed data in order to arrive at specified decisions.

Identification Data - A simple element of an information system which indicates the relation of other types of data to some physical system. Examples are payroll numbers, stock numbers, machine numbers and department numbers.

Interior System - The interior system refers to the details of machine programming and operation. It embraces such factors as card and record format, report format, operating rules, and equipment programming itself.

Pair, or Pair (i,j) - Denotes the notation used to indicate a prime or deletion relation.

N-tuple (i,j,k,...,n) - The notation used to indicate concomitant relations.

Prime Relation - The system relation which indicates that an element at some given level of a data system is merely required for the preparation of some higher level element. All data input is prime to the first document upon which it is recorded and to its source.

Quantifier - A symbol used to express either numerically or symbolically the extent of a system relation. Prime relations use as a quantifier ordinary numerals n, concomitant relations employ a special quantifier (c), where c is an integer, and deletion relations use a symbolic quantifier -1.

Quantitative Data - Data which arise as a result of a count, measurement, or computation.

Regular Cell - A cell in a matrix wherein the value is a regular scalar, n.

Regular Index - A row or column index of a matrix where the index is the identification of a system element formed as a result of a prime relation.

Simple Element - Any piece of recorded data, numeric, alphabetic, or symbolic, which exists in an information system as input, output, or intermediate form, regardless of the means of recording.

Status Data - Data which indicate status or existence. This type of data is not quantitative.

System Element - General term for either a simple element or a compound element where the distinction is not needed.

<u>Triangularization</u> - The logical operation of ordering the identifications of system elements in such a manner that when these identifications are used as indices for the S and S' matrices, the matrices will be in triangular form.

<u>Type One Data</u> - Identification data.

<u>Type Two Data</u> - Quantitative data.

<u>Type Three Data</u> - Status data.

APPENDIX B

MECHANIZED TRIANGULARIZATION AND REINDEXING

Triangularization

For this process, a choice is available between the use of a computer and use of conventional card equipment. Since the operation is essentially that of alternately sorting and collating, both of which are accomplished in computers only with a great attendant loss of efficiency, it is felt that the equipment of choice for problems of usual size is the conventional card equipment.

The detailed procedure, paraphrasing the manual algorithm, presumes as input manually prepared (but not necessarily ordered) lists of prime and deletion relation pairs (i,j) and concomitant relation n-tuples (i,j,k,...,n), together with their respective quantifiers n, -1, or (c).

1. Establish the maximum length of identification present in the lists, and establish a card format such that there are two fields, each the length of the maximum length of identification. A third field, usually two digits in length, should be established for recording the quantifiers.

2. For each of the prime and deletion relations, punch a card showing each pair of identifications (including the arbitrary i = 0 identification for input elements) together with the quantifier punched in the three designated fields. In the balance of the discussion of the procedure, the field associated with the first (ith) identification of the pair will be referred to as the "i field," the field associated

with the second (jth) identification of the pair as the "j field," and the quantifier as the "q field." The quantifier for prime relations will be punched as an unsigned numeric value. The quantifier for deletion relations will be punched as J (11 - 1 punch), where the 11 punch will be construed as negation.

3. For each of the concomitant relation n-tuples, punch the following cards, using the same format as established in step 1, and with the quantifier for all cards punched as A (12 - 1 punch):

a. Punch a card showing the first designation, i, of the n-tuple in the i field and the second designation j of the n-tuple in the j field, and the quantifier A in the q field. Also punch an 11 zone punch in column 80.

b. Punch cards, one for each designation beyond the second designation j, with the first designation of the n-tuple, i, in the j field, and the third or subsequent designations k,...,n in the i field, and the quantifier A in the q field.

As an example, the n-tuple K23 B10 J19 P14 would result in three cards being punched:

```
K23 B10 A              X (Col. 80)
J19 K23 A
P14 K23 A
```

All punching in steps 2 and 3 will have all fields right justified.

4. If the manual listings were not separate, separate the three lists by sorting on the units position of the q field. No zone indicates

prime relations, 11 zone indicates deletions, 12 zone indicates concomitant relations.

5. It is possible, in the practical case, that duplicate cards exist either through error, or through the fact that input elements used in more than one report have been identified twice. To eliminate duplicates, sort each of the three decks established in step 4 regarding the i and j fields as a single field, then check for duplicates on the collator.

6. Set aside, temporarily, the concomitant and deletion relation decks. Reproduce the prime relations deck, preferably on cards of a different color. Designate one of these prime relations decks as deck A, the other as deck B.

7. Sort the concomitant relations deck on column 80. Set aside temporarily the cards without an 11 punch in column 80. Designate the concomitant relations cards with a punch in column 80 as deck C.

8. Sort deck A with minor on the i field and major on the j field. Sort deck B with minor on the j field and major on the i field. Sort deck C with major on the j field.

9. On the collator, perform a three pocket match, with deck A in the primary feed, wired to read the j field, deck B in the secondary feed, wired to read the i field. Select unequal primaries. All basic set-up switches must be on except multi-S & S. The result of this operation will be the selection of all system elements which appear as a j designation only. On the first pass, this will be the highest level elements of the system. This step corresponds to step 2 of the manual algorithm.

10. Sort deck B with a minor on the i field and major on the j field. Only the major sort is physically required, due to the previous sort on this deck. Deck B is now in the same sequence as the original deck A.

11. On the collator, perform a three pocket match, with deck B in the primary feed, the cards selected from deck A in step 9 in the secondary feed, both feeds wired to read a combined i and j field. Select equal primaries. All basic set-up switches must be on except multi-S & S. This operation results in removing from deck B the cards corresponding to the cards selected from deck A in step 9. This step corresponds with step 3 of the manual algorithm. The highest system element cards have now been removed from both decks. The cards selected from deck B may be discarded.

12. On the collator, perform a two pocket match-merge with the deck C in the primary feed and the cards selected from deck A in step 9 in the secondary feed, both feeds wired to read the j field. Select equal secondaries to merge behind primaries. The result of this operation is to place concomitant relations adjacent to and behind the elements to which they pertain. The step performs the function of ordering of the concomitant relations done in the first part of step 5 of the manual algorithm.

13. Sort deck A with minor on the j field and major on the i field. Only the major sort is physically required, due to the previous sort on this deck.

14. Deck A and deck B now correspond to the reduced lists of the manual algorithm, except that the role of deck A and deck B have

been interchanged. Change the designation of deck A to deck B, and deck B to deck A, and repeat steps 9 through 14 until all three decks are exhausted, with the connotation in these steps that the reduced decks A, B, and C are being used, and that decks A and B interchange designation at the completion of each pass. At each pass, successively lower levels of systems elements will be removed.

If level designation is desired, the merged deck resulting at the end of step 12 can be gang punched with a suitable designation to indicate level. For the concomitant cards (11 punch in column 80), the level pertains to the designation in the i field, while for the prime relation cards, it pertains to the j field. Such designation is elective, since the algorithms to follow do not require this designation.

If on some pass step 9 results in no cards being selected, a cycle has been encountered, and the operation must be terminated in order to permit the manual determination of the cause of the cycle. The cause will be either a punching or other clerical error in transcription, or an error on the part of the analyst in defining the system relations.

15. The single deck resulting from the accumulation of decks produced on the several passes through step 12 is now reproduced, with selection to reproduce cards exactly in the absence of a punch in column 80, and interchange of the j and i fields in the presence of a punch in column 80. This performs the second portion of the manual step 5, that of inserting after j in the record the first index, i, of the n-tuple when $j' = j$.

At the completion of this step the triangularization, as such, has been completed. It is now necessary for computer operation of the balance of the technique to reindex the systems elements, and to take certain other steps for preparation of the input for the $S - S^*$ and $S' - S'^*$ algorithms.

Reindexing and Input Preparation

The essential purposes of the steps in this section are to reindex the systems elements in order to simplify the computation of the S^* and S'^* matrices, to recombine the decks representing the three categories of system relations, and to arrange the cards in proper order for input to the further algorithms.

The numbering scheme used for reindexing depends largely upon the computer to be used, the conventions used in addressing its memory, and the memory location which will be used as a location for the lower right hand corner of the matrix. Furthermore, two possibilities exist in this respect:

1. An absolute indexing system can be used, wherein the initial number assigned and the increment between assignments is determined based on the three considerations above.

2. A relative indexing system may be used, where the consecutive integers 1, 2, 3, ..., n are assigned, and the programs for the algorithms for specific machine configurations make a conversion within the computer from the relative system to a system which is absolute for the machine in question.

Regardless of the choice made, the following machine operations are applicable.*

1. Prepare a master deck with the desired numbering sequence punched in a convenient field. In addition, punch 11 in any convenient column, e.g., column 79, as a control punch.

2. Merge this master deck with the reproduced deck obtained in step 15 of the previous section.

3. Reproduce the old identification number (j field) into an appropriate field of the master deck. Punch control on the reproducer to punch on 11 in column 79.

4. Sort the master cards, now with the sequence number and old identification number, out of the combined deck based on the 11 punch in column 79. The non-master cards are of no further use.

5. Take now the deck which was the input to step 15 of the preceding section, that is, the deck from which the reproduction was made, the deletion relations deck which was previously laid aside, and the balance of the concomitant relations cards which do <u>not</u> have an 11 punch in column 80. Combine these cards into a single deck, hereinafter referred to as the "input deck."

6. Sort the input deck on the j field. Sort the master deck on the old identification field.

7. Merge the master deck with the input deck, comparing on the sort fields of step 6, and placing the master card <u>ahead</u> of each old

---

* A different, more efficient method can be used if a sequential numbering device is available on the reproducing punch. It has not been specified here, as this equipment is seldom available.

identification (j field).

8. Reproduce the new identification (index j) into the input deck in an appropriate field. Punch control to punch on non-11 in column 79.

9. Sort the master cards out of the combined deck, based on the presence of an 11 punch in column 79.

10. Re-sort the input deck on the i field. There will be a substantial number of cards with zero in the i field, and these cards may now be discarded.

11. Merge the master deck with the input deck, comparing on the sort fields of step 10, and placing the master card <u>ahead</u> of each old identification (i field). Some master cards will not find corresponding cards in the input deck.

12. Reproduce the new identification (index i) into the input deck in an appropriate field. Punch control to punch on non-11 in column 79.

13. Sort the master cards out of the combined deck, based on the presence of an 11 punch in column 79.

The reindexing is complete at this point. The old i and j fields will not be used in the computer operation as such, but must be preserved if it is desired to have the computer output in terms of the original identifications. In order to avoid confusion in the balance of the discussion relative to input preparation, the new index fields, whether reproduced into a new card or not, will be now referred to as the i and j fields.

It will be recalled that in the manual computation of the S^* and

S'^* matrices, the first cell of the S or S' matrix used was that which was the lowest in the right-most non-zero column. In the computer solution, the S and S' matrices are not formed in memory, but exist conceptually, represented by the input deck created in the immediately preceding steps. The row and column indices of the cell are represented by the i and j fields, respectively, and the cell value is in the q field.

At this point, therefore, due to the method of creating the input deck, it is in reverse order from that required by the computation. It must therefore be re-sorted into proper order; that is, so that the input will be read up columns, starting at the right-most non-zero column of the (conceptual) S or S' matrix. In addition, it is desirable to place a sequence number in each input card in order to guard against lost or out-of-sequence data in future operations. The next two steps perform this operation.

14. Re-sort the input deck, with minor on the i field (row index) and major on the j field (column index). This is a <u>reverse sort</u>. Remove cards from pockets of the sorter nines first, zeros last.

15. Sequence number the input deck, using any convenient field, and any standard sequencing technique.

The input deck is now ready to be used for the computation of the S^* and S'^* matrices.

APPENDIX C

COMPUTER PROGRAMS FOR SYSTEMS ANALYSIS

Introduction

To implement the computation of the basic algorithms, four computer programs have been written and tested. The first program computes the number of zero cells below the diagonal of S^* and above the first non-zero entry of a given column, and estimates an upper bound for the memory required for a given matrix. The other three programs perform the computation of the S^* and S'^* matrices, and furnish a printout of the non-zero matrix elements.

The program designations, their function, and the form and sequence of input and output is as follows:

Program 1. This program computes information relative to the number of zero cells between the diagonals and the highest non-zero cells in the columns of S^*. The number of such cells is indicated, and the program also indicates the upper bound on the number of cells required and the actual amount of memory space needed in the computer for program 2 as written. The input to this program is a lead card, the same used in program 2 and program 3, specifying the dimension of the S matrix. This is followed by the input cards obtained from the reindexing operation, re-sorted with major sort on the j field, minor sort on the i field.

Program 2. This program computes the S^* matrix, and has as output one card for each non-zero cell of S^*, including the diagonal cells. The input required is the triangularized, reindexed deck obtained from the operations detailed in Appendix B. In addition, a lead card is required to indicate the dimension of the S matrix; that is, the number of systems elements in the system. Deletion relation cards can be left in the input deck, as the program will merely pass them.

Program 3. This program computes the S'^{*} matrix. Input and output are the same as for program 2.

Program 4. This program takes the output of program 2 or program 3, finds the original designations of the systems elements, and prints the column designations of the columns of S^{*} or S'^{*}, the row designation of non-zero cells in the column (if any), and the value of the cell. The input is (1) the master deck from step 13 of the reindexing operation, Appendix B; (2) a lead card containing titling information for the print-out; and (3) the output cards from either program 2 or program 3, in the order of major sort on the j field, minor sort on the i field.

Principal Program Features

The main consideration in writing these programs was that of conserving memory in order to accommodate as large as possible a matrix. Since in many installations only a card type 1401 is available, the programs do not use tape in the execution of the programs, although it is required for compilation. However, programs compiled on a tape machine can be run on a card machine, so the compilation could be done by a service bureau, execution on the user's machine. All programs are written in 1401 Autocoder. If larger system matrices were required in a given case, programs 2 and 3 could be re-written to use either tape or disk memory.

Programs 2 and 3 are essentially alike, with program 3 differing in only a few instructions in order to handle the logical operations called for by the deletion relation. A lead card establishes the dimension of the matrix in both cases, where the dimension is the number of systems elements in the system under study. Program 2 ignores cards with the deletion relation. After the first data card is read, all of the cells of the columns with a column index greater than that shown on the first card are computed, since these columns all consist of merely the diagonal cell. They are also punched as computed. The program then follows the algorithm substantially as outlined in Chapter VI, except that columns are not extracted for the multiplication, nor is a vector V designated as such. Rather, the multiplication takes place a cell at a time and the product is added directly to the proper cell of the column being computed. Two multiplication routines are used, one for regular quantifiers on the input card (labeled REGMUL)*, and one for concomitant quantifiers (labeled CONCOM). Since most of the quantifiers encountered in practice will be either 1 or (1), these multiplication routines are actually addition routines, with the routine repeated the number of times indicated by the quantifier.

As each column is completed, it is punched, one card for each cell, starting at the bottom of the column. Zero cells are not punched. Zero cells between the top non-zero cell and the diagonal are not stored, but are counted, and the number of such cells is stored along

---

* Labels refer to the subroutine labeling on the program listings, Appendix D.

with the column number in a data word representing the diagonal cell. The diagonal 1 is not stored, but is inferred as needed by the equality of row and column index.

Each card, as it is read, requires searching memory for the proper column for multiplication. This is accomplished by first establishing the number of columns to be skipped, which is one less than the difference between the row and column index on the input card. Then, using the data word associated with the diagonal element of each column, the location of the next can be computed based on column index, matrix dimension, and the number of zero cells eliminated from the given column. When the proper number has been skipped, the address of the proper column is available.

In program 3, deletion cards force a machine character # (Pound) into the proper cell of the column being computed to prevent any further entries into that cell. Other cards for the same column require examination of each cell for the presence of # to avoid placing any other value there. When a column is complete, the cells containing # are changed to zero, and are therefore not punched.

The other change in program 3 is that if a column has any nonzero entries other than the diagonal, then the diagonal is not used in the computation. This is accomplished by recognition that columns with only the diagonal element have stored only the data word containing the column index and the number of zero cells which were eliminated.

Program 2 requires about 1650 positions of memory including the read, punch and print areas. Therefore, in a 1401 with 8K memory, 6350 positions are available for the matrix. Due to the zero cell elimination, storage requirements will differ with specific application. However, since about 112 elements can be accommodated without any matrix compression, it is estimated that this size machine will accommodate the matrix for an information system of about 150 elements. If a larger 1401 is available, the program can be modified to make use of the larger memory merely by changing the constant labeled HIGH in the program listings to a figure one less than the machine capacity. While program 3 requires about 1850 positions, less storage is needed for the S'^{*} matrix, so that program 2 is the guiding program.

Program 4 is a relatively straightforward print routine. The master cards used in reindexing the original input deck are used to set up a table in memory bearing both the original and the new indices. The output cards from either program 2 or 3 are then read, the old indices from the table associated with the new ones from the cards, and the indices and values printed out, seven per line, with the proper column index. Concomitant cells are indicated by an asterisk (*).

Program 1 is also straightforward, and computes the quantities shown in Chapter VII relative to the amount of computer space the matrix for a given system will require. This program is intended for use prior to program 2 and 3, if there is doubt about the amount of storage required. If the final quantity printed as a result is less than 6350, then a 1401 with 8K memory will accommodate the system matrix.

Appendix D shows complete program listings in 1401 Autocoder for these programs. Comments in the listings specify input and output card format, and indicate the purpose of major sections of the programs.

APPENDIX D

Table 27. Program 1—Compute Upper Bound of Matrix Size

```
                         COMPUTE UPPER BOUND OF MATRIX SIZE                         GHB-1               PAGE     1

SEQ PG LIN  LABEL  OP     OPERANDS                                            SFX CT  LOCN  INSTRUCTION TYPE  CARD

101  1 01   000    JOB    COMPUTE UPPER BOUND OF MATRIX SIZE
102  1 02          CTL    4401
103 99 99   *  INPUT CARD FIELDS - I-FIELD CC 23-25, J-FIELD CC 26-28
104 99 99   *  SORT IS J-FIELD MAJOR, I-FIELD MINOR. LEAD CARD HAS S-MATRIX
105 99 99   *  DIMENSION IN CC 1-3.
106  1 03          ORG    333                                                                 0333
107 99 99   *  SETS COUNTERS, READS LEAD CARD
108  1 04   BEGIN  MLCWA +0000,ZED=4                                              7  0333  L 772 776             4
109  1 05          MLCWA +001,JAY=3                                               7  0340  L 779 782             4
110  1 06          SW     1                                                       4  0347  , 001                 4
111  1 07          R                                                              1  0351  1                     4
112  1 08          ZA     3,ENN=3                                                 7  0352  + 003 785             4
113  1 09          SW     23,26                                                   7  0359  , 023 026             4
114  1 091         MLCWA +000,REALJ=3                                             7  0366  L 788 791             5
115  1 092         MLCWA '0',SWONE=1                                              7  0373  L 792 793             5
116 99 99   *  READS CARD, TESTS LAST CARD, TEST TO SEE IF THIS COLUMN HAS
117 99 99   *  BEEN COMPUTED.
118  1 10   READ   R                                                              1  0380  1                     5
119  1 11          BLC    SETSW                                                   5  0381  B 433 A               5
120  1 111         MLZS   +1,28                                                   7  0386  Y 794 028             5
121  1 12          C      28,REALJ                                                7  0393  C 028 791             5
122  1 13          BU     TEST                                                    5  0400  B 417 /               5
123  1 14          B      READ,SWONE,0                                            8  0405  B 380 793 0           6
124  1 15          B      REST                                                    4  0413  B 516                 6
125 99 999  *  DETERMINES WHETHER J-INDEX DENOTES NON-NULL OR NULL COLUMN
126  1 17   TEST   C      28,JAY                                                  7  0417  C 028 782             6
127  1 18          BE     JINL                                                    5  0424  B 444 S               6
128  1 19          B      JNOTL                                                   4  0429  B 491                 6
129 99 999  *  SETS SWITCH ON LAST CARD
130  2 01   SETSW  MLNS   +1,SWONE                                                7  0433  D 794 793             6
131  2 02          B      READ+6                                                  4  0440  B 386                 6
132 99 999  *  COMPUTES NUMBER OF ZERO CELLS BETWEEN DIAGONAL AND FIRST NON-
133 99 999  *  ZERO CELL IN NON-NULL COLUMN
134  2 04   JINL   MLC    28,REALJ                                                7  0444  M 028 791             7
135  2 05          A      +1,JAY                                                  7  0451  A 794 782             7
136  2 06          A      25,ZED                                                  7  0458  A 025 776             7
137  2 07          S      28,ZED                                                  7  0465  S 028 776             7
138  2 08          S      +1,ZED                                                  7  0472  S 794 776             7
139  2 09          B      READ,SWONE,0                                            8  0479  B 380 793 0           8
140  2 10          B      REST                                                    4  0487  B 516                 8
141 99 999  *  COMPUTES NUMBER OF ZERO CELLS IN NULL COLUMN
142  2 12   JNOTL  A      ENN,ZED                                                 7  0491  A 785 776             8
143  2 13          S      JAY,ZED                                                 7  0498  S 782 776             8
144  2 14          A      +1,JAY                                                  7  0505  A 794 782             8
145  2 15          B      TEST                                                    4  0512  B 417                 8
146 99 999  *  COMPUTES NUMBER OF ZERO CELLS IN NULL COLUMNS AFTER LAST
147 99 999  *  INPUT CARD
```

Table 27. Program 1—Compute Upper Bound of Matrix Size (Continued)

```
SEQ PG LIN  LABEL  OP     OPERANDS                                                  SFX CT  LOCN  INSTRUCTION TYPE  CARD

148  2 17   REST   C      JAY,ENN                                                        7  0516  C 782 765             9
149  2 18          BE     DOYOU                                                          5  0523  B 553 S               9
150  2 19          A      ENN,ZED                                                        7  0528  A 785 776             9
151  2 20          S      JAY,ZED                                                        7  0535  S 782 776             9
152  2 21          A      +1,JAY                                                         7  0542  A 794 782             9
153  2 22          B      REST                                                           4  0549  B 516                 9
154 99 999  *  COMPUTES AND PRINTS TOTAL ZERO CELLS Z, UPPER BOUND OF CELLS
155 99 999  *  IN MATRIX, U, AND NUMBER OF MEMORY POSITIONS NEEDED.
156  3 01   DOYOU  ZA     ENN,PROD-4                                                     7  0553  + 785 744            10
157  3 02          M      ENN,PROD                                                       7  0560  ' 785 748            10
158  3 03          MLCWA  PROD,SAVE-7                                                    7  0567  L 748 801            10
159  3 04          A      ENN,PROD                                                       7  0574  A 785 748            10
160  3 05          ZA     PROD,BPROD-2                                                   7  0581  + 748 739            10
161  3 06          M      +5,BPROD                                                       7  0588  ' 802 741            11
162  3 07          S      ZED,BPROD-1                                                    7  0595  S 776 740            11
163  3 08          CS     332                                                            4  0602  / 332                11
164  3 09          CS                                                                    1  0606  /                    11
165  3 10          MLCWA  'NUMBER OF ZERO CELLS Z IS',230                                7  0607  L 827 230            11
166  3 11          MLCWA  EDITA,235                                                      7  0614  L 752 235            11
167  3 12          MCE    ZED,235                                                        7  0621  E 776 235            12
168  3 13          W                                                                     1  0628  2                    12
169  3 14          CC     K                                                              2  0629  F K                  12
170  3 141         CS     299                                                            4  0631  / 299                12
171  3 15          MLCWA  'UPPER BOUND U IS',220                                         7  0635  L 843 220            12
172  3 16          MLCWA  EDITB,230                                                      7  0642  L 761 230            12
173  3 161         MCE    BPROD-1,230                                                    7  0649  E 740 230            12
174  3 17          W                                                                     1  0656  2                    13
175  3 18          CC     K                                                              2  0657  F K                  13
176  3 19          CS     299                                                            4  0659  / 299                13
177  3 20          A      ENN,SAVE                                                       7  0663  A 785 801            13
178  3 21          A      ENN,SAVE                                                       7  0670  A 785 801            13
179  3 22          A      ENN,SAVE                                                       7  0677  A 785 801            13
180  3 23          S      ZED,SAVE                                                       7  0684  S 776 801            13
181  3 24          S      ZED,SAVE                                                       7  0691  S 776 801            14
182  3 25          MLCWA  'AMOUNT OF 1401 STORAGE NEEDED IS',237                         7  0698  L 875 237            14
183  4 01          MLCWA  EDITC,245                                                      7  0705  L 768 245            14
184  4 02          MCE    SAVE,245                                                       7  0712  E 801 245            14
185  4 03          MLCWA  'POSITIONS FOR PROGRAM NO 1',272                               7  0719  L 901 272            14
186  4 04          W                                                                     1  0726  2                    14
187  4 05          CC     L                                                              2  0727  F L                  14
188  4 06          H      BEGIN                                                          4  0729  . 333                15
189 99 999  *  CONSTANTS AND SYMBOLS
190  5 12   BPROD  DCW    =9                                                             9  0741                       15
191  5 13   PROD   DCW    =7                                                             7  0748                       15
192  5 14   EDITA  DCW    '     0'                                                       4  0752                       15
193  5 15   EDITB  DCW    '              0'                                              9  0761                       15
194  5 16   EDITC  DCW    '          '                                                   7  0768                       16
                   DCW    +0000                                                          4  0772             LIT       16
        195 ZED           =04                                                            4  0775             AREA      16
                          +001                                                           3  0779             LIT       16
```

Table 27. Program 1—Compute Upper Bound of Matrix Size (Continued)

```
SEQ PG LIN  LABEL   OP     OPERANDS                                            SFX CT  LOCN  INSTRUCTION TYPE   CARD

       109  JAY            =03                                                      3  0782              AREA     16
       112  LMN            =03                                                      3  0785              AREA     16
                           +000                                                     3  0788              LIT      16
       114  REALJ          =03                                                      3  0791              AREA     17
                           '0'                                                      1  0792              LIT      17
       115  SWONE          =01                                                      1  0793              AREA     17
                           +1                                                       1  0794              LIT      17
       158  SAVE           =07                                                      7  0801              AREA     17
                           +5                                                       1  0802              LIT      17
       165                 'NUMBER OF ZERO CELLS Z IS'                             25  0827              LIT      17
       171                 'UPPER BOUND U IS'                                      16  0843              LIT      18
       182                 'AMOUNT OF 1401 STORAGE NEEDED IS'                      32  0875              LIT      19
       185                 'POSITIONS FOR PROGRAM NO 1'                            26  0901              LIT      20
195  5 17           END    BEGIN                                                            / 333 080          21
```

Table 28. Test Input—Program 1

```
018        TEST PRUGRAM
         W3        P101011001                                    0031
         C5        R20A003002                                    0030
         C4        R20A004002                                    0029
         R1        R205005002                                    0028
         C3        R20J006002                                    0027
         C2        R20J007002                                    0026
         W4        R201008002                                    0025
         C3        C50A006003                                    0024
         C2        C40A007004                                    0023
         C3        R10A006005                                    0022
         C2        R10A007005                                    0021
         W3        R101011005                                    0020
         C1        R10J012005                                    0019
         D1        R10J015005                                    0018
         D3        R10J017005                                    0017
         C2        C30A007006                                    0016
         C1        C20A012007                                    0015
         H1        W401009008                                    0014
         H2        W401010008                                    0013
         D2        W401016008                                    0012
         C1        W30A012011                                    0011
         W1        W301013011                                    0010
         W2        W301014011                                    0009
         S1        W30J018011                                    0008
         D3        C10A017012                                    0007
         S1        C10A018012                                    0006
         D1        W101015013                                    0005
         D2        W101016013                                    0004
         D3        W101017013                                    0003
         D1        W201015014                                    0002
         S1        W201016014                                    0001
```

Table 29. Test Output--Program 1

```
 NUMBER OF ZERO CELLS Z IS   45

UPPER BOUND U IS       126

 AMOUNT OF 1401 STORAGE NEEDED IS     288 POSITIONS FOR PROGRAM NO 1
```

Table 30. Program 2--Compute Composed of Analysis Matrix

```
SEQ PG LIN  LABEL  OP     OPERAND                                                  SFX CT  LOCN  INSTRUCTION TYPE  CARD

101  1 01   000    JOB    COMPUTE COMPOSED OF ANALYSIS MATRIX
102  1 02          CTL    4401
103 99 99   *  INPUT CARD FIELDS- G-FIELD CC 21-22, I-FIELD CC 23-25, J-FIELD
104 99 99   *  CC 26-28, CARD SEQUENCE NUMBER CC 76-79. SORT IS A REVERSE SORT
105 99 99   *  J-FIELD MAJOR, I-FIELD MINOR
106 99 99   *  PROGRAM STARTS AT SETUP, WHICH IS IN PUNCH AREA AND IS CLEARED
107 99 99   *  AFTER USE. ANY RESTART EXCEPT SEQUENCE ERROR REQUIRES RELOADING
108 99 99   *  OF PROGRAM. LEAD CARD HAS MATRIX SIZE IN CC 1-3
109 99 99   *  READS LEAD CARD, SETS COUNTERS
110 11 01          ORG    81                                                                      0081
111 11 02          CS     199                                                       4     0081   / 199                4
112 11 021         MLCWA  '999',1540                                                7     0085   L V21 W40            4
113 11 03          B      READ                                                      4     0092   B 333                4
114 11 04          ORG    101                                                                     0101
115 11 05   SETUP  CS     80                                                        4     0101   / 080                5
116 11 06          SW     1,11                                                      7     0105   , 001 011            5
117 11 07          R                                                                1     0112   1                    5
118 11 08          ZA     3,COLNO                                                   7     0113   + 003 094            5
119 11 09          SW     21,23                                                     7     0120   , 021 023            5
120 11 10          SW     26,76                                                     7     0127   , 026 076            5
121 11 11          ZA     +1,SEQ                                                    7     0134   + V22 V07            6
122 11 12          MLC    COLNO,DIMEN                                               7     0141   M 094 V10            6
123 11 13          MLNS   +1,SW1                                                    7     0148   D V22 V18            6
124 11 14          MLNS   +0,SW4                                                    7     0155   D V23 V50            6
125 11 15          B      81                                                        4     0162   B 081                6
126 99 99   *  SETS ALL COLUMNS WITH INDEX GREATER THAN THAT OF FIRST DATA
127 99 99   *  CARD READ
128 12 01          ORG    201                                                                     0201
129 12 02   EARLY  MLC    HIGH,EARLY+13                                             7     0201   M 091 214            7
130 12 03          MLCWA  GAP,0                                                     7     0208   L 097 000            7
131 12 04          MLC    HIGH,*+14                                                 7     0215   M 091 235            7
132 12 05          MA     THREEN,*+7                                                7     0222   = V00 235            7
133 12 06          MLCWA  COLNO,0                                                   7     0229   L 094 000            7
134 12 07          MLCWA  COLNO,106                                                 7     0236   L 094 106            8
135 12 071         MLZS   '0',106                                                   7     0243   Y V24 106            8
136 12 08          B      PDIAG                                                     4     0250   B #92                8
137 12 09   STEPN  S      +1,COLNO                                                  7     0254   S V22 094            8
138 12 10          C      28,COLNO                                                  7     0261   C 028 094            8
139 12 11          BE     RESET                                                     5     0268   B 291 S              8
140 12 12          A      +1,GAP                                                    7     0273   A V22 097            9
141 12 13          MA     SIXN,HIGH                                                 7     0280   = V03 091            9
142 12 14          B      EARLY                                                     4     0287   B 201                9
143 12 15   RESET  MLNS   +0,SW1                                                    7     0291   D V23 V18            9
144 12 16          B      COLSET                                                    4     0298   B 177                9
145  1 03          ORG    333                                                                     0333
146 99 99   *  THIS SECTION READS DATA CARD, PERFORMS SEQUENCE CHECK, AND
147 99 99   *  PASSES SELECTION RELATION CARD. TEST IS CONTROL BREAK ON J-INDEX
```

Table 30. Program 2—Compute Composed of Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP     OPERANDS                                        SFX CT  LOCN  INSTRUCTION TYPE  CARD

148 99 99   *  CARD ESTABLISHES NUMBER OF COLUMNS TO SKIP IN SEARCH FOR PROPER
149 99 99   *  COLUMN FOR MULTIPLICATION. SLIDE FINDS THE COLUMN.
150  1 04   READ   R                                                           1  0333  1                  10
151  1 041         MLZS  +0,79                                                 7  0334  Y V23 079          10
152  1 05          C     79,SEQ                                                7  0341  C 079 V07          10
153  1 06          BU    MIXED                                                 5  0346  B 747 /            10
154  1 07          A     +1,SEQ                                                7  0353  A V22 V07          10
155  1 071         MLZS  +0,26                                                 7  0360  Y V23 028          10
156  1 08          BWZ   READ,22,K                                             8  0367  V 333 022 K        11
157  1 09   CARD   C     28,COLNO                                              7  0375  C 028 094          11
158  1 10          BH    CLOSE                                                 5  0382  B 785 U            11
159  1 11          MLC   HIGH,SEARCH=3                                         7  0387  M U91 V27          11
160  1 12          MA    SIX,SEARCH                                            7  0394  = V16 V27          11
161  1 13          MLCWA 25,LOOK=3                                             7  0401  L 025 V30          12
162  1 14          S     28,LOOK                                               7  0408  S 028 V30          12
163  1 15          S     +1,LOOK                                               7  0415  S V22 V30          12
164  1 16          MLCWA COLEN,BRL=3                                           7  0422  L W20 V33          12
165  1 17          A     +2,BRL                                                7  0429  A V34 V33          12
166  1 18   SLIDE  MLCWA BRL,ARL=3                                             7  0436  L V33 V37          13
167  1 19          C     +000,LOOK                                             7  0443  C V40 V30          13
168  1 20          BE    THIS                                                  5  0450  B 505 S            13
169  1 21          MLC   SEARCH,*+4                                            7  0455  M V27 465          13
170  1 22          S     0,ARL                                                 7  0462  S 000 V37          13
171  1 221         MLZS  '0',ARL                                               7  0469  Y V24 V37          14
172  1 23          MA    ARL                                                   4  0476  = V37              14
173  1 24          MA    ARL,SEARCH                                            7  0480  = V37 V27          14
174  1 25          S     +1,LOOK                                               7  0487  S V22 V30          14
175  2 01          S     +1,BRL                                                7  0494  S V22 V33          14
176  2 02          B     SLIDE                                                 4  0501  B 436              14
177 99 99   *  FINDS BOTTOM OF COLUMN TO BE MULTIPLIED.
178  2 03   THIS   MLCWA DIMEN,NOML=3                                          7  0505  L V10 V43          15
179  2 04          MLC   SEARCH,*+11                                           7  0512  M V27 529          15
180  2 05          MA    THREEN,*+4                                            7  0519  = V00 529          15
181  2 06          S     0,NOML                                                7  0526  S 000 V43          15
182  2 07          MLCWA NOML,ARL                                              7  0533  L V43 V37          15
183  2 08          MLC   SEARCH,*+4                                            7  0540  M V27 550          16
184  2 09          S     0,ARL                                                 7  0547  S 000 V37          16
185  2 091         MLZS  '0',ARL                                               7  0554  Y V24 V37          16
186  2 10          MA    ARL                                                   4  0561  = V37              16
187  2 11          MLCWA SEARCH,TRACE=3                                        7  0565  L V27 V46          16
188  2 12          MA    ARL,TRACE                                             7  0572  = V37 V46          16
189  2 13          MLCWA HIGH,FOLLOW=3                                         7  0579  L U91 V49          17
190  2 14          BWZ   CONCOM,22,B                                           8  0586  V 849 022 B        17
191  2 15          B     REGMUL                                                4  0594  B 618              17
192 99 99   *  TEST FOR LAST CARD
193  2 16   REMAIN BLC   LAST                                                  5  0596  B 607 A            17
194  2 17          B     READ                                                  4  0603  B 333              17
195 99 99   *  S IS SWITCH ON LAST CARD
196  2 18   LAST   MLNS  1,SW+1                                                7  0607  D C01 V50          17
197  2 19          B     CLOSE                                                 4  0614  B 785              17
```

Table 30. Program 2—Compute Composed of Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP    OPERANDS                                                          SFX CT  LOCN  INSTRUCTION TYPE  CARD

198 99 99   *  MULTIPLICATION FOR REGULAR NUMERAL ON INPUT CARD. BACK IS A
199 99 99   *  REENTRY POINT FOR HOLDZ.
200 99 99   *  COMP IS A REENTRY POINT FOR ZZONE AND REPLC
201  3 01   REGMUL ZA    22,CTR2=2                                                              7  0618  + 022 V52           18
202  3 02          S     +1,CTR2                                                                7  0625  S V22 V52           18
203  3 03          C     TRACE,SEARCH                                                           7  0632  C V46 V27           18
204  3 04          BE    DIAG                                                                   5  0639  B 751 S             18
205  3 05          MLC   FOLLOW,*+7                                                             7  0644  M V49 657           18
206  3 06          BWZ   HOLDZ,0,S                                                              8  0651  V 813 000 S         19
207  3 08   BACK   MLC   TRACE,*+11                                                             7  0659  M V46 676           19
208  3 09          MLC   FOLLOW,*+7                                                             7  0666  M V49 679           19
209  3 10          A     0,0                                                                    7  0673  A 000 000           19
210  3 11          BCE   REPLC,SW3,1                                                            8  0680  B 824 V58 1         19
211  3 12          MLC   TRACE,*+7                                                              7  0688  M V46 701           20
212  3 13          BWZ   ZZONE,0,S                                                              8  0695  V 733 000 S         20
213  3 14   COMP   C     +00,CTR2                                                               7  0703  C V54 V52           20
214  3 15          BU    REGMUL+7                                                               5  0710  B 625 /             20
215  3 16          MA    TWON,TRACE                                                             7  0715  = V13 V46           20
216  3 17          MA    TWON,FOLLOW                                                            7  0722  = V13 V49           21
217  3 18          B     REGMUL                                                                 4  0729  B 618               21
218 99 99   *  TAGS A CELL AS CONCOMITANT QUANTIFIER
219  3 20   ZZONE  MLC   FOLLOW,*+7                                                             7  0733  M V49 746           21
220  3 21          MLZS  SLASH,0                                                                7  0740  Y V17 000           21
221  3 22          B     COMP                                                                   4  0747  B 703               21
222 99 99   *  FINDS NUMBER OF NON-STORED ZERO CELLS IN COLUMN, SETS ADDRESS FG
223 99 99   *  FORCING IN OF MULTIPLICATION OF DIAGONAL
224  4 01   DIAG   MLC   SEARCH,*+4                                                             7  0751  M V27 761           21
225  4 02          ZA    0,CTR3=3                                                               7  0758  + 000 V57           22
226  4 04          C     +000,CTR3                                                              7  0765  C V40 V57           22
227  4 05          BE    INS                                                                    5  0772  B 795 S             22
228  4 051         S     +1,CTR3                                                                7  0777  S V22 V57           22
229  4 06          MA    TWON,FOLLOW                                                            7  0784  = V13 V49           22
230  4 07          B     DIAG+14                                                                4  0791  B 765               22
231 99 99   *  MULTIPLICATION OF DIAGONAL
232  4 08   INS    MLC   FOLLOW,*+7                                                             7  0795  M V49 808           23
233  4 09          A     22,0                                                                   7  0802  A 022 000           23
234  4 10          B     REMAIN                                                                 4  0809  B 594               23
235 99 99   *  SETS SWITCH TO INDICATE CONCOMITANT CELL IN COLUMN
236  4 12   HOLDZ  MLNS  +1,SW3=1                                                               7  0813  D V22 V58           23
237  4 13          B     BACK                                                                   4  0820  B 659               23
238 99 99   *  TAGS A CELL AS A CONCOMITANT QUANTIFIER, RESETS A SWITCH
239  4 14   REPLC  MLC   FOLLOW,*+7                                                             7  0824  M V49 837           23
240  4 15          MLZS  SLASH,0                                                                7  0831  Y V17 000           24
241  4 16          MLNS  +0,SW3                                                                 7  0838  D V23 V58           24
242  4 17          B     COMP                                                                   4  0845  B 703               24
243 99 99   *  MULTIPLICATION FOR A CONCOMITANT QUANTIFIER ON INPUT CARD
244 99 99   *  ZCOIN FORCES A ZERO WHEN CONCOMITANTS ARE MULTIPLIED TOGETHER
245  5 01   CONCOM ZA    22,CTR2                                                                7  0849  + 022 V52           24
246  5 02          S     +1,CTR2                                                                7  0856  S V22 V52           24
247  5 03          C     TRACE,SEARCH                                                           7  0863  C V46 V27           24
```

Table 30. Program 2—Compute Composed of Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP     OPERANDS                                                   SFX CT  LOCN  INSTRUCTION TYPE  CARD

248  5 04          BE     CONDIA                                                          5  0870  B 974 S              25
249  5 041         MLC    TRACE,*+7                                                       7  0875  M V46 898            25
250  5 042         C      +00,0                                                           7  0882  C V54 000            25
251  5 043         BE     ZEDIN                                                           5  0889  B 956 S              25
252  5 05          MLC    TRACE,*+7                                                       7  0894  M V46 907            25
253  5 06          BWZ    ZEDIN,0,S                                                       8  0901  V 956 000 S          25
254  5 07          MLC    TRACE,*+11                                                      7  0909  M V46 926            26
255  5 08          MLC    FOLLOW,*+7                                                      7  0916  M V49 929            26
256  5 09          A      0,0                                                             7  0923  A 000 000            26
257  5 10          MLC    FOLLOW,*+7                                                      7  0930  M V49 943            26
258  5 11          MLZS   SLASH,0                                                         7  0937  Y V17 000            26
259  5 12          C      +00,CTR2                                                        7  0944  C V54 V52            27
260  5 13          BU     CONCOM+7                                                        5  0951  B 856 /              27
261  5 14   ZEDIN  MA     TWON,TRACE                                                      7  0956  = V13 V46            27
262  5 15          MA     TWON,FOLLOW                                                     7  0963  = V13 V49            27
263  5 16          B      CONCOM                                                          4  0970  B 849                27
264 99 99   *  FINDS NUMBER OF NON-STORED ZERO CELLS - CONCOMITANT MULT
265  5 18   CONDIA MLC    SEARCH,*+4                                                      7  0974  M V27 984            27
266  5 19          ZA     0,CTR3                                                          7  0981  + 000 V57            28
267  5 21          C      +000,CTR3                                                       7  0988  C V40 V57            28
268  5 22          BE     CONINS                                                          5  0995  B *18 S              28
269  5 221         S      +1,CTR3                                                         7  1000  S V22 V57            28
270  5 23          MA     TWON,FOLLOW                                                     7  1007  = V13 V49            28
271  5 24          B      CONDIA+14                                                       4  1014  B 988                28
272 99 99   *  MULTIPLICATION OF DIAGONAL - CONCOMITANT
273  6 01   CONINS MLC    FOLLOW,*+7                                                      7  1018  M V49 *31            29
274  6 02          A      22,0                                                            7  1025  A 022 000            29
275  6 03          MLC    FOLLOW,*+7                                                      7  1032  M V49 *45            29
276  6 04          MLZS   SLASH,0                                                         7  1039  Y V17 000            29
277  6 05          B      REMAIN                                                          4  1046  B 598                29
278 99 99   *  FORMS AND STORES DATA WORD IN PLACE OF DIAGONAL. CONTAINS COLUMN
279 99 99   *  INDEX AND NUMBER OF ZERO CELLS ELIMINATED
280  7 01   SETDIG MLC    TOP,*+7                                                         7  1050  M W08 *63            29
281  7 02          MLCWA  GAP,0                                                           7  1057  L U97 000            30
282  7 03          MA     THREEN,TOP                                                      7  1064  = V00 W08            30
283  7 04          MLC    TOP,*+7                                                         7  1071  M W08 *84            30
284  7 05          MLCWA  COLNO,0                                                         7  1078  L U94 000            30
285  7 06          MLC    BOTTOM,HIGH                                                     7  1085  M W11 U91            30
286 99 99   *  PUNCHES DIAGONAL CELL CARD ON LAST COLUMN  GIVES JOB DONE
287 99 99   *  MESSAGE AND HALTS
288  7 08   PDIAG  MLCWA  106,103                                                         7  1092  L 106 103            31
289  7 09          MLCWA  '01',108                                                        7  1099  L V60 108            31
290  7 10          P                                                                      1  1106  4                    31
291  7 11          B      STEPR,SW1,1                                                     8  1107  B 254 V18 1          31
292  7 12          S      +1,COLNO                                                        7  1115  S V22 U94            31
293  7 13          B      COLSET,SW4,0                                                    8  1122  B 177 V50 0          31
294  7 131         CS     332                                                             4  1130  / 332                32
295  7 132         CS                                                                     1  1134  /                    32
296  7 133         MLC    'JOB DONE',208                                                  7  1135  M V68 208            32
297  7 134         W                                                                      1  1142  2                    32
```

Table 30. Program 2—Compute Composed of Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP    OPERANDS                                              SFX CT  LOCN  INSTRUCTION TYPE  CARD

298  7 135         H     *-4                                                        4  1143  . 742                32
299 99 99   *  ERROR ROUTINE FOR SEQUENCE CHECK
300  7 14   MIXED  CS    332                                                        4  1147  / 332                32
301  7 15          CS                                                               1  1151  /                    32
302  7 16          MLC   'SEQUENCE-NEXT SB       FIX AND RESTART',237               7  1152  M W05 237            33
303  7 161         MLZS  '0',SEQ                                                    7  1159  Y V24 V07            33
304  7 17          MLC   SEQ,221                                                    7  1166  M V07 221            33
305  7 171         MLZS  +1,SEQ                                                     7  1173  Y V22 V07            33
306  7 18          W                                                                1  1180  2                    33
307  7 19          H     READ                                                       4  1181  . 333                33
308 99 99   *  ON COLUMN CONTROL BREAK, FINDS ENDS OF COLUMN
309  8 01   CLOSE  B     EARLY,SW1,1                                                8  1185  B 201 V18 1          34
310  8 02          MLC   CELLS+6,TOP=3                                              7  1193  M U18 W08            34
311  8 03          MLC   HIGH,BOTTOM=3                                              7  1200  M U91 W11            34
312  8 04          MLC   COLNO,106                                                  7  1207  M U94 106            34
313  8 041         MLZS  '0',106                                                    7  1214  Y V24 106            34
314  8 05          MLC   DIMEN,ROWNO=3                                              7  1221  M V10 W14            35
315  8 06          MLCWA +000,GAP                                                   7  1228  L V40 U97            35
316 99 99   *  ELIMINATES ZERO CELLS UNDER DIAGONAL
317  8 07   HERE   MLC   TOP,*+7                                                    7  1235  M W08 S48            35
318  8 08          C     +00,0                                                      7  1242  C V54 000            35
319  8 09          BU    PUNCH                                                      5  1249  B S72 /              35
320  8 10          MA    '002',TOP                                                  7  1254  = W17 W08            36
321  8 11          A     +1,GAP                                                     7  1261  A V22 U97            36
322  8 12          B     HERE                                                       4  1268  B S35                36
323 99 99   *  PUNCH ROUTINE, EXCEPT DIAGONAL.   ZERO CELLS NOT PUNCHED
324 99 99   *  STEP CHANGES ADDRESSES
325  8 13   PUNCH  MA    TWON,TOP                                                   7  1272  = V13 W08            36
326  8 131         C     TOP,BOTTOM                                                 7  1279  C W08 W11            36
327  8 132         BE    SETDIG                                                     5  1286  B *50 S              36
328  8 14          MLC   BOTTOM,*+7                                                 7  1291  M W11 T04            37
329  8 15          C     +00,0                                                      7  1298  C V54 000            37
330  8 16          BE    STEP                                                       5  1305  B T47 S              37
331  8 17          MLCWA ROWNO,103                                                  7  1310  L W14 103            37
332  8 171         MLZS  '0',103                                                    7  1317  Y V24 103            37
333  8 18          MLC   BOTTOM,*+4                                                 7  1324  M W11 T34            38
334  8 19          MLCWA 0,108                                                      7  1331  L 000 108            38
335  8 191         BWZ   SWEEP,108,B                                                8  1338  V U78 108 B          38
336  8 20          P                                                                1  1346  4                    38
337  8 21   STEP   MA    TWON,BOTTOM                                                7  1347  = V13 W11            38
338  8 22          C     TOP,BOTTOM                                                 7  1354  C W08 W11            38
339  8 23          BE    SETDIG                                                     5  1361  B *50 S              39
340  8 24          S     +1,ROWNO                                                   7  1366  S V22 W14            39
341  8 25          B     PUNCH+7                                                    4  1373  B S79                39
342 99 99   *  FINDS NUMBER OF CELLS IN NEXT COLUMN TO BE COMPUTED.  CELLS
343 99 99   *  SETS EACH CELL TO ZERO, ALSO TESTS TO KEEP FROM STORING TOO
344 99 99   *  LARGE A MATRIX.
345  9 01   COLSET MLC   DIMEN,COLLN=3                                              7  1377  M V10 W20            39
346  9 02          S     COLNO,COLLN                                                7  1384  S U94 W20            39
347  9 03          MLC   +000,CTR1=3                                                7  1391  M V40 W23            39
```

Table 30. Program 2—Compute Composed of Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP     OPERANDS                                              SFX CT  LOCN  INSTRUCTION TYPE  CARD

348  9 04          MA     SIXN,HIGH                                                  7  1398  = V03 U91            40
349  9 05          MLC    HIGH,CELLS+6                                               7  1405  M U91 U18            40
350  9 06   CELLS  MLCWA  +00,0                                                      7  1412  L V54 000            40
351  9 07          A      +1,CTR1                                                    7  1419  A V22 W23            40
352  9 08          C      CTR1,COLEN                                                 7  1426  C W23 W20            40
353  9 09          BE     CARD                                                       5  1433  B 375 S              41
354  9 10          MA     TWON,CELLS+6                                               7  1438  = V13 U18            41
355  9 11          C      '999',1640                                                 7  1445  C V21 W40            41
356  9 12          BU     OVRFLO                                                     5  1452  B U61 /              41
357  9 13          B      CELLS                                                      4  1457  B U12                41
358 99 99   *  ERROR ROUTINE IF MEMORY CAPACITY IS EXCEEDED
359  9 15   OVRFLO CS     332                                                        4  1461  / 332                41
360  9 16          CS                                                                1  1465  /                    41
361  9 17          MLC    'MEMORY FULL',211                                          7  1466  M W34 211            42
362  9 18          W                                                                 1  1473  2                    42
363  9 19          H      *-4                                                        4  1474  . U73                42
364 99 99   *  CLEARS PLUS SIGN FROM REGULAR CELL
365  9 26   SWEEP  MLZS   '0',108                                                    7  1478  Y V24 108            42
366  9 27          B      STEP-1                                                     4  1485  B T46                42
367 99 99   *  CONSTANTS AND SYMBOLS
368 10 01   HIGH   DSA    7999                                                       3  1491  I9Z                  42
369 10 02   COLNO  DCW    +000                                                       3  1494                       42
370 10 03   GAP    DCW    +000                                                       3  1497                       43
371 10 04   THREEN DSA    15997                                                      3  1500  I9G                  43
372 10 05   SIXN   DSA    15994                                                      3  1503  I9D                  43
373 10 06   SEQ    DCW    +0000                                                      4  1507                       43
374 10 07   DIMEN  DCW    +000                                                       3  1510                       43
375 10 08   TWON   DSA    15998                                                      3  1513  I9H                  43
376 10 10   SIX    DSA    006                                                        3  1516  006                  43
377 10 11   SLASH  DC     '/'                                                        1  1517                       44
378 10 12   SW1    DCW    '0'                                                        1  1518                       44
                   DCW    '999'                                                      3  1521              LIT      44
                          +1                                                         1  1522              LIT      44
                          +0                                                         1  1523              LIT      44
                          '0'                                                        1  1524              LIT      45
        159 SEARCH        =03                                                        3  1527              AREA     45
        161 LOOK          =03                                                        3  1530              AREA     45
        164 BRL           =03                                                        3  1533              AREA     45
                          +2                                                         1  1534              LIT      45
        166 ARL           =03                                                        3  1537              AREA     45
                          +000                                                       3  1540              LIT      45
        178 HOME          =03                                                        3  1543              AREA     46
        187 TRACE         =03                                                        3  1546              AREA     46
        189 FOLLOW        =03                                                        3  1549              AREA     46
        196 SW4           =01                                                        1  1550              AREA     46
        201 CTR2          =02                                                        2  1552              AREA     46
                          +00                                                        2  1554              LIT      46
        225 CTR3          =03                                                        3  1557              AREA     46
        236 SW3           =01                                                        1  1558              AREA     47
                          '01'                                                       2  1560              LIT      47
```

Table 30. Program 2—Compute Composed of Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL   OP     OPERANDS                                        SFX CT  LOCN  INSTRUCTION TYPE  CARD

       296                 'JOB DONE'                                            8  1568              LIT    47
       302                 'SEQUENCE-NEXT SB       FIX AND RESTART'             37  1605              LIT    48
       310  TOP            =03                                                   3  1608              AREA   49
       311  BOTTOM         =03                                                   3  1611              AREA   49
       314  ROWNO          =03                                                   3  1614              AREA   49
                           '002'                                                 3  1617              LIT    49
       345  COLEN          =03                                                   3  1620              AREA   49
       347  CTR1           =03                                                   3  1623              AREA   49
       361                 'MEMORY FULL'                                        11  1634              LIT    49
379 12 17           END    SETUP                                                        / 101 080             50
```

Table 31. Program 3—Compute Contained in Analysis Matrix

```
SEQ PG LIN  LABEL  OP    OPERANDS                      SFX CT  LOCN  INSTRUCTION TYPE  CARD

101  1 01   000    JOB   COMPUTE CONTAINED IN ANALYSIS MATRIX
102  1 02          CTL   4401
103 99 99   *  INPUT CARD FIELDS- Q-FIELD CC 21-22, I-FIELD CC 23-25, J-FIELD
104 99 99   *  CC 26-28, CARD SEQUENCE NUMBER CC 76-79. SORT IS A REVERSE SORT
105 99 99   *  J-FIELD MAJOR, I-FIELD MINOR
106 99 99   *  PROGRAM STARTS AT SETUP, WHICH IS IN PUNCH AREA AND IS CLEARED
107 99 99   *  AFTER USE. ANY RESTART EXCEPT SEQUENCE ERROR REQUIRES RELOADING
108 99 99   *  OF PROGRAM. LEAD CARD HAS MATRIX SIZE IN CC 1-3
109 99 99   *  READS LEAD CARD, SETS COUNTERS
110 11 01          ORG   81                                           0081
111 11 02          CS    199                              4  0081  / 199                4
112 11 021         MLCWA '999',1350                       7  0085  L X20 Y50            4
113 11 03          B     READ                             4  0092  B 333                4
114 11 04          ORG   101                                          0101
115 11 05   SETUP  CS    80                               4  0101  / 080                5
116 11 06          SW    1,11                             7  0105  , 001 011            5
117 11 07          R                                      1  0112  1                    5
118 11 08          ZA    3,COLNO                          7  0113  + 003 W92            5
119 11 09          SW    21,23                            7  0120  , 021 023            5
120 11 10          SW    26,76                            7  0127  , 026 076            5
121 11 11          ZA    +1,SEQ                           7  0134  + X21 X05            6
122 11 12          MLC   COLNO,DIMEN                      7  0141  M W92 X08            6
123 11 13          MLNS  +1,SW1                           7  0148  D X21 X16            6
124 11 14          MLNS  +0,SW4                           7  0155  D X22 X49            6
125 11 15          B     81                               4  0162  B 081                6
126 99 99   *  SETS ALL COLUMNS WITH INDEX GREATER THAN THAT OF FIRST DATA
127 99 99   *  CARD READ
128 12 01          ORG   201                                          0201
129 12 02   EARLY  MLC   HIGH,EARLY+13                    7  0201  M W89 214            7
130 12 03          MLCWA GAP,0                            7  0208  L W95 000            7
131 12 04          MLC   HIGH,*+14                        7  0215  M W89 235            7
132 12 05          MA    THREEN,*+7                       7  0222  = W98 235            7
133 12 06          MLCWA COLNO,0                          7  0229  L W92 000            7
134 12 07          MLCWA COLNO,106                        7  0236  L W92 106            8
135 12 071         MLZS  '0',106                          7  0243  Y X23 106            8
136 12 08          D     PDIAG                            4  0250  B S24                8
137 12 09   STEPR  S     +1,COLNO                         7  0254  S X21 W92            8
138 12 10          C     28,COLNO                         7  0261  C 028 W92            8
139 12 11          BE    RESET                            5  0268  B 291 S              8
140 12 12          A     +1,GAP                           7  0273  A X21 W95            9
141 12 13          MA    SIXN,HIGH                        7  0280  = X01 W89            9
142 12 14          B     EARLY                            4  0287  B 201                9
143 12 15   RESET  MLNS  +0,SW1                           7  0291  D X22 X16            9
144 12 16          B     COLSET                           4  0298  B V39                9
145  1 03          ORG   333                                          0333
146 99 99   *  THIS SECTION READS DATA CARD, PERFORMS SEQUENCE CHECK, AND
147 99 99   *  BRANCHES ON DELETION CARD.  TEST IS CONTROL BREAK ON J-INDEX.
```

Table 31. Program 3—Compute Contained in Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP     OPERANDS                                                          SFX CT  LOCN  INSTRUCTION TYPE  CARD

148 99 99   *  CARD ESTABLISHES NUMBER OF COLUMNS TO SKIP IN SEARCH FOR PROPER
149 99 99   *  COLUMN FOR MULTIPLICATION. SLIDE FINDS THE COLUMN.
150  1 04   READ   R                                                                          1     0333  1                   10
151  1 041         MLZS  +0,77                                                                7     0334  Y X22 079           10
152  1 05          C     79,SEQ                                                               7     0341  C 079 X05           10
153  1 06          BU    MIXED                                                                5     0348  B S79 /             10
154  1 07          A     +1,SEQ                                                               7     0353  A X21 X05           10
155  1 071         MLZS  +0,28                                                                7     0360  Y X22 028           10
156  1 09   CARD   C     28,COLNO                                                             7     0367  C 028 W92           11
157  1 10          BH    CLOSE                                                                5     0374  B 117 U             11
158  1 01          BWZ   DELETE,22,K                                                          8     0379  V 977 022 K         11
159  1 11          MLC   HIGH,SEARCH=3                                                        7     0387  M W89 X26           11
160  1 12          MA    SIX,SEARCH                                                           7     0394  = X14 X26           11
161  1 13          MLCWA 25,LOOK=3                                                            7     0401  L C25 X29           12
162  1 14          S     28,LOOK                                                              7     0408  S 028 X29           12
163  1 15          S     +1,LOOK                                                              7     0415  S X21 X29           12
164  1 16          MLCWA COLEN,BRL=3                                                          7     0422  L Y24 X32           12
165  1 17          A     +2,BRL                                                               7     0429  A X33 X32           12
166  1 18   SLIDE  MLCWA BRL,ARL=3                                                            7     0436  L X32 X36           13
167  1 19          C     +000,LOOK                                                            7     0443  C X39 X29           13
168  1 20          BE    THIS                                                                 5     0450  B 505 S             13
169  1 21          MLC   SEARCH,*+4                                                           7     0455  M X26 465           13
170  1 22          S     0,ARL                                                                7     0462  S 000 X36           13
171  1 221         MLZS  '0',ARL                                                              7     0469  Y X23 X36           14
172  1 23          MA    ARL                                                                  4     0476  = X36               14
173  1 24          MA    ARL,SEARCH                                                           7     0480  = X36 X26           14
174  1 25          S     +1,LOOK                                                              7     0487  S X21 X29           14
175  2 01          S     +1,BRL                                                               7     0494  S X21 X32           14
176  2 02          B     SLIDE                                                                4     0501  B 436               14
177 99 99   *  FINDS BOTTOM OF COLUMN TO BE MULTIPLIED.
178  2 03   THIS   MLCWA DIMEN,NUML=3                                                         7     0505  L X08 X42           15
179  2 04          MLC   SEARCH,*+11                                                          7     0512  M X26 529           15
180  2 05          MA    THREEN,*+4                                                           7     0519  = W98 529           15
181  2 06          S     0,NUML                                                               7     0526  S 000 X42           15
182  2 07          MLCWA NUML,ARL                                                             7     0533  L X42 X36           15
183  2 08          MLC   SEARCH,*+4                                                           7     0540  M X26 550           16
184  2 09          S     0,ARL                                                                7     0547  S 000 X36           16
185  2 091         MLZS  '0',ARL                                                              7     0554  Y X23 X36           16
186  2 10          MA    ARL                                                                  4     0561  = X36               16
187  2 11          MLCWA SEARCH,TRACE=3                                                       7     0565  L X26 X45           16
188  2 12          MA    ARL,TRACE                                                            7     0572  = X36 X45           16
189  2 13          MLCWA HIGH,FOLLOW=3                                                        7     0579  L W89 X48           17
190  2 14          BWZ   CONCOM,22,B                                                          8     0586  V 902 022 B         17
191  2 15          B     NULE                                                                 4     0594  B 625               17
192 99 99   *  TEST FOR LAST CARD
193  2 151  REMAT  MLNS  +0,[illegible]                                                       7     0598  D X22 X17           17
194  2 16          BLC   LAST                                                                 5     0605  B 614 A             17
195  2 17          B     READ                                                                 4     0610  B 333               17
196 99 99   *  SETS SWITCH ON LAST CARD
197  2 18   LAST   MLNS  1,SW4=1                                                              7     0614  D 001 X49           18
```

Table 31. Program 3—Compute Contained in Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP    OPERANDS                                                 SFX CT  LOCN  INSTRUCTION TYPE  CARD

198  2 19          B     CLOSE                                                      4     0621  B T17               18
199 99 99   *  MULTIPLICATION FOR REGULAR NUMERAL ON INPUT CARD. BACK IS A
200 99 99   *  REENTRY POINT FOR HOLDZ.
201 99 99   *  COMP IS A REENTRY POINT FOR ZZONE AND REPLC
202  3 001  NULL   C     '000',ARL                                                  7     0625  C X52 X36           18
203  3 002         BU    CONDIT                                                     5     0632  B 785 /             18
204  3 01   REGMUL ZA    22,CTR2=2                                                  7     0637  + 022 X54           18
205  3 02          S     +1,CTR2                                                    7     0644  S X21 X54           18
206  3 03          C     TRACE,SEARCH                                               7     0651  C X45 X26           19
207  3 04          BE    PATCH                                                      5     0658  B 796 S             19
208  3 05          MLC   FOLLOW,*+7                                                 7     0663  M X48 676           19
209  3 06          BWZ   HOLDZ,0,S                                                  8     0670  V 866 000 S         19
210  3 07          MLC   FOLLOW,*+7                                                 7     0678  M X48 691           19
211  3 071         BCE   COMP+12,0,=                                                8     0685  B 749 000 =         20
212  3 08   BACK   MLC   TRACE,*+11                                                 7     0693  M X45 710           20
213  3 09          MLC   FOLLOW,*+7                                                 7     0700  M X48 713           20
214  3 10          A     0,0                                                        7     0707  A 000 000           20
215  3 11          BCE   REPLC,SW3,1                                                8     0714  B 877 X60 1         20
216  3 12          MLC   TRACE,*+7                                                  7     0722  M X45 735           21
217  3 13          BWZ   ZZONE,0,S                                                  8     0729  V 767 000 S         21
218  3 14   COMP   C     +00,CTR2                                                   7     0737  C X56 X54           21
219  3 15          BU    REGMUL+7                                                   5     0744  B 644 /             21
220  3 16          MA    TWON,TRACE                                                 7     0749  = X11 X45           21
221  3 17          MA    TWON,FOLLOW                                                7     0756  = X11 X48           22
222  3 18          B     REGMUL                                                     4     0763  B 637               22
223 99 99   *  TAGS A CELL AS CONCOMITANT QUANTIFIER
224  3 20   ZZONE  MLC   FOLLOW,*+7                                                 7     0767  M X48 780           22
225  3 21          MLZS  SLASH,0                                                    7     0774  Y X15 000           22
226  3 22          B     COMP                                                       4     0781  B 737               22
227 99 99   *  SETS A SWITCH TO BYPASS MULTIPLICATION WHEN COLUMN CONSISTS
228 99 99   *  OF ONLY THE DIAGONAL
229  3 15   CONDIT MLNS  +1,SW5                                                     7     0785  D X21 X17           22
230  3 16          B     REGMUL                                                     4     0792  B 637               23
231 99 99   *  FINDS NUMBER OF NON-STORED ZERO CELLS IN COLUMN, SETS ADDRESS FO
232 99 99   *  FORCING IN OF MULTIPLICATION OF DIAGONAL
233  4 001  PATCH  B     REMAIN,SW5,1                                               8     0796  B 598 X17 1         23
234  4 01   DIAG   MLC   SEARCH,*+4                                                 7     0804  M X26 814           23
235  4 02          ZA    0,CTR3=3                                                   7     0811  + 000 X59           23
236  4 04          C     +000,CTR3                                                  7     0818  C X39 X59           23
237  4 05          BE    INS                                                        5     0825  B 848 S             23
238  4 051         S     +1,CTR3                                                    7     0830  S X21 X59           24
239  4 06          MA    TWON,FOLLOW                                                7     0837  = X11 X48           24
240  4 07          B     DIAG+14                                                    4     0844  B 818               24
241 99 99   *  MULTIPLICATION OF DIAGONAL
242  4 08   INS    MLC   FOLLOW,*+7                                                 7     0848  M X45 861           24
243  4 09          A     22,0                                                       7     0855  A 022 000           24
244  4 10          B     REMAIN                                                     4     0862  B 598               24
245 99 99   *  SETS SWITCH TO INDICATE CONCOMITANT CELL IN COLUMN
246  4 12   HOLDZ  MLNS  +1,SW3=1                                                   7     0866  O X21 X60           25
247  4 13          B     BACK                                                       4     0873  B 693               25
```

Table 31. Program 3—Compute Contained in Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP    OPERANDS                                                 SFX CT  LOCN  INSTRUCTION TYPE  CARD

248 99 99   *  TAGS A CELL AS A CONCOMITANT QUANTIFIER, RESETS A SWITCH
249  4 14   REPLC  MLC   FOLLOW,*+7                                                   7   0877  M X48 890           25
250  4 15          MLZS  SLASH,0                                                      7   0884  Y X15 000           25
251  4 16          MLNS  +0,SW3                                                       7   0891  D X22 X60           25
252  4 17          B     COMP                                                         4   0898  B 737               25
253 99 99   *  MULTIPLICATION FOR A CONCOMITANT QUANTIFIER ON INPUT CARD
254 99 99   *  ZEDIN FORCES A ZERO WHEN CONCOMITANTS ARE MULTIPLIED TOGETHER
255  5 01   CONCOM ZA    22,CTR2                                                      7   0902  + 022 X54           26
256  5 02          S     +1,CTR2                                                      7   0909  S X21 X54           26
257  5 03          C     TRACE,SEARCH                                                 7   0916  C X45 X26           26
258  5 04          BE    CONDIA                                                       5   0923  B /06 S             26
259  5 041         MLC   TRACE,*+7                                                    7   0928  M X45 941           26
260  5 042         C     +00,0                                                        7   0935  C X56 000           27
261  5 043         BE    ZEDIN                                                        5   0942  B ‡88 S             27
262  5 05          MLC   TRACE,*+7                                                    7   0947  M X45 960           27
263  5 06          BWZ   ZEDIN,0,S                                                    8   0954  V ‡88 000 S         27
264  5 061         MLC   FOLLOW,*+7                                                   7   0962  M X48 975           27
265  5 062         BCE   ZEDIN,0,=                                                    8   0969  B ‡88 000 =         28
266 99 99   *  PUTS =-SYMBOL IN COLUMN TO INDICATE DELETION
267  6 06   DELETE MLC   CELLS+6,FOLLOW                                               7   0977  M V80 X48           28
268  6 07          MLC   25,NOML                                                      7   0984  M 025 X42           28
269  6 08          S     COLNO,NOML                                                   7   0991  S W92 X42           28
270  6 09          S     +1,NOML                                                      7   0998  S X21 X42           28
271  6 091         MLZS  '0',NOML                                                     7   1005  Y X23 X42           29
272  6 10          MA    NOML                                                         4   1012  = X42               29
273  6 11          MA    NOML,FOLLOW                                                  7   1016  = X42 X48           29
274  6 12          MLC   FOLLOW,*+7                                                   7   1023  M X48 ‡36           29
275  6 13          MLCWA '0=',0                                                       7   1030  L X62 000           29
276  6 14          B     REMAIN                                                       4   1037  B 598               29
277  5 07          MLC   TRACE,*+11                                                   7   1041  M X45 ‡58           30
278  5 08          MLC   FOLLOW,*+7                                                   7   1048  M X48 ‡61           30
279  5 09          A     0,0                                                          7   1055  A 000 000           30
280  5 10          MLC   FOLLOW,*+7                                                   7   1062  M X48 ‡75           30
281  5 11          MLZS  SLASH,0                                                      7   1069  Y X15 000           30
282  5 12          C     +00,CTR2                                                     7   1076  C X56 X54           31
283  5 13          BU    CONCOM+7                                                     5   1083  B 909 /             31
284  5 14   ZEDIN  MA    TWON,TRACE                                                   7   1088  = X11 X45           31
285  5 15          MA    TWON,FOLLOW                                                  7   1095  = X11 X48           31
286  5 16          B     CONCOM                                                       4   1102  B 902               31
287 99 99   *  FINDS NUMBER OF NON-STORED ZERO CELLS - CONCOMITANT MULT
288  5 18   CONDIA MLC   SEARCH,*+4                                                   7   1106  M X26 /16           31
289  5 19          ZA    0,CTR3                                                       7   1113  + 000 X59           32
290  5 21          C     +000,CTR3                                                    7   1120  C X39 X59           32
291  5 22          BE    CONINS                                                       5   1127  B /50 S             32
292  5 221         S     +1,CTR3                                                      7   1132  S X21 X59           32
293  5 23          MA    TWON,FOLLOW                                                  7   1139  = X11 X48           32
294  5 24          B     CONDIA+14                                                    4   1146  B /20               32
295 99 99   *  MULTIPLICATION OF DIAGONAL - CONCOMITANT
296  6 01   CONINS MLC   FOLLOW,*+7                                                   7   1150  M X48 /63           33
297  6 02          A     22,0                                                         7   1157  A 022 000           33
```

Table 31. Program 3—Compute Contained in Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP    OPERANDS                                                      SFX CT  LOCN  INSTRUCTION TYPE  CARD

298  6 03          MLC   FOLLOW,*+7                                                        7  1164  M X48 /77            33
299  6 04          MLZS  SLASH,?                                                           7  1171  Y X15 000            33
300  6 05          B     REMAIN                                                            4  1176  B 598                33
301 99 99   *  FORMS AND STORES DATA WORD IN PLACE OF DIAGONAL. CONTAINS COLUMN
302 99 99   *  INDEX AND NUMBER OF ZERO CELLS ELIMINATED
303  7 01   SETDIG MLC   TOP,*+7                                                           7  1182  M Y12 /95            33
304  7 02          MLCWA GAP,0                                                             7  1189  L W95 000            34
305  7 03          MA    THRESH,TOP                                                        7  1196  = W98 Y12            34
306  7 04          MLC   TOP,*+7                                                           7  1203  M Y12 S16            34
307  7 05          MLCWA COLNO,0                                                           7  1210  L W92 000            34
308  7 06          MLC   BOTTOM,HIGH                                                       7  1217  M Y15 W89            34
309 99 99   *  PUNCHES DIAGONAL CELL CARD ON LAST COLUMN  GIVES JOB DONE
310 99 99   *  MESSAGE AND HALTS
311  7 08   PDIAG  MLCWA 106,103                                                           7  1224  L 106 103            35
312  7 09          MLCWA '01',108                                                          7  1231  L X64 108            35
313  7 10          P                                                                       1  1238  4                    35
314  7 11          B     STEPR,SW1,1                                                       8  1239  B 254 X16 1          35
315  7 12          S     +1,COLNO                                                          7  1247  S X21 W92            35
316  7 13          B     COLSET,SW4,0                                                      8  1254  B V39 X49 0          35
317  7 131         CS    332                                                               4  1262  / 332                36
318  7 132         CS                                                                      1  1266  /                    36
319  7 133         MLC   'JOB DONE',208                                                    7  1267  M X72 208            36
320  7 134         W                                                                       1  1274  2                    36
321  7 135         H     *-4                                                               4  1275  . S74                36
322 99 99   *  ERROR ROUTINE FOR SEQUENCE CHECK
323  7 14   MIXED  CS    332                                                               4  1279  / 332                36
324  7 15          CS                                                                      1  1283  /                    36
325  7 16          MLC   'SEQUENCE-NEXT SB       FIX AND RESTART',237                      7  1284  M Y09 237            37
326  7 161         MLZS  '0',SEQ                                                           7  1291  Y X23 X05            37
327  7 17          MLC   SEQ,221                                                           7  1298  M X05 221            37
328  7 171         MLZS  +1,SEQ                                                            7  1305  Y X21 X05            37
329  7 18          W                                                                       1  1312  2                    37
330  7 19          H     READ                                                              4  1313  . 333                37
331 99 99   *  ON COLUMN CONTROL BREAK, FINDS ENDS OF COLUMN
332  8 01   CLOSE  B     EARLY,SW1,1                                                       8  1317  B 201 X16 1          38
333  8 02          MLC   CELLS+6,TOP-3                                                     7  1325  M V80 Y12            38
334  8 03          MLC   HIGH,BOTTOM-3                                                     7  1332  M W89 Y15            38
335  8 04          MLC   COLNO,106                                                         7  1339  M W92 106            38
336  8 041         MLZS  '0',106                                                           7  1346  Y X23 106            38
337  8 05          MLC   DIMEN,S[illegible]+3                                              7  1353  M X08 Y18            39
338  8 06          MLCWA +000,GAP                                                          7  1360  L X39 W95            39
339 99 99   *  ELIMINATES ZERO CELLS UNDER DIAGONAL
340  8 061  HERE   MLC   TOP,*+7                                                           7  1367  M Y12 T80            39
341  8 062         B     REMOVE,0,=                                                        8  1374  B W40 000 =          39
342  8 07          MLC   TOP,*+7                                                           7  1382  M Y12 T95            39
343  8 08          C     +00,0                                                             7  1389  C X56 000            40
344  8 09          BU    PUNCH                                                             5  1396  J 019 /              40
345  8 10          MA    '002',TOP                                                         7  1401  = Y21 Y12            40
346    11          A     +1,GAP                                                            7  1408  A X21 W95            40
347  8 12          B     HERE                                                              4  1415  B T67                40
```

Table 31. Program 3—Compute Contained in Analysis Matrix (Continued)

```
SEQ PG LIN  LABEL  OP     OPERANDS                                                   SFX CT  LOCN  INSTRUCTION TYPE  CARD

348 99 99   *  PUNCH ROUTINE, EXCEPT DIAGONAL.   ZERO CELLS NOT PUNCHED
349 99 99   *  STEP CHANGES ADDRESSES
350  8 13   PUNCH  MA     TWON,TOP                                                     7   1419  = X11 Y12              40
351  8 131         C      TOP,BOTTOM                                                   7   1426  C Y12 Y15              41
352  8 132         BE     SETDIG                                                       5   1433  B 782 S                41
353  8 133         MLC    BOTTOM,*+7                                                   7   1438  M Y15 051              41
354  8 134         B      TAKOUT,0,=                                                   8   1445  B W58 000 =            41
355  8 14          MLC    BOTTOM,*+7                                                   7   1453  M Y15 066              41
356  8 15          C      +00,0                                                        7   1460  C X56 000              42
357  8 16          BE     STEP                                                         5   1467  B V09 S                42
358  8 17          MLCWA  ROWNO,103                                                    7   1472  L Y18 103              42
359  8 171         MLZS   '0',103                                                      7   1479  Y X23 103              42
360  8 18          MLC    BOTTOM,*+4                                                   7   1486  M Y15 096              42
361  8 19          MLCWA  0,108                                                        7   1493  L 000 108              43
362  8 191         BWZ    SWEEP,108,B                                                  8   1500  V W76 108 B            43
363  8 20          P                                                                   1   1508  4                      43
364  8 21   STEP   MA     TWON,BOTTOM                                                  7   1509  = X11 Y15              43
365  8 22          C      TOP,BOTTOM                                                   7   1516  C Y12 Y15              43
366  8 23          BE     SETDIG                                                       5   1523  B 782 S                43
367  8 24          S      +1,ROWNO                                                     7   1528  S X21 Y18              44
368  8 25          B      PUNCH+7                                                      4   1535  B U26                  44
369 99 99   *  FINDS NUMBER OF CELLS IN NEXT COLUMN TO BE COMPUTED.  CELLS
370 99 99   *  SETS EACH CELL TO ZERO, ALSO TESTS TO KEEP FROM STORING TOO
371 99 99   *  LARGE A MATRIX.
372  9 01   COLSET MLC    DIMEN,COLEN=3                                                7   1539  M X08 Y24              44
373  9 02          S      COLNO,COLEN                                                  7   1546  S W92 Y24              44
374  9 03          MLC    +000,CTR1=3                                                  7   1553  M X39 Y27              44
375  9 04          MA     SIX0,HIGH                                                    7   1560  = X01 W89              44
376  9 05          MLC    HIGH,CELLS+6                                                 7   1567  M W89 V80              45
377  9 06   CELLS  MLCWA  +00,0                                                        7   1574  L X56 000              45
378  9 07          A      +1,CTR1                                                      7   1581  A X21 Y27              45
379  9 08          C      CTR1,COLEN                                                   7   1588  C Y27 Y24              45
380  9 09          BE     CARD                                                         5   1595  B 367 S                45
381  9 10          MA     TWON,CELLS+6                                                 7   1600  = X11 V80              46
382  9 11          C      '999',1350                                                   7   1607  C X20 Y50              46
383  9 12          BO     OVRFLO                                                       5   1614  B W23 Z                46
384  9 13          B      CELLS                                                        4   1619  B V74                  46
385 99 99   *  ERROR ROUTINE IF MEMORY CAPACITY IS EXCEEDED
386  9 15   OVRFLO CS     332                                                          4   1623  / 332                  46
387  9 16          CS                                                                  1   1627  /                      46
388  9 17          MLC    'MEMORY FULL',211                                            7   1628  M Y38 211              46
389  9 18          W                                                                   1   1635  2                      47
390  9 19          H      *-4                                                          4   1636  . W35                  47
391 99 99   *  REMOVE AND TAKOUT CHANGE --SYMBOL TO ZERO AFTER COLUMN IS
392 99 99   *  COMPUTED
393  9 20   SWEEP  MLC    TOP,*+7                                                      7   1640  M Y12 W53              47
394  9 21          MLCWA  +00,0                                                        7   1647  L X56 000              47
395  9 22          B      HERE+15                                                      4   1654  B T82                  47
396  9 23   TAKOUT MLC    BOTTOM,*+7                                                   7   1658  M Y15 W71              47
397  9 24          MLCWA  +00,                                                         7   1665  L X56 000              47
```

Table 31. Program 3--Compute Contained in Analysis Matrix (Continued)

```
SEQ PG LIN LABEL   OP     OPERANDS                                            SFX CT  LOCN  INSTRUCTION TYPE  CARD

398  9 25          B      STEP                                                     4  1672  B V09               48
399 99 99  *  CLEARS PLUS SIGN FROM REGULAR CELL
400  9 26  SWEEP   MLZS   '0',108                                                  7  1676  Y X23 108           48
401  9 27          B      STEP-1                                                   4  1683  B V08               48
402 99 99  *  CONSTANTS AND SYMBOLS
403 10 01  HIGH    DSA    7999                                                     3  1689  I9Z                 49
404 10 02  COLNO   DCW    +000                                                     3  1692                      49
405 10 03  GAP     DCW    +000                                                     3  1695                      49
406 10 04  THREEN  DSA    15997                                                    3  1698  I9G                 49
407 10 05  SIXN    DSA    15994                                                    3  1701  I9D                 49
408 10 06  SEQ     DCW    +0000                                                    4  1705                      49
409 10 07  DIMEN   DCW    +000                                                     3  1708                      49
410 10 08  TWON    DSA    15998                                                    3  1711  I9H                 49
411 10 10  SIX     DSA    006                                                      3  1714  006                 49
412 10 11  SLASH   DC     '/'                                                      1  1715                      49
413 10 12  SW1     DCW    '0'                                                      1  1716                      49
414 10 14  SW5     DC     '0'                                                      1  1717                      49
                   DCW    '999'                                                    3  1720              LIT     49
                          +1                                                       1  1721              LIT     50
                          +0                                                       1  1722              LIT     50
                          '0'                                                      1  1723              LIT     50
       159 SEARCH         =03                                                      3  1726              AREA    50
       161 LOOK           =03                                                      3  1729              AREA    50
       164 BRL            =03                                                      3  1732              AREA    50
                          +2                                                       1  1733              LIT     50
       166 ARL            =03                                                      3  1736              AREA    51
                          +000                                                     3  1739              LIT     51
       178 NOML           =03                                                      3  1742              AREA    51
       187 TRACE          =03                                                      3  1745              AREA    51
       189 FOLLOW         =03                                                      3  1748              AREA    51
       197 SW4            =01                                                      1  1749              AREA    51
                          '000'                                                    3  1752              LIT     51
       204 CTR2           =02                                                      2  1754              AREA    52
                          +00                                                      2  1756              LIT     52
       235 CTR3           =03                                                      3  1759              AREA    52
       246 SW3            =01                                                      1  1760              AREA    52
                          '0='                                                     2  1762              LIT     52
                          '01'                                                     2  1764              LIT     52
       319                'JOB DONE'                                               8  1772              LIT     52
       325                'SEQUENCE-NEXT SB       FIX AND RESTART'                37  1809              LIT     53
       333 TOP            =03                                                      3  1812              AREA    54
       334 BOTTOM         =03                                                      3  1815              AREA    54
       337 ROWNO          =03                                                      3  1818              AREA    54
                          '002'                                                    3  1821              LIT     54
       372 COLEN          =03                                                      3  1824              AREA    54
       374 CTR1           =03                                                      3  1827              AREA    54
       388                'MEMORY FULL'                                           11  1838              LIT     54
415 12 17          END    SETUP                                                             / 101 080           55
```

Table 32. Test Input--Program 2 and 3

```
018         TEST PROGRAM
          S1        W201018014                                   0001
          D1        W201015014                                   0002
          D3        W101017013                                   0003
          D2        W101016013                                   0004
          D1        W101015013                                   0005
          S1        C10A018012                                   0006
          D3        C10A017012                                   0007
          S1        W30J018011                                   0008
          W2        W301014011                                   0009
          W1        W301013011                                   0010
          C1        W30A012011                                   0011
          D2        W401016008                                   0012
          H2        W401010008                                   0013
          H1        W401009008                                   0014
          C1        C20A012007                                   0015
          C2        C30A007006                                   0016
          D3        R10J017005                                   0017
          D1        R10J015005                                   0018
          C1        R10J012005                                   0019
          W3        R101011005                                   0020
          C2        R10A007005                                   0021
          C3        R10A006005                                   0022
          C2        C40A007004                                   0023
          C3        C50A006003                                   0024
          W4        R201008002                                   0025
          C2        R20J007002                                   0026
          C3        R20J006002                                   0027
          R1        R205005002                                   0028
          C4        R20A004002                                   0029
          C5        R20A003002                                   0030
          W3        P101011001                                   0031
```

Table 33. Memory Dump Showing Arrangement of Matrix in Memory--Program 2 and 3

```
                                                               0+0+0+0+0+0+00A00I0A0/0A0A0B0A0A0A00B
                                                               1 1 1 1 1 1 1  1  1 1 1 1 1 1 1 1 1
00+0/0/0E0V0V0A0A0A0E0V0E0E1+0F0E0E00C00B0/0+0+0+0+0+0+0+0+0+0+0+0+00000B0/0+0+0+0+0+0+0+0+0+0+0+00E
1  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1  1 1 1 1 1 1 1 1 1 1 1 1 1 1     1 1 1 1 1 1 1 1 1 1 1 1
00+0/0/0+0+0+0A0/0A0A0B0A0A0A00F00+0/0+0+0+0+0+0+0+0+0+0+0+00G00D0/0+0+0+0+0+0+00H00+0A0A0+0+0+0+0+0
1  1 1 1 1 1 1 1 1 1 1 1 1 1  1  1 1 1 1 1 1 1 1 1 1 1 1  1  1 1 1 1 1 1 1 1  1  1 1 1 1 1 1 1 1
A0+0+00I00I01+00H01A00+0/0A0A0B0A0A0A01B00D0/0/01C00A0A0A0A0+01B00+0A0+0+0A01E00C01F00B01G00A01H00++
 1 1 1  1  1  1  1  1  1 1 1 1 1 1 1 1  1  1 1 1  1  1 1 1 1 1  1  1 1 1 1 1  1  1  1  1  1  1  1
```

Table 34. Program 4—Print Solution Matrices

```
SEQ PG LIN  LABEL  OP    OPERANDS                                        SFX CT  LOCN  INSTRUCTION TYPE  CARD

101  1 01   000    JOB   PRINT SOLUTION MATRICES
102  1 02          CTL   4401
103 99 99   *  INPUT IS MASTER DECK FROM REINDEXING STEP. NEW INDICES ARE
104 99 99   *  IN CC 1-3, ORIGINAL INDICES IN CC 4-13. SORT IS ON NEW INDEX
105 99 99   *  NEXT IS TITLE CARD, TITLE IN CC 1-40, TYPE OF ANALYSIS IN
106 99 99   *  CC 52-64.
107 99 99   *  NEXT ARE OUTPUT CARDS FROM PROGRAM 1 OR 2, MAJOR SORT ON J-FIELD
108 99 99   *  CC 4-6, MINOR ON I-FIELD CC 1-3. Q-FIELD IS CC 7-8
109  1 03          ORG   333                                                            0333
110 99 99   *  CLEARS PUNCH AREA, SETS CONSTANTS
111  1 04   READY  CS    332                                             4   0333  / 332              4
112  1 05          CS                                                    1   0337  /                  4
113  1 06          CS                                                    1   0338  /                  4
114  1 061         MLZS  '0',BASIC-1                                     7   0339  Y C41 C30          4
115  1 07          SW    1,4                                             7   0346  , 001 004          4
116  1 08          MLCWA '000',IND1                                      7   0353  L C44 089          4
117 99 99   *  READS AND STORES MASTER CARDS
118  1 09   MREAD  R                                                     1   0360  1                  4
119  1 10          BWZ   SHIFT,79,2                                      8   0361  V 394 079 2        5
120  1 11          MLCWA 3,NEW                                           7   0369  L 003 7T1          5
121  1 12          MLCWA 13,OLD                                          7   0376  L 013 7U1          5
122  1 13          MA    '013',IND1                                      7   0383  = C47 089          5
123  1 14          B     MREAD                                           4   0390  B 360              5
124 99 99   *  PRINTS HEADINGS, SETS UP FOR DATA CARDS
125  1 16   SHIFT  CW    4                                               4   0394  ) 004              5
126  1 17          MLCWA 40,284                                          7   0398  L 040 284          6
127  1 18          W                                                     1   0405  2                  6
128  1 19          CC    L                                               2   0406  F L                6
129  1 20          CC    L                                               2   0408  F L                6
130  1 21          SW    51                                              4   0410  , 051              6
131  1 211         CS    299                                             4   0414  / 299              6
132  1 22          MLCWA 'ELEMENT',215                                   7   0418  L C54 215          6
133  1 23          MLCWA 64,280                                          7   0425  L 064 280          7
134  1 24          W                                                     1   0432  2                  7
135  1 25          CC    L                                               2   0433  F L                7
136  2 01          CS    80                                              4   0435  / 080              7
137  2 011         CS    299                                             4   0439  / 299              7
138  2 02          SW    1,4                                             7   0443  , 001 004          7
139  2 03          SW    7                                               4   0450  , 007              7
140  2 04          MLCWA '000',IND2                                      7   0454  L C44 094          8
141  2 05          MLCWA '000',IND1                                      7   0461  L C44 089          8
142 99 99   *  READS DATA CARDS, CHECKS FOR LAST CARD, CHECKS FOR COLUMN
143 99 99   *  CONTROL BREAK, CHECK FOR FULL PRINT LINE
144  2 11   DREAD  R                                                     1   0468  1                  8
145  2 12          BLC   FINAL                                           5   0469  B 707 A            8
146  2 13          C     6,3                                             7   0474  C 006 003          8
147  2 14          BE    CLOSE                                           5   0481  B 622 S            8
```

Table 34. Program 4—Print Solution Matrices (Continued)

```
SEQ PG LIN  LABEL  OP     OPERANDS                                              SFX CT  LOCN  INSTRUCTION TYPE  CARD

148  2 15          BCE    PRINT,CTR1,H                                              8   0486  B 598 C62 H          9
149 99 99   *  FINDS ORIGINAL ROW INDICES CORRESPONDING TO NEW INDICES, PLACES
150 99 99   *  IN PRINT AREA. ALSO QUANTITY TO PRIOT AREA. ASSEMBLES  SEVEN
151 99 99   *  CELLS DATA, THEN BRANCHES TO PRINT
152  2 16   SEEK   MLC    BASIC,WHERE=3                                             7   0494  M C31 C57            9
153  2 17          ZA     3,PROD-3                                                  7   0501  + 003 C34            9
154  2 18          M      +13,PROD                                                  7   0508  * C59 C37            9
155  2 19          SW     PROD-2                                                    4   0515  , C35                9
156  2 191         S      +1,PROD                                                   7   0519  S C60 C37           10
157  2 192         MLZS   '0',PROD                                                  7   0526  Y C41 C37           10
158  2 20          MA     PROD,WHERE                                                7   0533  = C37 C57           10
159  2 201         CW     PROD-2                                                    4   0540  ) C35               10
160  2 22          MLC    WHERE,*+4                                                 7   0544  M C57 554           10
161  2 23          MLC    0,ROWIND+X2                                               7   0551  M 000 2K5           10
162  2 24          MLCWA  EDIT2,QUANT+X2                                            7   0558  L C39 2K9           11
163  2 25          MCE    8,QUANT+X2                                                7   0565  E 008 2K9           11
164  2 251         BWZ    STAR,8,S                                                  8   0572  V 713 008 S         11
165  3 01   EXTRA  A      +1,CTR1=2                                                 7   0580  A C60 C62           11
166  3 02          MA     '017',IND2                                                7   0587  = C65 094           11
167  3 03          B      DREAD                                                     4   0594  B 468               12
168 99 99   *  PRINTS SEVEN CELL VALUES, THEN RESETS FOR NEXT PRINT LINE
169  3 07   PRINT  W                                                                1   0598  2                   12
170  3 08          CS     332                                                       4   0599  / 332               12
171  3 09          CS                                                               1   0603  /                   12
172  3 12          MLCWA  '000',IND2                                                7   0604  L C44 094           12
173  3 13          ZA     +1,CTR1                                                   7   0611  + C60 C62           12
174  3 14          B      SEEK                                                      4   0618  B 494               12
175 99 99   *  RESETS FOR NEXT COLUMN
176  3 16   CLOSE  C      '001',6                                                   7   0622  C C68 006           13
177  3 17          BE     ONLY                                                      5   0629  B 649 S             13
178  3 18          W                                                                1   0634  2                   13
179  3 19          CS     332                                                       4   0635  / 332               13
180  3 20          CS                                                               1   0639  /                   13
181  3 21          MA     '013',IND1                                                7   0640  = C47 089           13
182  3 22          CC     K                                                         2   0647  F K                 13
183  3 23   ONLY   MLC    'INPUT DATA',ROWIND+X0                                    7   0649  M C78 225           14
184  3 25          MLC    OLD,212                                                   7   0656  M 701 212           14
185  3 26          MLCWA  '000',IND2                                                7   0663  L C44 094           14
186  3 27          ZA     +1,CTR1                                                   7   0670  + C60 C62           14
187  3 28          BCE    DREAD,SW2,+                                               8   0677  B 468 C40 +         14
188  3 281         W      DONE                                                      4   0685  2 689               15
189 99 99   *  END OF JOB ROUTINE
190  4 01   DONE   CC     K                                                         2   0689  F K                 15
191  4 011         CS     299                                                       4   0691  / 299               15
192  4 02          MLCWA  'JOB DONE',212                                            7   0695  L C86 212           15
193  4 03          W                                                                1   0702  2                   15
194  4 04          H      *-4                                                       4   0703  . 702               15
195 99 99   *  SETS SWITCH ON LAST CARD
196  4 07   FINAL  MLNS   +1,SW2                                                    7   0707  D C60 C40           15
197  4 08          B      DREAD+6                                                   4   0714  B 474               16
```

Table 34. Program 4—Print Solution Matrices (Continued)

```
SEQ PG LIN  LABEL  OP    OPERANDS                                                 SFX CT  LOCN  INSTRUCTION TYPE   CARD

198 99 99   *  SETS SYMBOL * TO IDENTIFY CONCOMITANT CELL
199  4 10   STAR   MLC   '*',230+X2                                                    7  0718  M C87 2L0           16
200  4 11          B     ENTRA                                                         4  0725  B 580               16
201 99 99   *  SYMBOLS AND CONSTANTS
202  5 01   DICTRY DA    200X13,X1                                                        0729   3328               49
203  5 02   OLD          4,13                                                             0741              FIELD   49
204  5 03   NEW          1,3                                                              0731              FIELD   83
205  5 04   IND1   EQU   89                                                               0089
206  5 06   BASIC  DCW   +DICTRY                                                       3  3331  7S9                116
207  5 07   PROD   DCW   =6                                                            6  3337                     116
208  5 08   QUANT  EQU   229                                                              0229
209  5 09   ROWIND EQU   225                                                              0225
210  5 10   IND2   EQU   94                                                               0094
211  5 11   EDIT2  DCW   ' 0'                                                          2  3339                     116
212  5 12   SW2    DCW   +0                                                            1  3340                     116
                   DCW   '0'                                                           1  3341              LIT    116
                         '000'                                                         3  3344              LIT    117
                         '013'                                                         3  3347              LIT    117
       132               'ELEMENT'                                                     7  3354              LIT    117
       152  WHERE        =03                                                           3  3357              AREA   117
                         +13                                                           2  3359              LIT    117
                         +1                                                            1  3360              LIT    117
       165  CTR1         =02                                                           2  3362              AREA   117
                         '017'                                                         3  3365              LIT    118
                         '001'                                                         3  3368              LIT    118
       133               'INPUT DATA'                                                 10  3378              LIT    118
       192               'JOB DONE'                                                    8  3386              LIT    118
                         '*'                                                           1  3387              LIT    118
213  5 13          END   READY                                                                  / 333 080          119
```

Table 35. Test Input—Program 4—Composed of Analysis

```
TEST DECK - PROGRAM 4 - COMPOSED OF ANALYSIS.

MAKE UP OF MASTER CARDS USED FOR REINDEXING AND OUTPUT CARDS FROM PROGRAM 2.


001        P1                                                                  -
002        R2                                                                  -
003        C5
004        C4                                                                  -
005        R1                                                                  -
006        C3                                                                  -
007        C2                                                                  -
008        W4                                                                  -
009        H1                                                                  -
010        H2                                                                  -
011        W3                                                                  -
012        C1                                                                  -
013        W1                                                                  -
014        W2                                                                  -
015        D1                                                                  -
016        D2                                                                  -
017        D3                                                                  -
018        S1                                                                  -
S* MATRIX - COMPOSED OF ANALYSIS                  IS COMPOSED OF
00100101
01100101
0120010/
01300101
01400101
01500102
01600101
01700101
01800101
00200201
0030020/
0040020/
00500205
0060020V
0070020V
00800201
00900201
01000201
01100205
0120020V
01300205
01400205
01500210
01600206
01700205
01800205
00300301
0060030/
00400401
0070040/
00500501
0060050/
0070050/
01100501
0120050/
01300501
01400501
```

Table 35. Test Input—Program 4—Composed of Analysis (Continued)

```
01500502
01600501
01700501
01800501
00600601
00700607
00700701
01200707
00800801
00900801
01000801
01600801
00900901
01001001
01101101
01201107
01301101
01401101
01501102
01601101
01701101
01801101
01201201
01701207
01801207
01301301
01501301
01601301
01701301
01401401
01501401
01801401
01501501
01601601
01701701
01801801
```

Table 36. Test Output—Program 4—Composed of Analysis

```
(* Denotes Concomitant Cell Value)

ELEMENT                                         IS COMPOSED OF

  P1          W3   1      C1   1*     W1   1      W2   1      D1   2      D2   1      D3   1
              S1   1

  R2          C5   1*     C4   1*     R1   5      C3   5*     C2   5*     W4   1      H1   1
              H2   1      W3   5      C1   5*     W1   5      W2   5      D1  10      D2   6
              D3   5      S1   5

  C5          C3   1*

  C4          C2   1*

  R1          C3   1*     C2   1*     W3   1      C1   1*     W1   1      W2   1      D1   2
              D2   1      D3   1      S1   1

  C3          C2   1*

  C2          C1   1*

  W4          H1   1      H2   1      D2   1

  H1    INPUT DATA

  H2    INPUT DATA

  W3          C1   1*     W1   1      W2   1      D1   2      D2   1      D3   1      S1   1

  C1          D3   1*     S1   1*

  W1          D1   1      D2   1      D3   1

  W2          D1   1      S1   1

  D1    INPUT DATA

  D2    INPUT DATA

  D3    INPUT DATA

  S1    INPUT DATA

JOB DONE
```

Table 37. Test Input—Program 4—Contained in Analysis

```
TEST DECK - PROGRAM 4 - CONTAINED IN ANALYSIS

MADE UP OF MASTER CARDS USED FOR REINDEXING AND OUTPUT CARDS FROM PROGRAM 3.

001         P1                                                                     -
002         R2                                                                     -
003         C5                                                                     -
004         C4                                                                     -
005         R1                                                                     -
006         C3                                                                     -
007         C2                                                                     -
008         W4                                                                     -
009         H1                                                                     -
010         H2                                                                     -
011         W3                                                                     -
012         C1                                                                     -
013         W1                                                                     -
014         W2                                                                     -
015         D1                                                                     -
016         D2                                                                     -
017         D3                                                                     -
018         S1                                                                     -
S-PRIME-* MATRIX - CONTAINED IN ANALYSIS             CONTAINS
00100101
01200107
01500102
01600101
01700101
00200201
00300207
00400207
00900201
01000201
01600206
00300301
00600307
00400401
00700407
00500501
00600507
00700507
01600501
00600601
00700607
00700701
01200707
00800801
00900801
01000801
01600801
00900901
01001001
01101101
01201107
01501102
01601101
01701101
01201201
01701207
01801207
```

Table 37. Test Input—Program 4—Contained in Analysis (Continued)

```
01301301
01501301
01601301
01701301
01401401
01501401
01801401
01501501
01601601
01701701
01801801
```

Table 38. Test Output—Program 4—Contained in Analysis

```
(* Denotes Concomitant Cell Value)

ELEMENT                                         CONTAINS

 P1          C1   1*      D1   2       D2   1       D3   1

 R2          C5   1*      C4   1*      H1   1       H2   1       D2   6

 C5          C3   1*

 C4          C2   1*

 R1          C3   1*      C2   1*      D2   1

 C3          C2   1*

 C2          C1   1*

 W4          H1   1       H2   1       D2   1

 H1   INPUT DATA

 H2   INPUT DATA

 W3          C1   1*      D1   2       D2   1       D3   1

 C1          D3   1*      S1   1*

 W1          D1   1       D2   1       D3   1

 W2          D1   1       S1   1

 D1   INPUT DATA

 D2   INPUT DATA

 D3   INPUT DATA

 S1   INPUT DATA
```

LITERATURE CITED

LITERATURE CITED

1. *A New Approach to Office Mechanization, Integrated Data Processing Through Common Language Machines,* American Management Association, 1954.

2. *Automated Data Processing,* Moore Business Forms, Inc., Niagara Falls, New York, Undated.

3. Brooks, G. H., *An Investigation of the "Gozinto" Method of Materials Requirements Determination,* International Business Machines Corp., Chicago, Illinois, 1961. (Privately circulated paper.)

4. Cantrell, H. N., "Where are Compiler Languages Going," *Datamation,* August, 1962, p. 25-28.

5. Ellis, H., *Organizing for Systems Development Engineering and Application,* unpublished paper presented to the Industrial Engineering Conference, Purdue University, April 5, 1960.

6. Evans, O. Y., *Decision Tables--A Preliminary Reference Manual,* International Business Machines Corp., White Plains, New York, September, 1961. (Ref. No. 1J1)

7. Evans, O. Y., *General Information Manual--Advanced Analysis Method for Integrated Electronic Data Processing,* International Business Machines Corp., F20-8047, undated.

8. *General Information Manual--COBOL,* International Business Machines Corp., F28-8053-1, White Plains, New York, 1961.

9. Giffler, B., *Mathematical Solution of Explosion and Scheduling Problems,* IBM Research Center, International Business Machines Corp., Yorktown Heights, RC-128, May, 1959.

10. Grad, B., "Tabular Form in Decision Logic," *Datamation,* July, 1961, p. 22-26.

11. Grad, B. and Canning, R. G., "Information Process Analysis," *Journal of Industrial Engineering,* November—December, 1959, p. 470-476.

12. Grosch, H. R. J., "Magic Languages," *Datamation,* February, 1963.

13. Henderson, P. B., *A Theory of Data Systems for Economic Decisions,* Ph.D. Dissertation, Massachusetts Institute of Technology, Department of Economics and Social Science, June, 1960.

14. Homer, E. D., "A Generalized Model for Analyzing Management Information Systems," *Management Science*, Vol. 8, 1962.

15. Kavanagh, T. F., "TABSOL--The Language of Decision Making," *Computers and Automation*, September, 1961, p. 15-22.

16. Kavanagh, T. F., "Decision Structure Tables--A Technique for Business Decision Making," *Journal of Industrial Engineering*, September-October, 1963, p. 249-258.

17. Kozmetsky, G. and Kirschner, P., *Electronic Computers and Management Control*, New York, McGraw-Hill, 1956.

18. Lieberman, I. J., "A Mathematical Model for Integrated Business Systems," *Management Science*, Vol. 2, 1956, p. 327-336.

19. MAP Systems Charting Technique, National Cash Register Company, Dayton, Ohio, MD 100-40, 1961.

20. McGee, W. C., "Generalization: Key to Successful Electronic Data Processing," *Journal of the ACM*, Vol. 6, 1959, p. 1-23.

21. McGee, R. C. and Tellier, H., "A Re-evaluation of Generalization," *Datamation*, July—August, 1960, p. 25-29.

22. Orchard-Hays, W., "Computer Languages," *International Science and Technology*, December, 1962, p. 33-41.

23. Pollack, S. L., "CODASYL, COBOL, & DETAB-X," *Datamation*, February, 1963, p. 60-61.

24. Postley, J. A., "Current Trends in Business Data Processing," *Data Processing*, December, 1961, p. 11-14.

25. Rowe, A. J., "A Research Approach in Management Controls," *Journal of Industrial Engineering*, May—June, 1960, p. 251-258.

26. Schmidt, R. N. and Meyers, W. E., *Electronic Business Data Processing*, Holt, Rinehart and Winston, New York, 1963, p. 6-16.

27. Shaw, C. J., Sr., "The Language Proliferation," *Datamation*, May, 1962, p. 34-36.

28. Steel, T. B., Jr., "Toward a Theory of Data Processing," *Data Processing*, February, 1962, p. 9-11.

29. Thomas, W. H., *An Investigation of the Use of Symbolic Logic for Data Systems Analysis*, unpublished project report, Purdue University, 1961.

30. Vazsonyi, A., *Scientific Programming in Business and Industry*, New York, Wiley, 1958.

31. Wagner, H. M., "Statistical Decision Theory as a Method for Information Processing," *Journal of Industrial Engineering*, Vol. X, No. 1, January—February, 1959, p. 25-32.

32. Young, J. W. and Kent, H. K., "Abstract Formulation of Data Processing Problems," *Journal of Industrial Engineering*, November—December, 1958, p. 471-479.

OTHER REFERENCES

OTHER REFERENCES

1. Carver, W. W., "Machines and Languages," *Data Processing*, December, 1962, p. 21-26.

2. Ellis, H., "Integrated Systems Produce Profit," *Advances in EDP and Information Systems*, American Management Association Bulletin #62.

3. Galler, B. A., *The Language of Computers*, McGraw-Hill, New York, 1962.

4. Head, R. V., "The Programming Gap in Real-Time Systems," *Datamation*, February, 1963, p. 39-41.

5. Lazzaro, V., Ed., *Systems and Procedures*, Prentice-Hall, Englewood Cliffs, N. J., 1959.

6. Ledley, R. S., *Programming and Utilizing Digital Computers*, McGraw-Hill, New York, 1962.

7. Leeds, H. D. and Weinberg, G. M., *Computer Programming Fundamentals*, McGraw-Hill, New York, 1961.

8. Martin, E. W., Jr., *Electronic Data Processing*, Irwin, Homewood, Illinois, 1961.

9. McCracken, D. D., Weiss, H., and Lee, T., *Programming Business Computers*, Wiley, New York, 1959.

10. Optner, S. L., *Systems Analysis for Business Management*, Prentice-Hall, Englewood Cliffs, N. J., 1960.

11. *Systems Analysis Techniques--A Survey*, Technical Report No. 1D1, International Business Machines Corp., White Plains, New York, January 15, 1961.

12. Utman, R. E., "Language Standards, A Status Report," *Datamation*, November, 1961, p. 26-28.

VITA

George H. Brooks was born in Hamden, Connecticut, on April 9, 1919, the son of Frank W. and Nellie O. (née Marquet) Brooks. He attended public school in Adams Basin and Spencerport, New York, graduating from Spencerport High School in June, 1936.

From his graduation until May, 1940, the author worked as a mechanic in the Allen-Wales Adding Machine Corporation, Ithaca, New York, and for a year, until June, 1941, as an Assistant Section Chief in the Census Bureau, Washington, D. C.

In June, 1941, the author entered the United States Army as a Private. He was discharged from the Army as a Technical Sergeant in September, 1943, to accept a commission as Second Lieutenant. He served in the South Pacific area, and was returned to the United States in January, 1946 and released from active duty as Captain. He subsequently served two three-month tours of duty as Captain, Chemical Corps, in 1947 and 1950.

In December, 1942, while in the military service, he was married to Hope Lewis. In September, 1946, he entered the University of Florida to pursue a program of study in Industrial Engineering. In January, 1947, a daughter, Marcia Hope, was born. He was graduated from the University of Florida in January, 1950, with the degree Bachelor of Industrial Engineering, with honors. He was elected to Phi Eta Sigma, freshman scholastic honorary, Sigma Tau, engineering honorary, and Phi Kappa Phi, scholastic honorary societies.

In June, 1950, the author entered the Georgia Institute of Technology where he undertook a program of study in Industrial Engineering, which he completed in June, 1951, and was awarded the degree Master of Science in Industrial Engineering. During the course of his studies he was also employed as part-time instructor in the School of Industrial Engineering.

In June, 1951, the author was employed by the Engineering Service Division, Engineering Department, E. I. duPont de Nemours and Co., Inc., Wilmington, Delaware. During the first three years of this employment he was assigned to the Spruance Cellophane plant, Richmond, Virginia, where he was the Chief Standards Engineer. In January, 1952, a son, Ralph Marquet, was born. In 1954 the author returned to Wilmington where he became a consultant and consultant supervisor in the Engineering Service Division. His experience for several years was in connection with the development of information and management control systems for several duPont operating departments.

In September, 1959, he accepted a position as Assistant Professor of Industrial Engineering at Purdue University. He was promoted to the rank of Associate Professor in September, 1961. He also did extensive consultation for the International Business Machines Corporation and the Community Hospital, Indianapolis, Indiana.

In 1963 the author was granted a National Science Foundation Science Faculty Fellowship, and re-entered the Georgia Institute of Technology to pursue a Ph.D. program in the School of Industrial Engineering. At the expiration of his fellowship, he accepted a position as Professor of Industrial and Systems Engineering at the University of Florida, where he is now employed.

The following publications resulted from the author's research and consulting activities while at Purdue:

"A New Technique for Prediction of Future Hospital Bed Needs," *Hospital Management*, Vol. 97, No. 6, June, 1964. (with H. L. Beenhakker)

"Predicting Hospital Bed Needs," *International Nursing Review*, (London), Vol. 11, No. 3, June, 1964. (with H. L. Beenhakker and W. C. McLin)

"An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem," accepted and scheduled for the January-February, 1965, issue of the *Journal of Industrial Engineering*. (with C. R. White)