This dissertation has been microfilmed exactly as received

DONOVAN, John Joseph, 1942-
INVESTIGATIONS IN SIMULATION AND SIMULATION LANGUAGES.

Yale University, Ph.D., 1967
Engineering, electrical

University Microfilms, Inc., Ann Arbor, Michigan
ACKNOWLEDGEMENT

Recognition and my gratitude I give to my advisor, Professor Conrad A. Wogrin, whose scientific criticisms, questions, and engineering insight have aided in the formation and expression of every idea presented in this dissertation. The penetrating lectures on mathematical logic by Professor Trenchard More have been especially helpful. Application of Professor More's mathematical background and my programming background has resulted in an effort to extend the work presented in these lectures to programming languages. This joint effort is the basis of Chapter III. I gratefully acknowledge the work of R. M. Smullyan and E. L. Post, which has also been the basis of Chapter III. I wish to extend my thanks to my fellow student James Gray for our early discussions of this work and to Donald McKay and Dr. David Seligson for their help with the clinical laboratory work. I am grateful to Professor Morton Kenefsky for his friendly encouragement.

I thank my fellow student Thomas Kabaservice and his wife for their careful proofreading of the final manuscript. Likewise, to Miss LaVerne Furs for her patience in typing this manuscript. To both Yale University and the National Institute of Health I extend thankful acknowledgement of their financial support, computer facilities, and staff.

I should like to also express my gratitude to my wife who has proofread every draft and uncovered many errors. Her unselfish attitude, encouragement, and softening of the disappointments have aided in the completion of this work.
CONTENTS

ACKNOWLEDGEMENT

FIGURES AND TABLES

SUMMARY

CHAPTER I  INTRODUCTION
  I.1 Preliminary Remarks
  I.2 Preliminary Definitions
  I.3 Digital Simulation
  I.4 Developments in Simulation
  I.5 Needs and Deficiencies
  I.6 Purpose of Dissertation
  I.7 Contributions

CHAPTER II  SYNTAX SPECIFICATION - A SURVEY
  II.1 Introduction
  II.2 Survey
  II.3 Phase Structure Grammars
  II.4 Shortcomings and Difficulties of Past Specifications
  II.4 Bases of Specification Presented

CHAPTER III  A RECURSIVE SPECIFICATION FOR COMPUTER LANGUAGES
  III.1 Canonic Systems

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
### III. 2 Limited FORTRAN

21

### III. 3 Conventions

22

### III. 4 Development of Canonic Systems for Computer Languages

23

### III. 5 Predicate, Variable, Term, Remark, Canon

24

### III. 6 Recursion

25

### III. 7 Edicts

26

### III. 8 Development of Canonic System for Limited FORTRAN

28

### III. 9 Canonic System of Limited FORTRAN

34

### III. 10 Canonic System of the Canonic System of Limited FORTRAN

36

### III. 11 Example of Derivation

37

### III. 12 Syntax of GPSS III

40

### III. 13 Examples of a Use of the Specification

44

### III. 14 Development and Motivation of a Complete Syntactical Specification of GPSS III

47

  a. Symbols, Spaces, Numbers, Letters

  b. Labels

  c. Standard Numerical Attributes

  d. Arguments

  e. Fields

  f. Statements

  g. Control Cards

  h. Program

48

49

51

51

53

55

61

62
### CHAPTER IV  TECHNIQUES FOR MODELING COMPLEX SYSTEMS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV.1  Introduction</td>
<td>73</td>
</tr>
<tr>
<td>IV.1  Modeling Technique</td>
<td>75</td>
</tr>
<tr>
<td>a. Underlying Systems</td>
<td>75</td>
</tr>
<tr>
<td>b. Abstraction Model</td>
<td>76</td>
</tr>
<tr>
<td>Set of n-tuples</td>
<td>76</td>
</tr>
<tr>
<td>Scalars</td>
<td>76</td>
</tr>
<tr>
<td>Subset of n-tuples</td>
<td>76</td>
</tr>
<tr>
<td>Translation</td>
<td>78</td>
</tr>
<tr>
<td>Substitution Rule</td>
<td>78</td>
</tr>
<tr>
<td>c. Digital Simulation Model</td>
<td>79</td>
</tr>
<tr>
<td>d. Procedure for Construction of a Digital Simulation</td>
<td>80</td>
</tr>
<tr>
<td>e. Application of a Model</td>
<td>81</td>
</tr>
<tr>
<td>IV.2  Model for a Clinical Laboratory</td>
<td>82</td>
</tr>
<tr>
<td>a. Isolation of Laboratory Functions</td>
<td>82</td>
</tr>
<tr>
<td>b. Parameters of Model</td>
<td>84</td>
</tr>
<tr>
<td>c. Simulation Output</td>
<td>85</td>
</tr>
<tr>
<td>IV.3  Digital Simulation of Laboratory</td>
<td>86</td>
</tr>
<tr>
<td>IV.4  Implementation</td>
<td>92</td>
</tr>
</tbody>
</table>
IV.5 Other Examples
   a. Example of Different Controlling Elements
   b. Aspect of System Not Directly Expressible in GPSS

IV.6 Validity of a Simulation Model

IV.7 Simulation of Medical Information System

CHAPTER V CONCLUSIONS AND PROPOSALS FOR CONTINUED RESEARCH

V.1 Results

V.2 Proposals For Continued Research.

APPENDIX A ALPHABETIC LISTING OF THE SYNTACTICAL SPECIFICATION OF GPSS III

APPENDIX B EXPERIMENTAL VERIFICATION OF THE SYNTAX OF GPSS III

B.1 Summary

B.2 Experimental Verification of Ambiguously or Incorrectly Stated Constructions in the GPSS manuals
   a. Syntactically Legal Program
   b. Symbol Table of Legal Program
   c. Assembled Legal Program
   d. Syntactically Illegal Program

APPENDIX C TABLE, DEMONSTRATING CONFIDENCE OF MODEL

APPENDIX D GPSS MODEL OF CLINICAL LABORATORY

LIST OF PRINCIPAL SYMBOLS AND ABBREVIATIONS

BIBLIOGRAPHY
FIGURES AND TABLES

Figure

I-1 A view of process involved in computer applications. 5

IV-1 A view of digital simulation. 73

IV-2 Underlying systems of complex systems. 75

IV-3 Underlying systems of a clinical chemistry laboratory. 83

IV-4 Simulation output 85a

IV-5 Overall view of the clinical chemistry laboratory (for handling blood samples). 91

IV-6 Utilization of computer time verses time of day. 108

D-1 Flow chart of a simulation of the clinical chemistry laboratory. 163

Table

IV-1 Comparison of actual laboratory events and times of occurrences with those simulated. 103
SUMMARY

One view of the process involved in most scientific computer applications is: starting with a problem, set up an abstraction of this problem, and implement this abstraction on a digital computer using a computer language. This dissertation is concerned with the application of this procedure to the digital simulation of complex systems such as a hospital, bank, or factory.

Chapters II and III are concerned with a major problem associated with the syntax of the computer languages used in this procedure, and more generally with the syntax of any computer language. As is evidenced by the works cited in Chapter II there has not been available prior to this writing a specification that has been used to characterize completely the syntax of computer languages. Syntax requirements, such as the correspondence between reference labels and statement labels, allowable column format, declaration of subscripted variables and the requirement that all statement labels of the same program be different, are examples of constructions that have not been defined in existing specifications. This investigation has applied known methods of recursive definition to the development of a complete syntactical specification of computer language.

A specification of the syntax of a limited portion of FORTRAN is presented. The restrictions that all statement labels must be different and all reference labels must correspond to statement labels in the same program is defined using this specification.

A complete syntactical specification of a computer simulation language, GPSS III, is presented in Chapter III and Appendix A. No other complete or accurate characterization of any form exists for the syntax of GPSS III. All syntactical constructions of GPSS III that were found by experiment to be incorrectly or ambiguously stated in the manuals are correctly defined by the GPSS specification presented in this dissertation. Some of the results of these experiments are discussed in Sections III.12-III.14 and Appendix B. In Appendix B experimental verification and examples of these constructions are also given.

In Chapter IV a way of viewing complex systems is presented. This viewpoint is composed of a set of fundamentals which are basic to complex systems and may be used to facilitate the understanding or simulation of these systems. These fundamentals form the abstraction in the view of the process involved in computer applications mentioned above.

Techniques have been presented in Chapter IV which may be used to implement this set of fundamentals on a digital computer.

A non-trivial, real and important example of the use of digital simulation is presented in Chapter IV. Results of a digital simulation of the Clinical Chemistry Laboratory of Yale-New Haven Hospital are discussed and the method used to obtain these results is discussed. Insight into the many problems of introducing data processing equipment into the laboratory has been obtained from the model.

A complete well documented model of the Clinical Chemistry Laboratory of Yale-New Haven Hospital is presented in Appendix D. The model may be used to study further simulation techniques or to understand the operation and procedures of the laboratory, or as an example of more conventional forms of the syntax of GPSS III then the examples given in Appendix B.

A Bibliography is presented in a manner that facilitates its use independently of the text of the dissertation.

July, 1966
CHAPTER I
INTRODUCTION

1.1 Preliminary Remarks

In recent years there has been a rapid growth in the use of digital computers for simulation work. The need for such simulation work has been generated by the ever-increasing complexity of systems that are being analysed and designed. The speed and capacity of modern digital computers have provided a means by which to expand simulation efforts.

1.2 Preliminary Definitions

Some terms which are used throughout this dissertation are defined as follows:

component: an entity (of a system) that may be independently identified and whose behavior is well-understood. Queues, equipment, or personnel may be components of systems.

system: an interacting or interdependent group of components forming a unified whole. The boundary of a system is that region in which the relationship between inputs and outputs of the system is considered to be dependent only on the system.

model: a representation of a system.

complex system: a system that contains large numbers of different components, variables, parameters, relationships, and events to which the system is responsive. A hospital, bank, or factory is a complex system.
**object system:** the system to be modeled, the system under study.

**simulation:** a process of constructing a dynamic representation of an object system. A simulation model is itself a system whose behavior may be interpreted according to a rule. The model together with the interpretation rule renders a description appropriate to the behavior of the object system.

**digital simulation:** a process of constructing a dynamic representation of an object system on a digital computer. A tool used in digital simulation is a computer language. A computer language has a **syntax** which is concerned with its form or grammar and a **semantics** which is concerned with its meaning.

**complete syntactical specification of a computer language:** a characterization that effectively defines all legal formations of strings of a computer language.

This usage of system and simulation terminology follows closely that of M.R. Lackner [1]. As noted by Jacobson [2] the term "computer simulation" under these definitions would mean a simulation of a computer. Other writers [3] have used "computer simulation" in the same sense that "digital simulation" is used above; however, this author agrees with Jacobson in that clarity suffers as a result.

I. 3 Digital Simulation

The process of constructing any representation or model of a system may be considered a simulation. With complex systems, however,
formal analytical or mathematical models often become difficult or even impossible to construct in terms of current mathematical knowledge. Digital simulation provides another method of constructing a model. Digital simulation may be used as an alternative to other simulation methods, as is the case when formal mathematical methods are difficult or impossible to apply. Or digital simulation may also be used in preference to other methods because of the ease by which a digital simulation model may be constructed, modified, or studied.

I.4 Developments in Simulation

One of the most significant developments in simulation has been the construction of simulation languages for computers. Many such languages have been constructed: DYNAMO, Forrester; GPSS, Gordon; SIMSCRIPT, Markowitz, Housner, and Karr; Control and Simulation Language, Laske and Buxton; General Simulation Program, Tocher and Owen; SIMPAC, Lackner; BLOODI, Kelly, Lochbaum,Vyssotsky. A history of some of these languages is found in [4]. All of these languages attempt to facilitate the accurate construction of system models.

I.5 Needs and Deficiencies

Because a language is a means of expressing thoughts or concepts, the effective use of a language for the construction of a model requires understanding and development of general concepts about systems to be simulated. If such thoughts or concepts are dependent on a particular
language many important aspects of a system may be missed because they are not directly expressible in the language.

As pointed out by Lackner [5] "A model is a caricature. Certain phenomena are exaggerated in a simulation, while others are ignored. The modeler attempts to characterize phenomena, including those things he judges important, while excluding those he judges insignificant." The exclusion or exaggeration of phenomena should be the modeler's prerogative and not a consequence of limitations of a computer language he is using.

Deficiencies in the available methodology of simulation are brought out by McMillan and Gonzales [6]. They cite a book by Sprowls [7] who notes that two analysts (C. P. Bonini and E. B. Roberts) conducted experiments with two decision-information models of a system in which both specified similar starting conditions but obtained contradictory results. Bonini defined a highly variable environment in terms of customer orders and market growth trend, and his model produced lower costs and higher sales and profits. Roberts, using Forrester's techniques, observed that the same situation led to great fluctuations of inventory, back orders, and employment and, therefore, fluctuations in profits.

With this example in mind we come back to the realization that in order to construct models which are more comprehensive, realistic and dynamic, and which duplicate the behavioral aspects of systems,
we need modeling techniques which are more powerful than those used in the past.

Another need arises from the large number, in many cases, of man hours involved in constructing a digital simulation model. It is not uncommon that by the time an accurate model has been constructed the need for such a model no longer exists. We need modeling techniques that speed up the process of constructing a model.

A computer language, usually a simulation language, is used to implement a model on a computer. With any language there is a syntax and semantics. We have discussed some needs and deficiencies associated with the use and semantics of computer languages used for simulation. There is also a major deficiency associated with the syntax of these languages and, more generally, with the syntax of any computer language. The deficiency is that there exists no known specification that has been used to characterize completely the syntax of programming languages.

I.6 Purpose of Dissertation

One view of the process involved in computer applications is depicted in Figure I-1.

![Figure I-1 A view of process involved in computer applications.](image)
A simulation of a system described by a set of differential equations may follow this process. The set of differential equations form the abstraction. These equations may be implemented on an analogue computer. They may also be implemented on a digital computer using numerical techniques and an algorithmic computer language, or using a language such as MADBLOCK, MIDUS, PACTOLUS, or BLOODI which is especially suited for simulation of systems that may be described by differential equations.

A system such as a hospital, bank or factory can not be directly described by a set of differential equations. These systems are composed of people, machines, queues, or other components which have quite different characteristics and which have complex relationships among these components. Such systems have many events to which they are responsive. Such systems are called complex systems.

This dissertation is concerned with the application of the process depicted in Figure I-1 to the digital simulation of complex systems.

The major purposes of this dissertation are the following four:

First, to present a way of viewing complex systems. This viewpoint is composed of a set of fundamentals which are basic to complex systems and may be used to facilitate the understanding or simulation of these systems. The set of fundamentals form the abstraction of Figure I-1.
Second, to present general techniques for implementing this abstraction on a digital computer. These techniques may be expressed in most computer languages.

The computer languages used to express these techniques have a syntax. A third purpose of this dissertation is to apply known methods of recursive definition to the development of a complete syntactical specification of computer languages. Fourth, to present a complete syntactical specification of a computer simulation language, GPSS III.

I. Contributions

The following are believed to be the major contributions of this dissertation:

Section III. 1 - III. 7 The application of known methods of recursive definition to the development of a complete syntactical specification for computer languages. There exists no known specification that has been used to characterize completely the syntax of programming languages.

The specification may be used by a programmer to check the syntactical validity of a string. The specification is more compact, concise, and accurate than a manual.

The specification may be used to define all acceptable strings of a proposed compiler. Hence, decisions affecting membership in the class of legal programs are not made during the construction of the compiler. In this manner the specification may assist the
compiler writer.

A compiler may be viewed as having two phases, a syntax phase and a translation phase. The syntax phase is concerned with recognizing the well-formed strings of its source language. The specification may be used in the construction of compilers whose syntax phase is independent of the source language being translated. That is, several source languages may be used with the same compiler.

III.2, III.8, III.9 A specification of a limited portion of FORTRAN is presented. The interesting constructions that are defined for the limited portion of FORTRAN are the following two: 1. All statement labels in the same program must be different. 2. All reference labels must correspond to statement labels in the same program.

III.12-III.14, Appendix A A complete syntactical specification of GPSS III, a simulation language, has been developed. No other complete or accurate characterization of any form exists for the syntax of GPSS III. Sections III.12 through III.14 present the development and motivation of the specification. Appendix A presents the specification in a form that facilitates its use as a syntax reference for GPSS III.

Appendix B Experimental verification is given of some of the syntactical constructions of GPSS that are inaccurately or ambiguously
stated in the GPSS manuals [9, 10] but are correctly defined in Appendix A.

IV. 2 - IV. 9 A way of viewing complex systems is presented. This viewpoint is composed of a set of fundamentals which are basic to complex systems and may be used to facilitate the understanding or simulation of these systems.

IV. 11 Techniques are developed for implementing this set of fundamentals on a digital computer.

These fundamentals and techniques are not dependent on a specific computer language, or system, and so do not suffer the limitations, discussed in Section I. 5, associated with such a dependence. Since these fundamentals and techniques are general, a modeler using them need not confront a completely new problem with each system simulated; hence, a savings in the number of man hours involved can be obtained.
Appendix D  In this appendix is presented a version of the GPSS program that was used to implement a model of the Clinical Chemistry Laboratory of Yale-New Haven Hospital on a digital computer. The program is well documented. The program may be used to study further the simulation techniques presented in Chapter IV; or to study the organization and procedures of a clinical chemistry laboratory as are briefly discussed in Chapter IV; or to be used as a reference of more conventional examples of the syntax of GPSS than those presented in Appendix B. The form of all of the constructions used in the program given in this section have been defined in Chapter III and Appendix A.

Bibliography
A Bibliography is presented which is a survey of the salient existing works which are related to this dissertation. The Bibliography may be used independently of the text of this dissertation. With each related group of references a brief note has been added stating the principal result of those works and their relationship to the work done in this dissertation.
CHAPTER II
SYNTAX SPECIFICATION - A SURVEY

II.1 Introduction

This chapter and Chapter III are concerned with a major problem associated with the syntax [8] of computer languages used for simulation, and more generally with the syntax of any computer language. The problem is that there exists no known specification that has been used to characterize completely the syntax of programming languages.

The major purpose of this chapter and Chapter III is twofold: First, to apply known methods of recursive definition to the development of a complete syntactical specification for computer languages. Second, to present a complete syntactical specification of a computer simulation language, GPSS III [9, 10]. No other complete or accurate characterization of any form exists for the syntax of GPSS.

The specification may be used by a programmer to check the syntactical validity of a string. An example of this use is given in Section IV.13. The specification is more compact, concise, and accurate than a manual.

The specification may be used to define all acceptable strings of a proposed compiler. Hence, decisions affecting membership in the class of legal programs are not made during the construction of the
compiler. In this manner the specification may assist the compiler writer.

A compiler may be viewed as having two phases, a syntax phase and a translation phase. The syntax phase is concerned with recognizing the well formed strings of the compiler's source language. The specification may be used in the construction of compilers whose syntax phase is independent of the source language being used. Using a specification, Irons [11], "syntax directed compilation," attempted to construct a compiler in this manner.

II.2 Survey

One of the most significant developments in the study of computer languages has been an attempt to formalize portions of their syntax. Some of the most prominent works are: Backus [12]; Knuth [13]; Burkhart [14]; Iverson [15]; Berman, J. Sharp, L. Sturges [16]; Huskey, Love, and Wirth [17]; Naur [18]; Rabinowitz [19]; Shaw [20]; Taylor, Turner and Waychoff [21]; Wirth [22]; [23]; Brown [24]. In these works the following languages have had portions of their syntax formalized:

ALGOL, JOVIAL, FORTRAN, NELIAC, APT, COBOL, BALGOL

(see references 24-33 for exposition of these languages).

Besides greatly improving communication between people, formalized syntax has led to new results in the theory and practice of programming. As early as 1963, Irons [11] was able to construct a compiler whose syntax phase was independent of the source language
being translated. That is, the same compiler could be used with several source languages. This work and works of other contributors, such as Brooker and Morris [34]; Cheatham and Sattley [35]; Conway [36]; Irons [11, 31]; Ledley and Wilson [38]; Mayoh [39]; Warshall [40], led to speculation that the entire compilation process could be made independent of the source language.

Formalization of syntax has also encouraged research in the general theory of phrase structure grammars [41-54], and the theory of languages and automata [55-61]. These writers have presented proofs of insolubility and ambiguities associated with phrase structure grammars. They have studied the construction of parsing systems. The most valuable theoretical insight gained into these diverse systems is the realization that their diversity is, in a sense, only superficial. Many of these systems have been shown to be equivalent to context-free phrase structure grammars, e.g. predictive grammars, dependency grammars have been shown equivalent. These writers have used idealized languages with the hope that results will be applicable to computer languages.*

All formulations of syntax previously mentioned are equivalent to context-free phrase structure grammars [46,47] as defined below.

Approaches to formalization of the semantics of programming language have been made. Some of the most prominent are: Wijngaarden [62]; Feldman [63]; Wirth and Weber [64]. However, if the semantics is

* The term computer language refers to practical as opposed to theoretical computer language.
formalized then the syntax must be formalized. All of these works use a context free phrase structure grammar [46,47] to specify syntax.

II. 3 Phrase Structure Grammars

A representative use of a context free phrase structure grammar has characterized a portion of the syntax of ALGOL 60 [18,28,33]. The form for grammatical rules used in the report [12] which officially defines ALGOL 60 is typified by:

\[
\text{(identifier)} := (\text{letter}) \text{ or } (\text{identifier})(\text{letter}) \text{ or } (\text{identifier})(\text{digit})
\]

This assertion can be read "An identifier is defined as a letter or an identifier followed by a letter or an identifier followed by a digit." This assertion may be considered a substitution rule in that the string to the right of ":=" may be substituted for the string to the left.

In a phrase structure grammar every rule acts like a substitution rule. Applying these substitution rules repeatedly to the string designating the syntactic name "program" or "sentence", we eventually arrive at a sequence of symbols in which no further substitution can be made. This sequence of symbols is a program or sentence in the language represented by the grammar. The process by which this sequence of symbols was produced is called a program derivation or sentence derivation.

The symbols designating syntactic names (e.g. program, sentence) for which substitution may be made, are called non-terminal characters; those undefined symbols which form the sentence or program are the
The following is an example of a context free phrase structure grammar. The words sentence, noun, predicate, verb are non-terminal characters. The symbols JOHN, BALL, HIT are terminal characters.

a. (sentence) := (noun)(predicate)

b. (predicate) := (verb)(noun)

c. (noun) := JOHN or BALL

d. (verb) := HIT

The sign := may be read "replaced by".

Successive substitutions starting with (sentence) gives the sequence

1. (sentence)

2. (noun)(predicate)

3. BALL (predicate)

4. BALL (verb)(noun)

5. BALL (verb) JOHN

6. BALL HIT JOHN

Statements 1-6 form a sentence derivation.

The above is a context free phrase structure grammar. Such a grammar is sometimes referred to simply as a phrase structure grammar [58]. However, this leads to the confusion of whether a context free or context sensitive phrase structure grammar is meant. In this work the term phrase structure grammar will be used to refer
to both context free or context sensitive. Otherwise it will be stated which is meant.

The general form of rules of a phrase structure grammar is

\[ \psi_1 \omega \psi_2 \rightarrow \psi_1 A \psi_2 \]

where \( \psi_1, \omega, \psi_2, A \) are strings.

We call a rule context free if \( \psi_1 = \omega = \psi_2 = \) null string, and otherwise context sensitive.

We call a phrase structured grammar context free if all its rules are context free, and otherwise a context sensitive phrase structure grammar [65].

We call a language a context free language if all of its strings can be generated by a context free phrase structure grammar.

We call a language a context sensitive language if all of its strings can be generated by a context sensitive phrase structure grammar.

II.4 Shortcomings and Difficulties of Past Specifications

Most attempts to specify the syntax of computer languages have used grammars which are equivalent to context free phrase structure grammars. Discrepancies between the syntax of the language specified and the syntax of computer languages arise because many computer languages are not context free (see reference Gilbert [65] page 90, Floyd [58] page 350, Diforino [66]). Further discrepancies arise because many computer languages have restrictions on column format and length of numbers or statements. These restrictions have not been characterized.
There exists no known specification that has been used to characterize completely the syntax of programming languages.

The representations previously mentioned are useful. However many constructions of computer languages are difficult or impossible to express in them. The following are examples of such constructions.

The statement GO TO 20 in a FORTRAN program or TRANSFER\textsuperscript{TO} 20 in a MAD program is not syntactically correct unless the reference label 20 corresponds to a statement label 20 in the same program. Likewise the reference label of a DO statement of FORTRAN and a WHENEVER statement of MAD must correspond to a statement label in the same program. All reference labels of MAD or FORTRAN must have corresponding statement labels in the same program.

All statement labels in a MAD or FORTRAN program must be different.

All subscripted variables must be declared by a declaration statement in FORTRAN or MAD programs.

These restrictions are syntactical in that programs using these constructions incorrectly are never executed but are rejected solely on their form.

There are other syntactical properties that have not been characterized in any known representation. Most computer languages have stringent restrictions on column format. All statements using card input in FORTRAN must start after column six. Most computer
languages do not allow statements of unlimited length. In a character-
ization of a language using card input each card must have eighty columns.
Some languages require special card formats to indicate a continuation
of a statement to another card. These constructions have not been
characterized in known specifications.

Bases of Formalization

The formalization presented here evolved from Post's canonical
systems(69) and Smullyan's elementary formal systems(68). In various
class notes on the application of elementary formal systems to the
definition of self-contained mathematical systems (75) Trenchard More
modified the definition of elementary formal systems so as to parallel
his definition of propositional complexes(67). I started to apply the
modified elementary formal systems to the recursive definition of the
context-free and context-sensitive parts of the syntax of programming
languages. As it became clear that the definitional needs of mathematical
systems and programming languages were the same in requiring many conclusions
from the same premisses, More made further modifications to produce the
definition of a canonic system. The canons or axioms of the system have
conjunctions of conclusions rather than single conclusions or disjunctions
of conclusions.

The example used to present the development of the specification is
the syntax of a limited portion of FORTRAN. The interesting constructions
that are defined for the example language are the following two: 1. All
statement labels in the same program must be different. 2. All reference
labels must correspond to statement labels in the same program. Following
this development, portions of the system that define the syntax of the
specification of limited FORTRAN are presented.
An example of the use of the specification of GPSS III to check the legal formations of strings of this language is given. The motivation and development of a complete syntactical specification of GPSS III is given in Chapter III. Appendix A presents the specification in a form that facilitates its use as a syntax reference for GPSS III. A partial list of constructions is presented in Appendix B that are inaccurately or ambiguously stated in the GPSS manuals but are correctly defined in the specification. Experimental verification of these constructions is also presented in Appendix B.
CHAPTER III
A RECURSIVE SPECIFICATION
FOR COMPUTER LANGUAGES

III, 1 Canonic Systems

A language consists of certain signs, and certain strings of these signs. Its syntax consists of rules for classifying and transforming these strings. The alphabet of a language consists of certain basic signs. A string is a sequence of signs [70].

A canonic system is a language that can recursively define the syntax of a second language. A canonic system may thus be defined by another higher level canonic system which in turn is defined by an even higher canonic system. This series of canonic systems starts with the canonic system that defines the language understudy (object language). The constructions involved in forming these higher level canonic systems soon repeat.

A canonic system consists of a number of canons. A canon is a rule which recursively classifies strings.
Canonic systems will be developed, by example, that facilitate the complete syntactical definition of computer languages. A canonic system that completely defines the syntax of a limited portion of FORTRAN, called limited FORTRAN, will be used as an example. Following this development, portions of the canonic system that defines the syntax of the canonic system that defines limited FORTRAN will be discussed. The remainder of the chapter will be concerned with the syntactical specification of GPSS III.

III. 2 Limited FORTRAN

labels: All labels will be single digits. For brevity only three digits will be considered: 1, 2, 3.

All statement labels in a program must be different.

All reference labels in a program must have corresponding statement labels.

statements: There are two permissible statements "I=I+ some number" and "GOTO some reference label". All statements need not have statement labels. The number in the statement "I=I+ some number" must be a fixed point number. Numbers are concatenations of digits.

card format: All statement labels must appear in column two. All statements must appear in column seven through eighty. All statements must start in column seven. At the end of a statement blanks fill in the remaining columns, if any, until the end of a card. No unspecified
spaces are permitted, i.e. GO TO2 is not legal, GOTO2 is legal.

Programs written in limited FORTRAN will be compiled and executed by a FORTRAN compiler. An example of a syntactically legal program of limited FORTRAN is given below.

```
3  I=I+1
   I=I+2
   GOTO3
1  I=I+2213111231223
```

The execution of the above program results in a permanent loop and the statement labeled 1 is never executed. The program's meaning or its semantical legality might be questioned. However, syntactically it is a legal program. A programmer may wish to determine how long a 7094 will run before making an error. We will not deny him this test.

III. 3 Conventions

Because it is very difficult for the reader to count blank spaces a bold face or twice underlined number will be used, e.g. A$B denotes the letter A followed by four spaces followed by the letter B. Therefore A$B is the same string as AB. The sign A denotes the null string, e.g. AAB and AB are the same string. A # will be used to denote the end of a card. Thus the example program above may be written as one string.

```
134I=I+169#6I=I+269#6GOTO369#
114I=I+221311123122357#
```
III. 4 Development of Canonic Systems for Computer Languages

Canonic systems will be developed, by example, which facilitate the complete syntactical definition of computer languages. A canonic system of limited FORTRAN will be developed. The entire canonic system of limited FORTRAN is presented in Section III-9.

The alphabet of limited FORTRAN consists of the following signs: 1 2 3 G O T O = + I. The purpose of the canonic system is to recursively classify strings over this alphabet. A canon of the canonic system recursively classifies one set of strings. A predicate is a name given to a well defined set of strings over the alphabet of the object language. For a programming language these sets are defined in such a way as to aid the user of the canonic system. The first classification to be defined for the syntax of limited FORTRAN is the set of digits. The canons below accomplish this. The sign ⊢ is an assertion sign.

\[ \begin{align*}
\vdash_1 \text{digit} \\
\vdash_2 \text{digit} \\
\vdash_3 \text{digit}
\end{align*} \]

These canons are read: "from no premises it can be asserted that 1 is a digit", "from no premises it can be asserted that 2 is a digit", "from no premises it can be asserted that 3 is a digit".

digit is a predicate that names the set of strings "1" "2" "3".

Note in the above canon there is a blank space between the digit 1 and the predicate digit. The blank space is there for ease of reading.
The space is associated with the predicate and not with the term 1, i.e. 1 followed by a space is not a digit, 1 is a digit. A blank space immediately to the right or left of all predicates is associated with the predicate. A blank space immediately to the right of all punctuation signs | ; _ will be associated with the punctuation sign.

The next set of strings defined is named number. The following two canons define the set number.

\[
d \text{digit} \mid d \text{number}
\]

\[
d \text{digit}; n \text{number} \mid n \text{nd number}
\]

The first canon above is read, "If the premise 'd is a digit' is assumed, then it is asserted that d is a number". The second canon is read, "If the premises 'd is a digit' and n is a number are assumed, then it can be asserted that n followed by d is a number".

The assertion sign \( \vdash \) is used to separate the premises from the conclusions. The general form of a canon is

\[
a; b; c; \ldots \vdash w; x; \ldots y; z
\]

and is read "If the premises a and b and c and ... are assumed then it can be asserted that w, x, ... y, z.

III. 5 Predicate, Variable, Term, Remark, Canon

The words digit, number, list, program, filler, card are predicates of limited FORTRAN that name certain sets of strings. The words differ, not in, in are predicates of degree two. A predicate of degree two names an ordered set of pairs of strings, e.g. the pair of strings
"1" "2" is a member of the set of strings named by differ. A predicate of degree three names an ordered set of strings whose members are triples. The predicate... statement labels in... with reference labels... is a predicate of degree three in the canonic system of limited FORTRAN. A predicate of degree n names an ordered set of n-tuples.

The small script letters x, y, p, d, n, c, z are variables. A term is a series of signs taken from the alphabet of the object language and/or variables, e.g. the string d and the string GOTOd are terms. A remark is a concatenation of terms with predicates in the following way: If P₁ is a predicate of degree one and t₁ is a term then t₁P₁ is a remark. If P₂ is a predicate of degree two and if t₁ and t₂ are terms then t₁P₂t₂ is a remark. If the string P₃aP₃b is a predicate of degree three and if t₁, t₂ and t₃ are terms then t₁P₃at₂P₃bt₃ is a remark. Thus x statement labels in p with reference labels h is a remark.

If R₁ R₂ ... Rₙ are remarks then R₁;R₂; ... ;R₁;R₁+1; ... ;Rₙ is a canon. R₁ R₂ ... Rₙ are premises. R₁+1... Rₙ are conclusions. The canon is read, "If the premises R₁ and R₂ and R₃ and ... and Rₙ are assumed then it can be asserted that R₁+1 or ... or Rₙ".

III. 6 Recursion

The recursive property of a canonic system must be thoroughly understood. The canons:

(1) \[ \vdash 1 \text{ digit} \]

(2) \[ \vdash 2 \text{ digit} \]
(3) \( \vdash 3 \text{ digit} \)

(4) \( d \text{ digit} \vdash d \text{ number} \)

(5) \( d \text{ digit}; n \text{ number} \vdash n \text{ number} \)

define a set called number, of strings by recursion. For example, the string 222 is a member of this set and may be recursively generated using the above five canons. First, canons two and four are used to obtain the conclusion "2 is a number". Next, letting \( d \) be 2 and \( n \) be 2 in canon five the conclusion "22 is a number" is obtained. Again, using canon five and letting \( d \) be 2 and letting \( n \) be 22 (since we know from above that 2 is a digit and 22 is a number) the conclusion "222 is a number" is obtained.

III. 7 Edicts

The redundancy among the form of the canons above strongly suggest the use of some abbreviations. An edict is an abbreviation of one or more canons.* There are three constructions that form edicts:

1. The canons

\[ \vdash 1 \text{ digit} \]

\[ \vdash 2 \text{ digit} \]

\[ \vdash 3 \text{ digit} \]

form the edict

\[ \vdash 1, 2, 3 \text{ digit} \]  

(III-1)

In general, if \( t_1 \ldots t_n \) are terms and if \( P_k \) is a predicate of degree one then the remarks: \( t_1 P_k t_2 P_k t_3 P_k \ldots t_n P_k \) are abbreviated

\[ t_1, t_2, t_3, \ldots , t_n P_k \]  

The sign \( \downarrow \) is a punctuation sign.

* The carrot \( \downarrow \) notation was suggested by Trenchard More in response to the frequent occurrence of having many terms specified by the same predicate in the specification of GPSS III.
2. Premises containing variables are used with conclusions containing the same variables. This relation between premises and conclusions will be called relevance.* Thus canons

\[
d \text{digit} \vdash d \text{number}
\]

\[
d \text{digit}; n \text{number} \vdash \text{nd number}
\]

form the edict

\[
d \text{digit}; n \text{number} \vdash d_1 \text{nd number}
\]

(III-2)

The conclusion of edict (III-2) "d \_ \text{nd number}" follows the construction of (1) above, i.e. "d \_ \text{nd number}" is an abbreviation of the two remarks "d \text{number}" and "nd \text{number}". The edict contains two premises. Only one of them, "d \text{digit}" is relevant to the conclusion "d \text{number}" since the variable d is the only variable appearing in the conclusion and d only appears in the first premise. The second conclusion "nd \text{number}" has two variables and these appear in both premises. Therefore, both premises are relevant to the conclusion "nd \text{number}".

3. It is often the case that canons have some of the same premises. The premises \( F_1, \ldots, F_n \) and the premises \( G_1, \ldots, G_m \) and \( M_1, \ldots, M_0 \) are common to the following canons in the manner shown below.

\[
F_1; \ldots; F_n; A_1; A_2; \ldots; A_{a-1} \vdash A_a; \ldots; A_b
\]

\[
F_1; \ldots; F_n; B_1; B_2; \ldots; B_{c-1} \vdash B_c; \ldots; B_d
\]

\[
F_1; \ldots; F_n; G_{j_1}; \ldots; G_{m_j}; X_1; \ldots; X_{e-1} \vdash X_e; \ldots; X_f
\]

\[
F_1; \ldots; F_n; G_{j_1}; \ldots; G_{m_j}; Y_1; \ldots; Y_{g-1} \vdash Y_g; \ldots; Y_h
\]

\[
F_1; \ldots; F_n; G_{j_1}; \ldots; G_{m_j}; M_{j_1}; \ldots; M_{j} \vdash Z_1; \ldots; Z_{i-1} \vdash Z_i; \ldots; Z_{j}
\]

* This contraction of canons was suggested by Trenchard More.
These canons are abbreviated to form the following construction
where $F_1; \ldots; F_n$ are taken to be the first premises of all conclusions,
$G_1; \ldots; G_m$ are taken to be the second premises and so on until the
words "end of premises".

\begin{itemize}
  \item \textbf{first premises} \hspace{1cm} F_1; F_2; \ldots; F_1; \\
      \hspace{1cm} A_1; A_2; \ldots; A_{a-1} \vdash A_a; \ldots; A_b \\
      \hspace{1cm} B_1; B_2; \ldots; B_{c-1} \vdash B_c; \ldots; B_d \\
  \item \textbf{second premises} \hspace{1cm} G_1; G_2; \ldots; G_m; \\
      \hspace{1cm} X_1; \ldots; X_{e-1} \vdash X_e; \ldots; X_f \\
      \hspace{1cm} Y_1; \ldots; Y_{g-1} \vdash Y_g; \ldots; Y_h \\
  \item \textbf{third premises} \\
  \item \textbf{end of premises}
\end{itemize}

III. 8 Development of Canonic System for Limited FORTRAN

This section is concerned with the development and motivation of
a canonic system for limited FORTRAN. The entire canonic system
will be presented in Section III. 9.

The sets of strings named in, differ, not in will be defined to express
the constructions arising from the limitations placed on labels
(Section III. 2). The restriction that all reference labels must also
occur as statement labels makes the predicate in convenient. * If a
string of statements is to be a program then the list of reference labels

* Convenient, not necessary, for there are many ways in which a
canonic system defining limited FORTRAN may be constructed. A way
has been chosen that facilitates the use of the system by programmers
to check legal constructions of limited FORTRAN.
must be in the list of statement labels.

A list of labels is a series of digits separated by commas, e.g. 1, 2, 3, is a list. The edict

\[
d \text{ digit}; x \text{ list} \vdash \Lambda, \text{ list}
\]

(III-3)
defines the set of strings named list.

The edicts

\[
x \text{ list} \vdash \Lambda \in x
\]

(III-4)
\[
x, y \text{ list}; c \in xy; d \text{ digit} \vdash \text{ cd, in xd, y}
\]

(III-5)
define the set of all ordered pairs of strings that are named in. Edict (III-4) is read, "If the premise 'x is a list' is assumed then it can be asserted that the null string is in x". Edict (III-5) is read, "If the premises 'x is a list' and 'y is a list' and 'c is in the string xy' and 'd is a digit' are assumed then it can be asserted that the string c followed by d followed by , is in the string x followed by c followed by , followed by y".

The restriction that all statement labels must be different makes the definition of the sets called differ and not in convenient. A program is a series of statements some of which have statement labels. If a statement with a statement label is added to a program its statement label must not be in the list of statement labels previously associated with that program.

The canons

\[
\vdash 1 \text{ differ } 2
\]

(III-6)
\[
\vdash 1 \text{ differ } 3
\]

(III-7)
\[ \vdash 2 \text{ differ } 3 \]  
\[ c \text{ differ } d \vdash d \text{ differ } c \]  
(III-8)  
(III-9)

define the set of all ordered pairs of digits that are different.

The canons

\[ d \text{ digit } \vdash d \text{ not in } \Lambda \]  
(III-10)

\[ d \text{ differ } x; d \text{ not in } y \vdash d \text{ not in } yx, \]  
(III-11)

define the set of ordered pairs of digits and lists for which the digit
is not in the list.

A program is defined in much the same way as a programmer may
construct one. That is, he writes a series of statements keeping track
of statement labels and reference labels. With each added statement he
checks to make certain that it has not been used as another statement
label in the same program. Finally he checks to see if all reference
labels correspond to statement labels.

A predicate of degree three will name the set of triples \((s, p, r)\)
where \(s\) is a list of statement labels, \(p\) is a string of statements with
or without statement labels, \(r\) is a list of reference labels. The name
of this set is the predicate \(\ldots \text{ statement labels in } \ldots \text{ with reference}
\text{ labels } \ldots \).

To start the recursion correctly \(s, p,\) and \(r\) must denote null strings.

Thus we state the cannon

\[ \vdash A \text{ statement labels in } \Lambda \text{ with reference labels } \Lambda \]   
(III-12)

The following canons complete the definition of the set of triples
called ... statement labels in ... with reference labels ...

\[ \text{first premises} \]

\[ s \text{ statement labels in } p \text{ with reference labels } r; \]
\[ n \text{ number}; \]
\[ \vdash s \text{ statement labels in } p \&I=I+n# \text{ with reference labels } r \]
\[ (\text{III-13}) \]
\[ d \text{ not in } s \vdash sd, \text{ statement labels in } p\&d4I=I+n# \text{ with reference labels } r \]
\[ (\text{III-14}) \]
\[ d \text{ digit } \vdash s \text{ statement labels in } p\&GOTOd# \text{ with reference labels } rd, \]
\[ (\text{III-15}) \]
\[ c \text{ not in } s; d \text{ digit } \vdash sc, \text{ statement labels in } p\&c4GOTOd# \text{ with reference labels } rd, \]
\[ (\text{III-16}) \]

\[ \text{end premises} \]

For a programmer to use these canons effectively as a reference of the syntax of a language he usually must go through one recursion to see the constructions involved. Take for example edict (III-16) above and examine some of its conclusions. Edict (III-16) is an abbreviation of the canon

\[ s \text{ statement labels in } p \text{ with reference labels } r; c \text{ not in } s; d \text{ digit } \]
\[ \vdash sc, \text{ statement labels in } p\&c4GOTOd# \text{ with reference labels } rd, \]

The first recursion must have \( s, p, \) and \( r \) denote the null string as is defined by canon (III-12). \( c \) may be any digit, say 2, as is defined by (III-1), (III-10). From edict (III-1) \( d \) may denote 3. Thus a conclusion to edict 16 is
2, statement labels in 124GOTO3# with reference labels 3, 

\[ \] 

\[ s \quad p \quad r \]

Using this conclusion, 2 may now denote "2,", p may denote "124GOTO3#", r may denote "3,", using canons (III-7), (III-9), (III-10), (III-11) c may denote "3," (i.e. 3, not in 2,), and from edict (III-1) d denotes 1.

The conclusion to edict (III-16) is now

2, 3, statement labels in 124GOTO3#134GOTO1# with reference labels 3, 1, 

\[ \] 

\[ s \quad p \quad r \]

If this process is repeated s will build up to include all statement labels in the string p and r will build up to include all reference labels in p.

Among the members of the set of triples (s, p, r) only a few satisfy the requirement, that all reference labels appear as statement labels in the string p. To define this subset another canon is used

\[ s \text{ statement labels in } p \text{ with reference labels } r; r \text{ in } s \implies \text{program} \]

(III-17)

For the purposes of the programmer using this specification to help him check legal strings, limited FORTRAN is essentially defined by the edicts and canons (III-1) through (III-17) previously discussed. However, one aspect of the language has not been defined.

Every statement must appear on a card that is 80 columns long. In the specification of limited FORTRAN as defined by canons and edicts
(III-1)-(III-17) no account of number of columns on a card has been made (all other aspects of card format have been defined). The correct number of blanks, if any, have not been filled in between the end of a statement and the end of a card mark #.

This aspect of a programming language was not defined as the canonic system of limited FORTRAN was built so as not to complicate the specification. However this aspect is part of the syntax of a language and can not be omitted from a complete syntactical specification. It is defined below.

A filler is the null string or a string of blank spaces.

\[
x \text{filler} \triangleright A_1 x_1 \text{filler}
\]  

(III-18)

Fillers will be inserted into members of the set of strings called program, edict (III-17). The resulting set of strings will be called any length card program. The canon below defines this set.

\[
a\#b \text{ program; } f \text{ filler } \triangleright A_1 a\#b \text{ any length card program}
\]

(III-19)

Note, because of the way in which a program is built up (canons and edicts (III-12) - (III-17)) the variable b in the first premise may denote the null string. However a may not denote the null string since # by itself is not a program.

There is a subset of the set of strings called any length card program that are composed of strings of length 80 characters separated by #. This set is defined using the following four edicts.

\[
\frac{1}{x_t} 1_t 2_t 3_t G_t O_t T_t O_t =_t +_t \text{ character space}
\]

(III-20)
The set \textit{card program} contains members that are concatenations of strings of 80 characters separated by \#. The intersection of the set \textit{card program} and any \textit{length card program} defines the set of all legal programs of limited FORTRAN that contain strings of 80 characters separated by \#. This set is defined by the canon.

\begin{equation}
\text{p any length card program; p card program } \vdash \text{ p legal card program}
\end{equation}

The canons and edicts (III-1) through (III-23) completely define the syntax of limited FORTRAN.

\textbf{III. 9 Canonic System of Limited FORTRAN}

\begin{align*}
digit & \vdash 1, 2, 3 digit & 1 \\
number & \vdash d \ digit; n \ number \vdash d, nd \ number & 2 \\
list & \vdash d \ digit; x \ list \vdash A, xd, list & 3 \\
in & \vdash x \ list \vdash A \ in x & 4 \\
x, y \ list; c \ in xy; d \ digit \vdash cd, in \ xd, y & 5 \\
differ & \vdash 1 \ differ 2 & 6 \\
& \vdash 1 \ differ 3 & 7 \\
& \vdash 2 \ differ 3 & 8 \\
c \ differ \ d \vdash d \ differ \ c & 9 \\
not \ in & \vdash d \ digit \vdash d \ not \ in \ A & 10 \\
& \vdash d \ differ \ x; d \ not \ in \ y \vdash d \ not \ in \ yx, & 11
\end{align*}
...statement labels

in... with reference labels

first premises

1 number: — s statement labels in p with reference labels r;

not in s f s statement labels in p+d4I=I+n# with reference labels r

d not in s f sd, statement labels in p+d4I=I+n# with reference labels r

d digit s statement labels in pGOTOd# with reference labels rd,

c not in s; d digit s sc, statement labels in p+cGOTOd# with reference labels rd,

end premises

program

s statement labels in p with reference labels r;

in s; p program

filler

x filler |- p program

any length card

af#b any length card

program

character space

T i 1 2 3 G t O t T t = t + t I character space

card

a 1 a 2 a 3 . . . a 80 character space a a a

. . . a 80 card

character space

c a card; x card program |- p program

any length card program

p any length card program; p card program

p legal card program
III. 10 Canonic System of the Canonic System of Limited FORTRAN

Starting with the canonic system that defines limited FORTRAN there is an unending series of canonic systems each of which defines the one below it. In Sections III. 4 through III. 9 we have defined the canonic system of limited FORTRAN in English. However, this canonic system could have been defined by another canonic system. This section is concerned with a partial development of the canonic system (called the second canonic system) of the canonic system of limited FORTRAN. Enough of the second canonic system is developed to define precisely the set of strings whose members are all the strings of limited FORTRAN and their productions. This set is called derivation. Canons of the second canonic system are called statutes.

The words canon, remark, term, derivation, amid, conclusion do not appear in the canonic system of limited FORTRAN (Section III. 9). These words define strings of the canonic system of limited FORTRAN. They are predicates in the second canonic system.

We have defined in English the sets of strings named remark and term (Section III. 5). The predicate canon is the name of the set of all canons and edicts changed to canons of the canonic system of limited FORTRAN (Section III. 9). The predicate conclusion is the name of all remarks that appear to the right of | in a canon. A remark is amid a list of remarks if and only if it is contained in that list.

The following two statutes of the second canonic system may now be stated. The sign \[\mathbb{\text{Ω}}\] is the assertion sign of the second canonic system.
The same rules as were stated for canons apply to statutes.

Statute (III-24) above is read "It can be asserted from no premises that the sign \( \vdash \) followed by the null string is a derivation". Statute (III-25) is read "If the premises \( rf \) is a derivation and \( c \) is a conclusion and \( gc \) is a canon are assumed, and if \( g \) is amid \( f \) then it can be asserted that \( \vdash fc \) is a derivation."

Statute (III-24) is necessary to start the recursion. Statute (III-25) may be interpreted as saying that if we have a derivation and we wish to add another string, then this string must be a conclusion of some canon and the premises to that canon must occur previously in the derivation.

### III. 11 Example of a Derivation

One member of the set of strings named derivation is composed of the following string of limited FORTRAN and its production.

\[
\begin{align*}
6I &= I + 1# \\
114I &= I + 12# \\
124GOTO1# & \text{ program}
\end{align*}
\]

This section will demonstrate how statutes (III-24) and (III-25) are used to recursively generate the above string and its production.

Lines 1 - 18 below form one string of the set called derivation.
The same recursion method will be used as was applied to the

canonic system of limited FORTRAN to show that the string 222 belongs
to the set "number" (Section III,6) and to show that the triplet
"2," "124GOTO3#" "3," belongs to the set "...statement labels in...
with reference labels..." (Section III,8).

The recursion will be applied to Statute (III-25) over and over again
until the desired string is reached. The first recursion on Statute (III-25)
has: A for f, "A statement labels inA with reference labels A" for c,
for g. The resulting fc is line 1 of the derivation below.

The first column of the derivation contains the number of that line
for reference purposes. The second column contains the number of
the canon or edict changed to a canon of the canonic system of limited
FORTRAN used (Section III.9). The variable gc is used to denote this
canon. The third column contains the numbers of the lines of the
derivation that are used as premises to the canon. The variable g is
used to denote these premises. The fourth column contains the result of
using the Statute 25.

Line 1 and 2 are generated by letting: f be line 1, c be edict 1
changed to a canon (1 digit), g be .

Lines 1 through 3 are generated by letting: f be lines 1 and 2, gc be
canon 2 (1 digit |1 number), g be line 2 (1 digit), c be 1 number.

Repeating this process over and over the entire string in the right
hand column of the derivation is generated. This string is a member of
the set "derivation" as is defined by the second canonic system.

<table>
<thead>
<tr>
<th>line number</th>
<th>canon gc</th>
<th>line number</th>
<th>Conclusions of Statute (III-25) fc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td></td>
<td>statement labels in ( \Lambda ) with reference labels ( \Lambda )</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td>1 digit</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1 number</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>1, 3</td>
<td>statement labels in ( \delta l=I+1 # ) with reference labels ( \Lambda )</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>2</td>
<td>1 not in ( \Lambda )</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
<td>2 digit</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>6, 3</td>
<td>12 number</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
<td>4, 7, 5</td>
<td>statement labels in ( \delta l=I+1 # | 14 l=I+12 # ) with reference labels ( \Lambda )</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td></td>
<td>1 differ 2</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>9</td>
<td>2 differ 1</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>6</td>
<td>2 not in ( \Lambda )</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>10, 11</td>
<td>2 not in 1</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>8, 12, 2</td>
<td>statement labels in ( \delta l=I+1 # | 14 l=I+12 # | 24 GOTO 1 # ) with reference labels 1,</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td></td>
<td>list</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>2, 14</td>
<td>list</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>15</td>
<td>in 2</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>14, 15, 16, 2</td>
<td>1, in 1, 2,</td>
</tr>
<tr>
<td>18</td>
<td>17</td>
<td>13, 17</td>
<td>( \delta l=I+1 # | 14 l=I+12 # | 24 GOTO 1 # ) program</td>
</tr>
</tbody>
</table>
III. 12 Syntax of GPSS III

The remaining sections of this chapter are concerned with the syntax of a computer simulation language, GPSS III (7090 version). Examples of a use of the specification to check legal formations of strings of GPSS III is given in Section III. 13. The motivation and developments of a complete syntactical specification of GPSS III are given in Section III. 14. Appendix A presents the specification in a form that facilitates its use as a syntax reference. Some of the syntactical constructions that were found to be inaccurately or ambiguously stated in the GPSS III manuals are discussed throughout the remaining sections of this chapter and in Appendix B. Experimental verification of these constructions is also presented in Appendix B.

The remaining sections are written for the reader who has had some contact with GPSS. The reader who has had no contact with GPSS may either read the remaining sections of this chapter for their main ideas and not get involved with the details of a particular construction or example; or he may read Chapter IV, study Appendix B and D, then come back and read these sections in detail.

The syntax of GPSS has been found in general to be less restrictive than is stated in the manuals. For example on page 19 and 224 of [9] the following statement appears: "A block symbol (label) must consist of from three to five alphanumeric, non blank characters, the first three of which must be letters. An additional restriction is that none
of the characters in the following set may be used in a block symbol
(label) + - * / ( & = ! , ) ."

It has been found by experiment that in certain cases the form of
labels may be far more general than this. E.g.,"A" "A12345" "A$., +"
"ABC=, (" "A LL" were accepted, compiled, and executed as statement
labels. Acceptable statement labels were found to be from one to six
(not three to five) characters, the first character must be a letter, the
remaining characters may be void, blank, or any IBM keypunch character
including + -*/(&=! , ). On page 225 of [9] is further stated:"Dollar
signs may never occur in the location field." Dollar signs were found
to be legal characters in the location field of statements and legal in
labels.

The most restrictive form of labels was found to be that for those used
arguments of statements (reference labels) and as arguments of
VARIABLE definition statements. "ABC*1" "ABC*$" "ABC. )" "ABC" were found to be legal reference labels. "ABC$1" "ABC=" "ABC$,
"ABC)." "ABC=1" were found to be legal labels for arguments of
VARIABLE definition statements.

A dollar sign before a label is recognized by GPSS as the label itself,
e.g. $ABC is the label ABC. By placing a dollar sign before a reference
label or a variable definition argument label, the following were found to
be legal reference labels: $A12345" "$A=BC" "$ABC(*)" "$A" and as
legal variable arguments "$ABC$$" "$A,, , ," "$AB".
Reference labels were found to be legal arguments in almost all statements.

The following were found to be illegal statement labels: "$A=BC" "ABCDEFG" "123A". The following were found to be illegal reference labels: "ABCDEF" "A+12" "A=BC1" "ABC'(*)". The following were found to be illegal reference labels with a $ in front: "$A+12" "$A,,," "$A+++" "$A LL".

After all these legal and illegal constructions were examined it was still found that the most restrictive form of a label (a form that may be used anywhere) is more general than is stated in the GPSS manuals, E.g. "ABC$=" "ABC')" "ABC.$" are labels that may be used anywhere.

In the specification of GPSS presented in this dissertation the most general permissible constructions are defined. In the case of labels, because of the diversity of legal forms depending on their context, (e.g., statement labels, reference labels) the definition becomes lengthy but precise.

In general the manuals are not clear as to when an argument of a statement may be void. E.g., is TEST E,,ALL or GATE SF,,ALL legal? In the TEST statement the first two arguments are void. In the GATE statement the first argument is void.

On page 224 [9] the following statement appears: "When a logical attribute or relational mnemonic is needed in the auxiliary TEST, GATE, and LOGIC blocks, it must begin in column 13 for GATE and TEST blocks,
and in column 14 for LOGIC blocks. It was found by experiment that the logical attribute or relational mnemonic may begin anywhere from column 13 to column 18 in these blocks.

In general the manuals are more restrictive than the GPSS actually is; however, there are instances when the manuals are not restrictive enough. E.g., on page 76 [10] appears the statement: "In addition, any card with a blank operation field will also be treated as a comments card. Thus, blank cards may be used to provide spaces on the listing if desired" is not correct." Blank cards are illegal and cause the entire program to be rejected.

On page 46 of [9] the statement: "Any nonblank character (except *) in column 1 will suppress printing of the block count" is not correct. Any character other than a letter or * or, in certain instances, a $ will cause an error which causes the entire program to be rejected. If a letter is used in column 1 it still does not suppress the printing of the block count statistics. This incorrect statement gives both semantically and syntactically incorrect information; semantically because a character in column 1 does not mean a suppressing of the block count statistics; syntactically because a card with any character except a letter or * or $ in column 1 will be rejected solely on its form.

For further examples of the form of legal and illegal statements consult Appendix B.

GPSS is syntactically very complicated. This is a result of the

* see Appendix B
num erous and different restrictions placed on all strings of GPSS. For
this reason the specification becomes much longer than it would if there
were fewer restrictions, e.g., if there were only one legal form
of labels as there is in FORTRAN, MAD or ALGOL. *

III.13 Examples of a Use of the Specification of GPSS

It is suggested that the reader who is not thoroughly familiar with
with GPSS read the second example, page 46 (which is not as involved)
and not read the one below.

Let us suppose a programmer wishes to know the legal forms of a
TEST statement in a program of GPSS. He searches the left column
of the canonic system, Appendix A, until he finds the predicate program.
All predicates are listed alphabetically in Appendix A. To the right he
finds the canon

s statement labels in p with reference labels r; r labels among statement
labels s + p program.

There are two premises that must be satisfied. This canon defines
a program in the same way as Canon III-17 of limited FORTRAN. Each
predicate used in a premise is defined in the specification and the
programmer may search the left hand column of the specification until
he finds these predicates. The names of the predicates have been
chosen to assist the programmer in remembering the construction
defined by the predicate.

The second premise used in the canon above is

r labels among statement labels s

This premise assures that all reference labels used in a program must
appear as statement labels. If the programmer looks at the canons and

* If a specification of a language, similar to the one presented in this
dissertation, preceded its implementation on a computer, the difficult
constructions could be eliminated. This was not done with GPSS III.
edicts that define the set of strings named labels among statement labels, he will find that r and s are sequences of statement labels separated by commas where all labels of r are among s. E.g., The pair of strings "ABC, A$=$" "ABCD, A$=$, X+()YZ, , , , , ABC" is a member of the set named labels among statement labels.

The programmer searches the left hand column for the predicate used in the first premise...statement labels in...with reference labels... Among the canons that define this predicate he will find the conclusion *  
+s1, statement labels in paTEST# with reference labels rz, g, i,  
He must now find the permissible values of the variables s1 p a t y f h r z g i by finding the relevant premises (Section III. 7, page 27). That is, he must find the premises that contain these variables. Then he must find the definitions of the predicates used in these premises. When this is done he will find: s is a list of statement labels, 1 is a label, p is a string of statements, a is a field that is six characters long followed by a space, a consists of 1 and the remaining characters are spaces, t is a test mnemonic field which is six characters long, t consists of a test mnemonic (L, LE, G, GE, E, NE) and blank spaces, y is an argument that may be void or any standard numerical attribute or any standard numerical attribute using indirect addressing or any reference label, f and h may be void or a comma or a comma followed by any standard numerical attribute or any indirect standard numerical attribute or any reference label. $A$* and ABC+1 are examples of reference labels. r is a list.

*The sign 1 is used to denote the letter l as in label. 1 denotes the number on.
of bases of reference labels. E.g., The reference label $A^*+$ has as its base $A^+$. This is the base that must appear as a statement label in the same program. $z$, $g$, $r$ are either void or if $y$, $f$ or $h$ were reference labels $z$, $g$ or $i$ are their respective base. E.g., If $y$ were $ABC+1$ then $z$ would be $ABC$.

Thus some examples of legal TEST statements are:

- $A^*+$ TEST E $A^*+3$, P1, $A^*+4$
- ABC TEST E P1, ABC
- Z12345 TEST E ,,ABC
- A TEST E
- ALL TEST E 1,$A-2,$
- A,, TEST E,,ABC

None of the above statements are legal forms according to the GPSS manual. Note the generality of statement labels, the generality of arguments, and the generality of the location of the test mnemonic $E$. Also note that when a reference label is used as an argument, its base must appear as a statement label. E.g., the argument $A-2$ is a reference label with a base $A$ and it is $A$ that must appear in the program.

Another example of the use of the specification by a programmer might be as follows. Suppose a programmer wishes to know the form of permissible comments or comment statements in GPSS. He searches the left hand column of the canonic system, Appendix A, until he finds the predicate comment program. To the right is the edict
a#b program; s spaces; f string \( \leftrightarrow \) a#f#b, a#*f#b, *f#b comment program

The edict above has three conclusions. The last *f#b comment program has two relevant premises (premises containing the same variable as a conclusion) and they are a#b program and f string. The programmer looks in the specification and finds that a string is simply a sequence of symbols of an IBM card punch. If a#p is a program b may be a program (which includes the null string). Thus the sign * in column 1 followed by a string followed by a program is a legal comment program.

The second conclusion indicates inserting a card "*f#" into a program is a legal construction.

The first conclusion indicates that one or more spaces after any statement followed by any string f is a legal construction. All statements consist of at least 18 spaces or characters; thus no comment may begin before column 20.

III. 14 Development and Motivation of a Complete Syntactical Specification of GPSS III

This section is concerned with the development of a complete syntactical specification of GPSS III (7090 version). The specification is a canonic system and follows the same interpretation and construction rules as were discussed in Section III. 1 through III. 11. The canonic system of GPSS is developed in subsections a through o. Each subsection consists of a number of canons or edicts followed by a discussion of the motivation for those canons or edicts. The reader may read each
canon or edict defining a set of strings of GPSS. Then he may read
the motivation for these edicts. The sections build upon themselves.
We first talk about the basis of GPSS, letters, symbols, digits, then
labels, next arguments. With these definitions we may then define
statements and programs.

a. Symbols, Spaces, Numbers, Letters

symbol | 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L
        M N O P Q R S T U V W X Y Z = \$ *

symbol or space x symbol x symbol or space

symbol or space or void x symbol or space x symbol or space

string x symbol or space x string

digit 0 1 2 3 4 5 6 7 8 9 digit

number d digit; n number d nd number

number or void x number x number or void


The predicates used were chosen to aid the reader in remembering
the set of strings named by the predicate. The set of strings symbol
or space have as members symbols and spaces and the null string.
When this predicate is used with a variable (x symbol or space or void)
x may be either a symbol or a space or a void. Abbreviations have
been used to shorten the length of predicates, sym for symbol. However, wherever confusion might result abbreviations are not used.

b. **Labels**

**label symbol**

- d digit; l letter $\downarrow \lambda \varnothing \downarrow \{ \cdot \downarrow \}$ label symbol  
  
  [note, (\+- are not]

**var label sym**

- x label symbol $\downarrow x_1$ var label sym  
  [note (+-*/ are not]

**ref label sym**

- x label symbol $\downarrow x_1 (\cdot /$ ref label sym  
  [note, + - are not]

**label**

- a, b, c letter; d, e label symbol $\uparrow$ abcde label

**fnc label**

- x label; a, b, c letter; d, e f sym or space or void  
  $\downarrow x_1$ abcdef fnc label

**var label**

- x label; a, b, c letter; d, e var label sym $\downarrow x_1$ abcde  
  var label

**stm label**

- x label; a letter; bcde sym or space or void; f symbol  
  $\downarrow x_1$ a, abcde, t stm labels

**plain ref label base**

- x label; a, b, c letter; d, e ref label sym; n number  
  $\downarrow x_1$ plain ref label base x; abcde plain ref label base abcde; abcde+n plain ref label base abcde

**indirect ref label base**

- x label; a letter; b, c, d, e f ref label sym; n number  
  $\downarrow$ $x1$ indirect ref label base x; $xabcde$ indirect ref label base abcdef; $xabcde+n$ indirect ref label base abcdef; $xabcde-n$ indirect ref label base abcdef
The predicates label symbol and label are used to define the form of a label that may be used anywhere.* If the programmer wishes more general forms of labels for a particular context he may further consult the other definitions such as function labels (fnc labels) or variable labels, which may be used as arguments of function data cards or variable statements. The first conclusion of each canon defining the various labels is read: "x is a ... label where x is a label"; thus, the various labels are at least the label that may be used anywhere. It is in part the various forms for labels which have made the specification of GPSS so lengthy.

There are two types of reference labels, a plain (e.g. ABC ABC+1 ABC-1) and an indirect (e.g. $ABC $ABC+1 $ABC-1). A $ before a label in GPSS denotes the label itself, e.g., $ABC denotes ABC. When using either type as an argument of a statement its base must appear as a statement label in the same program. Hence, it is necessary to use predicates of degree two to specify the reference labels; one degree refers to the reference label (e.g., $ABC01) the other to its base (e.g. ABC).

*label names a set that is the intersection of all the various sets of labels.
The predicate `ref label base` names the set of ordered pairs that is the union of the sets named `plain ref label base` and `indirect ref label base`.

c. **Standard Numerical Attributes**

```

stand num attrib \[ p pref stand num attrib; n number \[ n \] pn, RN1, CL1 \] stand num attrib sna

stand num attrib \[ p pref stand num attrib; n number \[ n \] *n \] pn, p*n, sna*sna

\[ RN1, CL1 \] stand num attrib sna* sna
```

Notice the difference between `stand num attrib sna` and `stand num attrib sna* sna`. The first defines a standard numerical attribute without indirect addressing and the second with indirect addressing. This distinction is necessary because indirect addressing in standard numerical attributes is not always permissible.

d. **Arguments**

```
arg sna* sna or ref label \[ a stand num attrib sna*sna; b ref label base c \] \[ a \] arg sna* sna or ref label x; b ref label base \[ \] \[ a \] arg sna* sna or ref label x, sna or ref label y
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
| arg num or ref label | \( n \) number; a plain ref label base b; c indirect ref label base d; \( n \) arg num or ref label \( A \); a arg num or ref label b; c arg num or ref label d |
| arg void num or ref label | a arg num or ref label b \( \vdash \) a arg void num or ref label b; \( A \) arg void num or ref label \( A \) |
| arg void , num or ref label | a arg void num or ref label b \( \vdash \) , a arg void , num or ref label b; \( A \) arg void , num or ref label \( A \) |
| arg sna or ref label | a stand num attrib sna; x ref label base y \( \vdash \) a arg sna or ref label \( A \); x arg sna or ref label y |
| arg void sna or ref label | a arg sna or ref label b \( \vdash \) a arg void sna or ref label b; \( A \) arg void sna or ref label \( A \) |
| arg sna* sna or nonrelative ref label | \( x \) stand num attrib san* sna; \( a, b, c \) letter; \( d, e, f, g, h \) ref label sym \( \vdash \) \( x \) arg sna* sna or nonrelative ref label \( A \); abcde arg sna* sna or nonrelative ref label \( A \); abcde; \( $\text{defgh} \) arg sna* sna or nonrelative ref label \( \text{defgh} \); \( W\text{defgh} \) arg sna* sna or nonrelative ref label \( \text{defgh} \); \( N\text{defgh} \) arg sna* sna or nonrelative ref label \( \text{defgh} \) |
| print mnemonic arg | \( \vdash \), MOV, FUT, CHA, I, N, W, F, S, O, T, X, \( A \) |
| print mnemonic arg | |
| transfer mnemonic arg | n number \( \vdash \) .n, FN, Pn, BOTH, PICK, SBR, SIM, ALL, \( A \) |
| transfer mnemonic arg | |
| ibsys tape arg | \( \vdash \) SYSOU2, SYSLB2, SYSP2 ibsys tape arg |
These canons and edicts define most of the permissible arguments of statements of GPSS. The predicates have been chosen to give information as to the form of the constructions named by the predicate. For example, members of the set named \texttt{arg void, sna* sna} or \texttt{ref label}, are the null string, a comma, all standard numerical attributes, plain or indirect addressing preceded by a comma, all reference labels preceded by a comma and their bases. In all cases where predicates of degree two are used to define arguments the second variable refers to the base of the reference label used in the argument. If a reference label is not used, the second variable denotes the null string.

Note the large number of different forms of arguments. This is due to the awkward restrictions placed on the various arguments depending on their context. For example the form of arguments allowable in a \texttt{GENERATE} statement are quite different from those of a \texttt{test} statement or a \texttt{SPLIT} statement or an \texttt{UNLINK} statement. Each one of these statements has different restrictions placed on its arguments.

\textbf{e. Fields}

\texttt{x space \mid l_1 x l_1 spaces}
54 spaces or void | x space \( \rightarrow x, \Lambda \) spaces or void 

six sym or spaces \( \rightarrow a, b, c, d, e, f \) sym or spaces | abcdef six sym or spaces

logic mnemonic \( \rightarrow R, I, S \) logic mnemonic 

logic mnemonic field \( \rightarrow abc \) six sym or spaces; b logic mnemonic; a, c spaces 
or void | abc logic mnemonic field

gate mnemonic \( \rightarrow M, N, U, I, N, S, S, N, S, E, S, N, E, L, R, L \) 
gate mnemonic 

gate mnemonic field \( \rightarrow abc \) six gym or spaces; b gate mnemonic; a, c spaces 
or void | abc gate mnemonic field

test mnemonic \( \rightarrow L, E, G, G, E, E, N, E \) test mnemonic 

test mnemonic field \( \rightarrow abc \) six sym or spaces; b test mnemonic; a, c spaces 
or void | abc test mnemonic field

loc field with sym label \( \rightarrow abc \) six sym or spaces; b stm label; a, c spaces void 

\( \rightarrow abc \) loc field with stm label b 

It was found by experiment, executing test programs, that the logic mnemonics used in LOGIC, GATE, or TEST statements may appear anywhere in columns 13 through 18. Hence the motivation of the predicates logic mnemonic field, gate mnemonic field, test mnemonic field. These predicates define sets of strings whose members are six characters long. All the characters are spaces except for the logic, gate, or test mnemonic used in this field.

Likewise there is a location field in which a statement label may appear. The field is six characters long. The statement label may be up
to six symbols or spaces long. The spaces before the first symbol
and after the last symbol are not considered part of the label. Thus the
field $A_1B_1$ contains a statement label $A_1B$. Statement labels must be kept
track of for possible use as reference labels. Thus a predicate of degree
two, a loc field with stm labels $b$. $a$ is the location field (e.g. $A_1B_1$)
and $b$ is the statement label in that field (e.g. $A_1B$).

f. Statements

... stm labels in... with ref labels

| - A stm labels in $A$ with ref labels $A$

first premises s stm labels in $p$ with ref labels $r$; a loc field with

label $q$; ---

| - s$^1$, stm labels in paBUFFER$^5#$ with ref labels $r$
| - s$^1$, stm labels in paTRACE$^6#$ with ref labels $r$
| - s$^1$, stm labels in paUNTRACE$^4#$ with ref labels $r$

second premises b arg sna* sna or ref label $c$; n number; y arg

void sna* sna or ref label $z$; ---

| - s$^1$, stm labels in paASSEMBLE$^3$b# with ref

labels $r_c$, $r_c$
| - s$^1$, stm labels in paEXECUTE$^4$b# with ref

labels $r_c$, $r_c$
| - s$^1$, stm labels in paGATHER$^5$b# with ref labels

$rc$, $rc$.
t logic mnemonic field

\[
\begin{align*}
& s_1, \text{stm labels in paLOGICtb# with ref labels } rc, \\
& s_1, \text{stm labels in paMARK7b# with ref labels } rc, \\
& s_1, \text{stm labels in paMATCH6b# with ref labels } rc, \\
& s_1, \text{stm labels in paPREEMPT4b# with ref labels } rc, \\
& s_1, \text{stm labels in paPRIORITY3y# with ref labels rz}, \\
& s_1, \text{stm labels in paPRIORITY3y, BUFFER# with ref labels rz}, \\
& s_1, \text{stm labels in paRELEASE4b# with ref labels rz}, \\
& s_1, \text{stm labels in paRETURN5b# with ref labels rz}, \\
& s_1, \text{stm labels in paSEIZE6b# with ref labels rz}, \\
& s_1, \text{stm labels in paTERMINATE2y# with ref labels rz}, \\
\end{align*}
\]

third premises

\[
\begin{align*}
& \text{d arg sna* sna or ref label e; f arg void, sna* sna or ref label g;} \\
& s_1, \text{stm labels in paCHANGE5b, d# with ref labels } rc, e, \\
& s_1, \text{stm labels in paDEPART5bf# with ref labels } rc, g, \\
& s_1, \text{stm labels in paENTER6bf# with ref labels } rc, \\
& \text{t gate mnemonic field } s_1, \text{stm labels in paGATE1tbf# with ref labels } rc, \\
& s_1, \text{stm labels in paLEAVE6bf# with ref labels } rc,
\end{align*}
\]
t link order arg \{ s\}, stm labels in paLINK?b\# with ref labels rc, g,
\{ s\}, stm labels in paLOOP?b, d\# with ref labels rc, e,
\{ s\}, stm labels in paQUEUE?b\# with ref labels rc, g,
\{ s\}, stm labels in paTABULATE?b\# with ref labels rc, g.

fourth premises h arg void, sna* sna or ref label i; j arg num or ref label k; m arg void number ref label o; q arg void, num or ref label u; v arg void num or ref label w;

\{ s\}, stm labels in paADVANCE\#yq\# with ref labels rz, u,
\{ s\}, stm labels in paADVANCE\#y, *n\# with ref labels rz,
\{ s\}, stm labels in paADVANCE\#y, F\#n\# with ref labels rz,
\{ s\}, stm labels in paADVANCE\#y, F\#n\# with ref labels rz,
\{ s\}, stm labels in paINDEX\#jq\# with ref labels rk, u,
\{ s\}, stm labels in paINITIAL\#xnq\# with ref labels ru,
t number void \{ s\}, stm labels in paINITIAL\#xn-t\# with ref labels r

\{ s\}, stm labels in paPRINT\#v, mt\# with ref labels rw,
\{ s\}, stm labels in paSPLIT\#y, d\# with ref labels rz, e
t test mnemonic field | s1, stm labels in paTEST\textsubscript{5y}, d, mh\# with ref
labels rz, e, o, i,

\text{t transfer mnemonic arg} | s1, stm labels in paTRANSFER\textsubscript{3fh}, d with ref
labels rz, g, i,

\text{t xact unlink limit arg sn} or ref label f\textsubscript{1}; f\textsubscript{2} \text{xact unlinked arg sn} or
ref label | s1, stm labels in paUNLINK\textsubscript{5b}, d, tf\textsubscript{2fh}\# with
ref labels rc, e, f\textsubscript{1}, f\textsubscript{3}, g, i,

\text{fifth premises} a\textsubscript{1} \text{arg sn}^* \text{sn} or ref label \text{no offset} a\textsubscript{2};

| s1, stm labels in paASSIGN\textsubscript{5a}\textsubscript{1} fh\# with ref
labels ra\textsubscript{2}, g, i,

| s1, stm labels in paASSIGN\textsubscript{5a}\textsubscript{1} yt\# with ref
labels ra\textsubscript{2}, z, i,

| s1, stm labels in paASSIGN\textsubscript{5a}\textsubscript{1} yh\# with ref
labels ra\textsubscript{2}, z, i,

| s1, stm labels in paSAVEVALUE\textsubscript{3a}\textsubscript{1} f\# with
ref labels ra\textsubscript{2}, g,

| s1, stm labels in paSAVEVALUE\textsubscript{3a}\textsubscript{1} yt\# with
ref labels ra\textsubscript{2}, z,

| s1, stm labels in paSAVEVALUE\textsubscript{3a}\textsubscript{1} yh\# with
ref labels ra\textsubscript{2}, z,
This method of defining statements is the same method as was used in the canonic system of limited FORTRAN to define statements. The set named by "s stm labels in p with ref labels r" is a triple \((s, p, r)\): \(s\) is a sequence of all statement labels, separated by commas, in \(p\); \(p\) is a sequence of statements, separated by \#; \(r\) is a sequence of reference labels, separated by commas, in \(p\).

The first canon of this section is necessary to start the recursion correctly. Using this first canon \(s, p, r\) may denote \(\Lambda\), the null string. After a program is built up a check will be made to see if all the reference
labels in the sequence $r$ are among the statement labels in the sequence $s$. Because of this check it was necessary to use predicates of degree two naming an ordered pair of strings $(bc)$ for all arguments; $b$ may denote a reference label if any were used, $c$ would denote the base of this reference label. Thus in the instruction `ASSEMBLE$_3^{\equiv}ABC+1$` defined above $b$ denotes $ABC+1$ and $c$ denotes $ABC$. It is the base $c$ that is added to the list of reference labels, because it is $c$ that must occur as a statement label in the same program.

Note in the canons above defining `BUFFER`, `TRACE` and `UNTRACE` statements, the statements are all followed by spaces. The spaces are necessary to ensure that comments (to be specified in Section III.14 j) do not occur before column 20 on a card.

Note the variety of arguments permissible in the different statements. By experimentation it has been found when void arguments are legal and such arguments are accordingly defined. It has also been found that more flexibility in the permissible forms of arguments is legal than the manuals imply, especially in the use of reference labels. E.g. `A$_3$.BC$_3^{\equiv}$TERMINATE$_3^{\equiv},A$_3$,BC+1` is a permissible statement.

Note that a statement label is permissible in the location field of an `INITIAL` statement in a program. However, the statement label does not become a member of the list of statement labels associated with that program. Thus, if a statement label is used with an `INITIAL` statement it may never be referenced.
Consult Appendix B for some constructions that are defined by the above canons but are not permissible according to the manuals. Appendix C may be studied for the more conventional constructions used in GPSS.

g. **Control Cards**

<table>
<thead>
<tr>
<th>alpha numeric</th>
<th>x letter; y digit; z alpha numeric</th>
<th>( \Lambda ), ( \gamma ), ( \zeta ) alpha numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>first premises</td>
<td>a field with stm label b; s stm labels in p with ref labels r; ---</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in paRESET6# with ref labels r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in pa$6JOB38# with ref labels r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in paEND8# with ref labels r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in paLIST7# with ref labels r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in paUNLIST5# with ref labels r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in paCLEAR6# with ref labels r</td>
<td></td>
</tr>
<tr>
<td>second premises</td>
<td>c arg num or ref label d; x alpha numeric; ---</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in paREAD7c# with ref labels rd,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in paREWIND5t# with ref labels r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) s stm labels in paSAVE7x# with ref labels r</td>
<td></td>
</tr>
<tr>
<td>third premises</td>
<td>y arg void num or ref label z; e arg void , num or ref label f; g arg void , num or ref label h; i arg void , num or ref label j; ---</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \vdash ) t ibsys tape ( \vdash ) s stm labels in paJOBTAPE4t, cdg# with ref labels rd, f, h,</td>
<td></td>
</tr>
</tbody>
</table>
The above canons specify the control cards of GPSS. Note that a statement label is permissible in the location field of every control card but this label is never added to the list of statement labels "s". Thus a statement label of a control card may never be referenced.

h. Program

\[
\begin{align*}
\text{list stm labels} & \quad a \text{ list stm label} ; b \text{ list stm label} \vdash A \text{ labels among stm labels} \\
\text{labels among} & \quad a , b \text{ list stm labels} ; c \text{ stm label} \vdash A \text{ labels among stm labels a} ; A \text{ labels among stm labels a} ; c , \text{ labels among stm labels ac} , b \text{ labels among stm labels b} ; c \text{ labels among stm labels} \vdash A \text{ labels among stm labels} \\
\text{program} & \quad a \text{ stm labels in p with ref labels} r ; r \text{ labels among stm labels} a \vdash p \text{ program} \end{align*}
\]

Section III. 14 f and III. 14 g defined sets of triples \((s, p, r)\): Where \(s\) is a sequence of statement labels separated by commas in \(p\); \(p\) is a sequence of statements; \(r\) is a sequence of reference labels separated by commas used in \(p\). In order for \(p\) to be a program all reference labels must correspond to statement labels in the same program. Thus members
of the list \( r \) of reference labels must be among the members of the list \( s \) of statement labels. The above canons specify this restriction. Note in the canon and edict defining labels among stm labels, the variables \( a, b, c, d, e \) or \( f \) may denote null strings. Thus it is possible to generate the conclusion \( AB, , , C \) labels among stm labels \( ZZ, AB, , , D, C, , EFG \). This form of conclusion was convenient because both \( s \) and \( r \) of the triples \((s, p, r)\) may have series of commas resulting from either, the location fields of statements containing null string as labels, or an argument that did not contain a reference label. For example a TEST statement with no statement label and no reference labels may have the conclusion from Section III.14 if: \( 1 \) denotes \( A \); \( y \) denotes \( P1 \), \( f \) denotes \( Q1 \), and \( h \) denotes \( 5 \); thus \( z, g, i \) would denote \( A \); and \( a \) would denote seven spaces. The resulting conclusion would be

\[
\vdash s, \text{stm labels in p} \text{TEST} \text{t} P1, Q1, 5 \text{ with ref labels } r, , ,
\]

i. Pseudo Operations

- **first premises** a field with stm label \( l \); b ref label; b labels among stm labels \( sb, \); s stm labels in \( p \) with ref labels \( r \); s spaces void; bs, ns, ts six sym or spaces;

  \[
  \vdash s \text{ stm labels in paORG8bs# with ref labels } r
  \]

- **n number or void** \( s \) stm labels in paICT8ns# with ref labels \( r \)

- **t stm label** \( s1, \) stm labels in paSYN8bs# with ref labels \( r \)

  \[
  \vdash s \text{ stm labels in paABS8# with ref labels } r
  \]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The above edicts define the constructions of pseudo-operations of GPSS. It was found by experiment that statement labels are permissible in all pseudo-operations but in only one (SYN) is the statement label added to the list of symbols by the GPSS compiler. Therefore, only the statement label of the pseudo-operation SYN may be used as a reference label. Note that 1 is added to the list of statement labels in the edict defining SYN. All other statement labels of pseudo-operations are not added to the list of statement labels s.

Note if a reference label is used in an ORG or SYN pseudo-operation then that label must be previously defined in the same program. The work previously is used to differentiate from the more usual case when the reference label may correspond to any statement label in the same program (before or after the statement containing the reference label).

j. Comments

Possible constructions of comments specified by the above edicts were discussed in Section III. 13. That discussion will not be repeated here. Note in the above edicts the variable b may denote the null string because # by itself is not a program.
k. Function Definition

function codes

n number | Cn, Dn, Ln, En, Mn function codes

five sym or spaces

a₁ b₁ c₁ d₁ e sym or space → abcd five sym or spaces

field with num

abc five sym or spaces; b number; a₁ c spaces or void

| = lacdl field with number b

fnc. card with ref label

a field with num b; c function code; d arg sn or

ref label e → a FUNCTION3c, d# fnc card with ref

label e

fnc data field sna or ref label

first premises a₁ c spaces void; s stand num attrib sn a;

abc six sym or spaces; b fnc label → abc fnc data field sna

or ref label b

asc six sym or spaces → asc fnc data field sna

or ref label

ae, fc six sym or spaces; e, f number; void → ae, fc fnc data field

or ref label

end premises

fnc data card with ref labels

a fnc data field sna or ref label b; c fnc data field sna

or ref label d; e fnc data field sna or ref label f; g fnc

data field sna or ref label h; i fnc data card with

ref labels j → acdg fnc data card with ref labels

b, d, f, h, ; iac fnc data card with ref labels jb, d,

eighty sym or spaces

a₁ a₂ a₃ ... a₈₀ sym or spaces → a₁ a₂ a₃ ... a₈₀ eighty

sym or spaces
The relatively large number of canons and edicts needed to specify the function definition cards is due to the difficult constructions and restrictions placed on these cards. The predicate field with number name: the set of strings that are the location field of all definition statements. These fields consist of a number and spaces; the first and last character must be a space (e.g. 120 A 251 357 are legal fields with numbers 20, 25, and 7 respectively).

A function data field is six characters in length. Contained in this field may be any standard numerical attribute (no indirect addressing) or any reference label or any decimal number (e.g. 1P13 ABCDEFG 123.41 are function data fields). Note no indirect addressing is permissible no relative addressing is permissible. A function data card is composed of a sequence of function data fields.

No comments are permissible on function data cards and no comment cards may be inserted between function data cards. This is the only card
to which these restrictions apply. The sign % is used to aid in specifying these restrictions and is interpreted as an end of card mark. The above canons and edicts define a function data card to be a sequence of function data fields. An eighty column function data card is a function data card followed by the correct number of spaces to make eight columns followed by a %. This construction and the manner of defining comments (Section III. 14 j) prevents comments from being added to or inserted between function data cards.

A function def is a function card followed by eighty column function data cards.

1. Variable Definition

\[
\begin{align*}
\text{variable op} & \mid +, -, \times, \div, (, ( \text{variable op} \\
\text{var arg sna or ref label} & \mid \text{stand num attrib sna; } \text{var label; } \text{letter; } e, f, g, h \text{ var label sym} \mid \text{a var arg sna or ref label A; } N\text{defgh var arg sna or ref label defgh; } W\text{defgh var arg sna or ref label defgh; } S\text{defgh var arg sna or ref label defgh; } I \text{ var arg sna or ref label I} \\
\text{var arg field} & \mid \text{var arg sna or ref label b; c variable op; d var arg field with ref labels e} \mid \text{a var arg field with ref labels b, } ; \text{dca var arg field with ref labels eb,} \\
\text{var def with ref labels} & \mid \text{a var arg field with ref labels b; c field with num d} \mid \text{cVARIABLE3a# var def with ref labels b;} \\
& \text{A var def with ref labels A}
\end{align*}
\]
A variable argument field is a sequence of variable arguments separated by variable operators. Again, predicates of degree two are used to keep track of the bases of reference labels used, if any. By keeping track of the bases, we may later check to see if all bases of reference labels used in a variable definition occur as statement labels in some program. If they do then this variable definition may be added to the program.

For examples of VARIABLE definition statements consult Appendixes B and D.

m. Storage Definition, Table Definition

<table>
<thead>
<tr>
<th>storage def with ref label</th>
<th>a field with num b; c arg num or ref label d</th>
<th>A storage def with ref label A; aSTORAGE e with ref label d</th>
</tr>
</thead>
<tbody>
<tr>
<td>table arg</td>
<td>a stand num attrib sna</td>
<td>aₜₐₐₐₜIA table arg</td>
</tr>
<tr>
<td>table def with ref labels</td>
<td>a field with num b; c table arg; n number; d arg</td>
<td>void , num or ref label e, f arg void , num or ref label g; h arg void , num or ref label i; j arg void , num or ref label k</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aQTABLE endfhj# table def with ref labels e, g, i, k,</td>
</tr>
</tbody>
</table>

n. Program

| program (three out of 3) | b func def with ref labels c; d var def with ref labels e; f storage def with ref label g; h table def with ref labels i; s stm labels in mn with ref labels r; rc, e, g, i labels among stm labels s | mbdfsfr program |

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
simulate program

\[ p \text{ program } \vdash \#6 \text{SIMULATE}\#p \text{ simulate program} \]

The canon above, defining program, allows function, variable, storage, or table definition cards to be inserted in any program provided that all reference labels used in these definitions have their bases among the statement labels associated with that program.

A GPSS program that has a card with the sign * in column one and the word SIMULATE starting in column 8 will, if the program is syntactically correct, compile and execute.

o. GPSS Program with Card Input

any length card program

\[ a\#b \text{ program; s spaces or void } \vdash \text{as}\#b \text{ any length card program} \]

seventy-one column cards

\[ a_1,a_2,a_3,\ldots,a_{71} \text{ sym or space; x seventy-one column cards } \vdash \Lambda x a_1 a_2 a_3 \ldots a_{71}\# \text{ seventy-one column card} \]

legal info card program

\[ p \text{ any length card program; p seventy-one column cards } \vdash p \text{ legal info card program} \]

eighteen sym or spaces

\[ a_1,a_2,a_3,\ldots,a_{18} \text{ sym or space } \vdash a_1 a_2 a_3 \ldots a_{18} \text{ eighteen sym or spaces} \]

symbols

\[ x \text{ symbol; y symbols } \vdash x,y \text{ symbols} \]

identify card program

\[ x\text{absd#y, wae#y legal info card program; a}_{1},a_{2},a_{3},\ldots,a_{9} \text{ sym space; a eighteen sym spaces; b, d string; e symbols; s space } \vdash x\text{absda}_{1}a_{2} \ldots a_{9}\#y_{w} a_{2} \ldots a_{9}\#y \text{ identify card program} \]

cards

\[ a_1 a_2 \ldots a_{80} \text{ sym or space; b cards } \vdash a_1 b a_1 a_2 \ldots a_{80}\# \text{ cards} \]
This section is concerned with filling in the proper number of spaces after the end of a statement, or after a statement and its comment, to complete a card of 80 columns. No attempt has been made in previous canons to assure that the end of card mark # was placed after 80 characters. We have assured that all statements begin in column 8 and all arguments begin in column 19. The first canon of this section allows the insertion of spaces between the character before the end of card mark # and the end of card mark. The resulting set of strings are programs that consist of a sequence of any length strings separated by #.

Out of all the members of the set named any length card program there is a subset that has as members sequences of strings (of exactly 71 characters) separated by # that are programs. We want the programs composed of strings of exactly 71 characters and not 80 characters, because the characters in column 72 through 80 are never recognized by GPSS. Let us take a case where an illegal construction would result if we included all 80 columns. Take the statement

```
ABC TRANSFER 222222222222222222222222222222222222222222222222
```

```
222; ABC; ABC+12#
```

This statement could be derived from the canon defining a TRANSFER statement. However, it does not form a legal card program even though
it is 80 characters long. It is not legal for two reasons. First, an argument of GPSS must terminate before column 72. If any one of the 2's had been a space this would have been a legal construction. Everything to the right of the space would have been taken as a comment.

Second, GPSS does not recognize column 72 through 80. Thus the label in the second argument ABC would be interpreted as A which would be an illegal reference label.

To define the subset of any length card programs which are 71 columns long we take the intersection of the set any length card program and the set 71 column cards. This intersection is called legal info card program.

If a card of a legal info card program has a continuous argument up to column 71 a space must occur in column 72 followed by any identifier to complete the 80 columns of a card. If at least one space occurs in columns 19 through 71 then column 72 through 80 may contain any character. The canon defining identify card program takes into account the two cases (continuous argument or not).

An identifier card program consists of legal programs that are composed of cards of from 71 columns to 80 columns long. There is a subset of the set identifier card program that is composed of programs containing cards of exactly 80 columns. To define this subset we take the intersection of the set identifier card program and the set cards. The resulting set is called legal card program.

The members of the set legal card program are all the syntactically
legal constructions of card programs of GPSS III (7090 version). If a GPSS III simulation run is desired, it is necessary to place a card with the sign * in column one and the word SIMULATE starting in column 8 as the first card of a program.
CHAPTER IV

TECHNIQUES FOR MODELING COMPLEX SYSTEMS

IV.1 Introduction

One view of the process involved in computer applications is depicted in Figure IV-1.

![Diagram of the process involving problem abstraction and computer implementation](image)

Figure IV-1 A view of digital simulation.

This dissertation is concerned with some of the issues depicted in the above figure when this process is applied to the digital simulation of complex systems. Chapters II and III were concerned with the syntax of computer languages used in this process.

The purpose of this chapter is to present a straightforward method for characterizing and simulating complex systems.

The method presented allows the user to a) model the elements and characteristics of a system by sets of n-tuples and b) specify the operations in the system by a sequence of translations on the n-tuples.
Before discussing our views on simulation, we will distinguish
the functional from the structural aspects of a model or a theory.
This distinction has been aptly made by Prof. A. Oettinger: [76]

"A functional model is like the electrical engineer's proverbial
"black box," where something goes in and something comes out, and
what is inside is unknown or relevant only to the extent of somehow
relating outputs to inputs. A structural model emphasizes the
contents of the box. A curve describing the current passing
through a semiconductor device as a function of the voltage applied
across its terminals is a functional model of this device that is
exceedingly useful to electron-circuit designers. Most often such
curves are obtained by fitting a smooth line to actual currents and
voltages measured for a number of devices. A corresponding structural
model would account for the characteristic slope of the curve in
terms that describe the transport of charge-carriers through semiconductors,
the geometry of the contacts and so forth. A good structural model
typically has greater predictive power than a functional one. In
this case it would predict changes in the voltage-current characteristics
when the geometry of the interface or the impurities in the semiconductor
are varied."

This chapter is concerned with the structural simulation of
discrete state-changing systems, where the changes in the state of
the system occur at discrete intervals. The systems are modeled
as network flow systems with the following characteristics:
the system contains "components" (or "elements") each of which performs a prescribed function;

items flow through the system, from one component to another, requiring the performance of the present component function before the item can move on to the next component;

components have a finite capacity and a non-zero process time; hence items may have to wait in "waiting lines" or "queues" before being processed.

To demonstrate this model we will use portions of a digital simulation of the clinical chemistry laboratories of Yale-New Haven Hospital. We use the structural model to analyze and predict system behavior. In particular, we wish to use the model to evaluate the effect of varying equipment parameters, hospital procedures, introduction of a time-shared digital computer system, software parameters, technician parameters. The principal measures of system performance produced by the model are the process time of biological samples, equipment utilization, job queues, and response time to an emergency request.

IV.1 Modeling Technique

a. Underlying Systems

Complex systems may be viewed as comprising two underlying systems, an information system and a materials handling system. The materials handling system deals with physical quantities. The
information system deals with information. Their interrelationship is depicted in Figure IV-2. We will use the term "complex system" or "system" to refer to the entire system, or the materials handling system, or the information system. The underlying systems of a complex system may themselves be complex systems.

![Diagram of underlying systems of a complex system]

Figure IV-2 Underlying systems of a complex system.

b. Abstracted Model

**Set of n-tuples**

Each element of a complex system may be represented by an n-tuple of scalars $x_1, x_2, \ldots, x_n$. The set of all n-tuples will be written $S_o$, where the subscript "o" denotes that $S_o$ is the set of n-tuples associated with the object system. (See Section I.2 for definition of object system.)

** Scalars**

The elements of the system have individual characteristics. Correspondingly, each $x_k$ of the n-tuple has a value that characterizes the particular element and a domain of permissible values. Some
scalars have initial values that never change while others have values that are continually changing. A scalar may refer to a property of the element, its priority, its time of entry into the system, its status, or its location in the system. We will characterize all objects in the system by a set of n-tuples, where n may be different for different objects.

**Subsets of n-tuples**

If the scalar \( x_k \) denoted weight or color of an n-tuple and this n-tuple characterized a queue, then the value of \( x_k \) would have no meaning. In general, if every element of a system were characterized by an n-tuple of exactly n scalars, many scalars characterizing a particular type of element would have no meaning. Elements with similar characteristics will be separated into subsets with a fixed number of scalars. If \( C_q \) denotes a subset q, the n-tuples would be written as follows:

\[
\begin{align*}
C_{q1}[x_1, \ldots, x_m] \\
\vdots \\
C_{qi}[x_1, \ldots, x_m] \\
C_{q1}[b_1, \ldots, b_n] \\
\vdots \\
C_{q2p}[b_1, \ldots, b_n] \\
\vdots \\
C_{qr}[t_1, \ldots, t_s]
\end{align*}
\]

where q denotes the particular subset, and r the member of that subset.
Translation

As time progresses in a system, operations occur. A rocket is launched, a blood sample is entered into the laboratory or a laboratory test result is recorded. We will account for these operations by changes in the value of the scalars of the n-tuples associated with the object. A change in the value of a single scalar will be called a translation. If $x_i$ is a scalar of the n-tuple $x_1, x_2, \ldots, x_i, \ldots, x_n$ then a translation on the n-tuple might result in the n-tuple $x_1, x_2, \ldots, x_i', \ldots, x_n$. Since an operation may result in many system changes, an operation in the object system may correspond to many translations among the set of n-tuples $S_o$. A specific translation does not necessarily operate on all n-tuples. It makes no sense for a ground crew to be launched. Further, the outcome of a translation must not result in scalar values that are outside the domain of possible values.

Precisely, a translation may be represented by

$$T[i, j, k, S]$$

where T denotes a translation, i denotes that the object is represented by an i-tuple and hence is in the subset of i-tuples for the system, j denotes the index of the member in the subset of i-tuples, k denotes the scalar used, and S denotes the substitution rule for replacing the scalar.

Substitution Rule

If $x_k$ is the scalar characterizing the time a rocket is launched, then the same substitution rule associated with $x_k$ would apply to
all i-tuples representing rocket ships. The substitution rule would be "replace the present value of \( x_k \) with the current time."

c. Digital Simulation Model

The number of scalars necessary to model every characteristic of an object or the number of n-tuples necessary to model to every element in the object system may be unlimited. When a digital simulation model \( S_m \) of an abstracted model \( S_o \) is constructed, the number of n-tuples and scalars of the model is bounded by the finite size and speed of the computers used for the simulation.

At this point in our discussion we will review our terminology. Three distinct systems are now being considered: 1. an object system, 2. the abstracted model, 3. the digital simulation model. Each of these systems has a corresponding terminology:

<table>
<thead>
<tr>
<th>Object System</th>
<th>Abstracted Model</th>
<th>Digital Simulation Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>collection of objects</td>
<td>set of n-tuples ( S_o )</td>
<td>set of n-tuples ( S_m )</td>
</tr>
<tr>
<td>object</td>
<td>n-tuple of ( S_o )</td>
<td>n-tuple of ( S_m )</td>
</tr>
<tr>
<td>characteristic</td>
<td>scalar</td>
<td>scalar</td>
</tr>
<tr>
<td>collection of objects with</td>
<td>subset of ( S_o )</td>
<td>subset of ( S_m )</td>
</tr>
<tr>
<td>similar characteristics</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In addition, time is measured differently in each of the three systems. The object system time is continuous and may be measured by a real clock. Model time is discrete and is measured by ordered
sequences of translations. Digital simulation time (the time it takes the computer to process the model) is continuous and may be measured by a real clock. We say that one hour of model time has passed when the translations that correspond to the operations in one hour of object time have been executed.

Timing of Operations

The timing of operations is controlled by elements of the system itself. Recognizing which elements control the timing of the operations in a system is of utmost importance to the understanding of the system. Correspondingly, the order of translations among the set of n-tuples, \( S_o \), should be controlled by the elements of the set \( S_o \).

d. Procedure for Constructing a Digital Simulation

We will construct a digital simulation for an object system in the following steps:

1. Choose the important elements and characteristics of the object system (the set \( S_o \)).

2. define the set of translations \( T[i,j,k,S] \) that correspond to the operations on these elements.

3. recognize the elements that control the timing of operations and define the timing of translations among the set \( S_o \).
4. define the set of n-tuples $S_m$ corresponding to the set $S_o$ and give initial values where necessary to the scalars of $S_m$.

5. write routines for the translations $T_m$ of $S_m$ corresponding to the translations $T_o$ of $S_o$.

6. implement a timing routine which will control the timing of the translations of $S_m$.

e. Application of a Model

A purpose of simulating a system is to answer questions about the performance of the system. What is the effect of various parameters on system performance? What is the performance of the system to a prescribed job load? What is the effect of various parameters of the components of the system? Are any components saturated or near saturation?

If a system is viewed as a set of n-tuples, the operation of the system may be analyzed quite simply. The performance of the system may be specified by a sequence of translations. The effect of various parameters of the components of the system may be investigated by varying the values of the scalars corresponding to these parameters. (For example, to study the effect of replacing a machine that nails heels on shoes with a machine that operates faster, we simply replace the scalars characterizing the time spent with each shoe with new values.) The effect of changing the route of flow of materials or information through the system may be investigated by changing the order of translations.

The saturation point of the system or a component is defined as that state of the system in which an addition of work to the system causes no additional output from the system. The saturation point
of a component of the system can be found in two ways. We can vary the values of the scalars of the n-tuple for the component and thereby determine the saturation parameters under a fixed load. We can also vary the number of n-tuples for the interacting components and thereby determine the job load that will saturate the component under study.

The remainder of the chapter will exemplify these points by discussing a digital simulation of the Clinical Chemistry Laboratory of the Yale New Haven hospital.

IV.2 Model for a Clinical Laboratory

a. Isolation of Laboratory functions

The main function of the clinical laboratory is to process biological specimens in order to extract medical information from them.

As with many complex systems the clinical chemistry laboratory is composed of two interrelated systems, a materials handling system and an information system. The information system may be subdivided into three information systems. An overall view of the four systems and their interrelationship is shown in Figure IV-3.
The first system, the materials handling system is concerned with the physical handling and processing of these specimens. This system handles inputs of specimens, chemical reagents, and energy. Its outputs are data and waste. The procedures that are followed in the laboratory comprise the second system, the control information system. The third system, called the management information system, is concerned with items such as personnel utilization and evaluation of changes in personnel, equipment, or procedures. The fourth system, called the medical information system, has as inputs data from the control management information systems. Its outputs are the compiled ward reports.

The model has enabled us to isolate and analyze many of the laboratory procedures and components. Thus the response to variations in the scalars of the model reflect the response of the system to
variations of characteristics.

For the remainder of this chapter we will consider only the materials handling system. The principal measures of system performance will be process time of biological samples, equipment utilization, job queues, response time to an emergency request.

b. Parameters of the Model

Some of the principal system parameters that were modeled by scalars are listed below:

A. Equipment parameters
   1. time to process an element
   2. capacity to hold unprocessed elements.
   3. number of elements which may be processed simultaneously

B. Software parameters
   1. priorities
   2. dynamic allocation of priorities. (The parameters for dynamic allocation of priorities enable us to study the effect of different scheduling policies.)
   3. modes of operation, i.e., "time slicing" and "demand operation." In time slicing mode each procedure (program) returns control to an executive system after a fixed amount of time. In demand operation mode control is returned to the executive program after each interrupt regardless of which program was being executed. These procedures control the printing
ward reports, patient reports, collecting data in real time, etc. (The effects on system performance in response to each of the modes was markedly different.)

C. Environment characteristics
1. number of technicions
2. number of centrifuges
3. number of stations

D. Computer parameters
1. speed of computation
2. the speed of various input out devices
3. storage size

c. Simulation Output
The modeler usually wants to know the response of his model to changes in the values of the parameters of his model. A straightforward way to provide this information is to print the status of the model at periodic intervals of time, i.e., by printing the values of pertinent scalars associated with each n-tuple. The table of Figure IV.4 below gives a portion of the output describing the state of the model of a clinical laboratory. This output is not in the best format but it does demonstrate how the viewpoint of systems presented earlier may be used to give meaningful output from a model. The sets of n-tuples presented in the table represent queues, storages, facilities, and controlling elements. The
<table>
<thead>
<tr>
<th>QUEUE NUMBER</th>
<th>MAXIMUM CONTENTS</th>
<th>AVERAGE CONTENTS</th>
<th>TOTAL ENTRIES</th>
<th>ZEROS</th>
<th>PERCENT</th>
<th>AVERAGE TIME/TRANS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.00</td>
<td>55</td>
<td>55</td>
<td>100.0</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0.78</td>
<td>47</td>
<td>2</td>
<td>4.3</td>
<td>64.91</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td>2.92</td>
<td>100</td>
<td>4.3</td>
<td>2.0</td>
<td>113.74</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>2.87</td>
<td>635</td>
<td>395</td>
<td>62.2</td>
<td>5.34</td>
</tr>
<tr>
<td>5</td>
<td>175</td>
<td>67.47</td>
<td>175</td>
<td>2</td>
<td>0.0</td>
<td>1503.64</td>
</tr>
<tr>
<td>6</td>
<td>170</td>
<td>66.90</td>
<td>173</td>
<td>0</td>
<td>0.0</td>
<td>1508.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FACILITY NUMBER</th>
<th>AVERAGE UTILIZATION</th>
<th>NUMBER ENTRIES</th>
<th>AVERAGE TIME/TRANS</th>
<th>SEIZING TRANS. NO.</th>
<th>PROMPTING TRANS. NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.54</td>
<td>218</td>
<td>3.51</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1.54</td>
<td>202</td>
<td>3.90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1.64</td>
<td>227</td>
<td>11.10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5.64</td>
<td>274</td>
<td>4.05</td>
<td>344</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4.96</td>
<td>448</td>
<td>5.56</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3.91</td>
<td>131</td>
<td>11.64</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>5.02</td>
<td>153</td>
<td>12.78</td>
<td>330</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>5.02</td>
<td>103</td>
<td>9.61</td>
<td>248</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>2114</td>
<td>103</td>
<td>6.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2.964</td>
<td>138</td>
<td>8.38</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUPPAGE NUMBER</th>
<th>CAPACITY</th>
<th>AVERAGE CONTENTS</th>
<th>AVERAGE UTILIZATION</th>
<th>ENTRIES</th>
<th>AVERAGE TIME/TRANS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32767</td>
<td>2.04</td>
<td>0.901</td>
<td>202</td>
<td>39.40</td>
</tr>
<tr>
<td>2</td>
<td>2264</td>
<td>2.46</td>
<td>0.491</td>
<td>92</td>
<td>117.06</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>5.43</td>
<td>0.645</td>
<td>139</td>
<td>117.43</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>11.46</td>
<td>0.914</td>
<td>113</td>
<td>3437.64</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>9.14</td>
<td>0.7033</td>
<td>13</td>
<td>2742.85</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>7.64</td>
<td>0.5744</td>
<td>13</td>
<td>2200.85</td>
</tr>
</tbody>
</table>
pertinent scalars associated with each queue n-tuple are maximum contents, total entries, etc. The name of the laboratory components that correspond to these n-tuples may be found by using the first part of Appendix D (e.g., QUEUE Number 2 corresponds to the line of whole blood biological samples at the punch bench). A modeler may use the characteristics of a queue to determine if a queue is progressively building up and thereby become saturated.

IV.3 Digital Simulation of Laboratory

We now outline the steps 1, 2, and 3 in section IV.1.d for an example simulation. Steps 4, 5, and 6 will be given in section IV.4.

1. Choice of important elements and characteristics in object system

The choice of important elements and characteristics in the object system may be very difficult. The level of detail, the elements, the characteristics chosen is dependent of the questions to be investigated when using the model. With respect to the laboratory we wished to answer questions which dictated a very detailed simulation (e.g., the effect of various parameters of the components of the system, the organization of the laboratory, saturation of components.

We wish to choose a set of n-tuples to characterize the chemistry laboratory and group the n-tuples into subsets, where the elements in a subset have the same number of scalars. Each subset will generally represent one type of element. The most obvious sets of elements in the laboratory are technologists, biological samples, and equipment. Other sets of elements include queues and storages.

The subset of the n-tuples representing biological specimens will be represented by a set of k-tuples \( x_1, x_2, \ldots, x_k \), for some number \( k \). Each scalar characterizes a particular sample and takes on values from a finite, discrete domain. The first scalar has a binary domain with value 1 if a urea test was requested and a
pertinate scalars associated with each queue n-tuple are maximum contentents, total entries, etc. The name of the laboratory components that correspond to these n-tuples may be found by using the first part of Appendix D. (e.g., QUEUE Number 2 corresponds to the line of whole blood biological samples at the punch bench). A modeler may use the characteristics of a queue to determine if a queue is progressively building up and thereby become saturated.

IV.3 Digital Simulation of Laboratory

We now outline the steps 1, 2, and 3 in section IV.1.d for an example simulation. Steps 4, 5, and 6 will be given in section IV.4

1. Choice of important elements and characteristics in object system

We wish to choose a set of n-tuples to characterize the chemistry laboratory and group the n-tuples into subsets, where the elements in a subset have the same number of scalars. Each subset will generally represent one type of element. The most obvious sets of elements in the laboratory are technologists, biological samples, and equipment. Other sets of elements include queues and storages.

The subset of the n-tuples representing biological specimens will be represented by a set of k-tuples $x_1, x_2, \ldots, x_k$, for some number k. Each scalar characterizes a particular sample and takes on values from a finite, discrete domain. The first scalar has a binary domain with value 1 if a urea test was requested and a
value 0 if it was not, and so forth for the remaining laboratory tests. The nineteenth scalar denotes the time of entry of the sample into the system, and its domain may be chosen as the index of one of the .1 minute intervals between 8:00 a.m. to 5:00 p.m.

Similarly other scalars of this subset of n-tuples are used to model other sample characteristics.

Some scalars have fixed values while others have values that change with respect to time or some other parameter of the system. For example, the number of samples that a technician has yet to process varies with time. In the model of the clinical laboratory, the assignment of a value to a scalar that varies with time or some other parameter is implemented with the use of the VARIABLE feature or FUNCTIONS feature (See Appendix D). For example, it was observed that the technician operating the chloride machine took an increasing length of time to place a sample in the machine as the day went on. This time may be approximated by the function shown above. This was implemented in the model (see FUNCTION number 71, Appendix D). It is interesting to note that when
we wished to investigate the effect of replacing this technician with a machine the only characteristic that had to be changed was the time to place a sample in a machine. This was accomplished by simply replacing the FUNCTION definition card with another one characterizing time for a machine sample in the machine.

2. Definition of translations:

The set of translations are chosen to simulate the operations of the object system, in this case the laboratory. One such operation is the change of a sample's priority. In the corresponding translation, T[i,j,k,S], i would denote the subset of n-tuples representing biological samples, j would denote the particular n-tuple, and k would denote the scalar characterizing priority. The substitution rule would be: "replace the value of scalar k with the value x" where the value of x is assumed to be known.

An operation that corresponds to many translations is the recording of arrival times of biological specimens. For one translation we let i and j denote the n-tuple characterizing the processing technician and k denote the scalar characterizing his state (busy or not busy). The substitution rule would be as follows: If the scalar designated by k is in the "busy" state, then this translation cannot be performed; if the scalar is in the "not busy" state, then the value of the scalar is given a new value to denote a busy state, and other translations on the same n-tuple for the technician are performed. Thus the values of the scalars characterizing utilization and other statistics are updated. The recording of arrival times must

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
also have a corresponding translation that updates the n-tuple characterizing the biological specimen waiting to be time punched.

3. Recognition of elements that control timing:

As stated previously, it is important to recognize the elements that control the timing of operations in the object system. Correspondingly, the n-tuples representing these controlling elements should control the timing of translations in the set $S_m$. The major controlling elements in the materials handling information system of the clinical laboratory are the biological specimens. If no samples arrive then no operations occur, except perhaps the technician's lunch or coffee breaks. If the sample is sent as an emergency it is handled accordingly.

The overall timing and flow of the materials handling is shown in Figure IV-5. Using the concept that certain elements control the timing, the laboratory may be viewed as a permanent structure through which biological samples flow. The samples arrive (point 1 in Figure IV-5) and are time stamped by a technician. Depending on the test requested, the samples go either to the technicians processing whole blood (point 2) or to the centrifuging area (point 3). At these points the samples are processed by personnel and equipment, and then continue on to a specific test area (points 6 through 12).

The n-tuples representing the biological samples must cause translations that change the values of their scalars characterizing their position in the system.
Figure IV-5 Overall view of the clinical chemistry laboratory (for handling blood samples).
IV.4 Implementation

The implementation of the simulation comprises steps 4, 5, and 6 of section IV.1.d. In the discussion so far, simulation has been described independently of any computer or computer language. To implement the simulation on a digital computer, computer languages such as ALGOL, MAD, or FORTRAN could be used. However, a great programming effort would be necessary to implement the set of n-tuples, translation routines, and timing routines. Since a simulation language has many of these routines incorporated in the language, using one of these languages is usually easier. The reason for using a simulation language rather than any other computer language is that many of the routines which implement the set of n-tuples, translations, and timing are written in various degrees of completeness depending on the language. For example, GPSS has entities that may readily be used to represent n-tuples.

A detailed model of the Clinical Chemistry Laboratory of the Yale-New Haven Hospital has been constructed using a modified* version of the computer simulation language GPSS III [9, 10, 71]. Parts of the laboratory were also simulated in another simulation language SIMSCRIPT [72]. The entire model is presented in Appendix D.

a. Implementing n-tuples

If FORTRAN or MAD is used, the n-tuples may be defined by a

* Modifications included the adaptation of GPSS to run on an IBM DCS 7094-7040 system [73], the adaptation of GPSS to handle a program as large as the model for the laboratory, and the addition of instructions to facilitate handling of sets.
DIMENSION statement. A(1), A(2), ..., A(20) would correspond to an n-tuple $a_1, a_2, ..., a_{20}$ and A(1) would refer to the scalar $a_1$. If a simulation language is used it is not always the task of the modeler to define the set of n-tuples $S_m$ that correspond to $S_0$. The language GPSS has predefined entities that may be used as n-tuples representing elements of an object system.

b. **Implementing Translations**

If FORTRAN, MAD, or SIMSCRIPT is used to implement the translations, $T[i,j,k,S]$, subroutines would have to be written that would operate on all scalars of the n-tuples i,j. The substitution rule would define the algorithm to be used by the subroutine. A modeler using SIMSCRIPT follows a similar procedure for most translations.

The simulation language, GPSS, provides several of the routines necessary to implement translations. The statement "SEIZE m" is used to specify operations on the subset of n-tuples for technicians; m is the index in the set of the n-tuple for a particular technician. The execution of this instruction corresponds to many translations, $T[i,j,k,S]$.

Part of the substitution rule for one of the translations corresponding to "SEIZE m" is: If the state of the scalar $x_k$ of $C_{im}$ is "not busy" change it to a busy state.

The substitution rules for the remaining major translations

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
associated with this instruction state the manner in which values
of scalars characterizing the utilization and other statistics
of the technologist change.

The translations that correspond to the flow of elements
operate on the scalars that characterize the location of the
elements. The resulting location is the object system may depend
on many other scalars.

For example, a sample for which an electrolyte test has been
requested goes to one of the spinners in the system. In the
Translation T[i,j,k,S] for this i denotes the subset of n-tuples
representing samples, j denotes the sample involved, and k denotes
the scalar characterizing the location of a sample in the system.
The substitution rule if "if an electrolyte test is requested (i.e.,
if the binary-valued specifying whether or not an electrolyte test is
requested has a value of 1) change scalar x_k to some value corresponding
to location of spinner ^1. If not change x_k to some value corresponding
to location of spinner ^2." GPSS has a routine to implement this
translation. The routine is called by the instruction "TEST E P3, KO,SPINI"
which is read, "test if scalar #3 of the n-tuple for this instruction
is equal to a constant number 0; if not go to SPINI, otherwise,
execute next instruction." Scalar #3 of a sample will have been
equal to 1 if an electrolyte test is requested and equal to 0
otherwise.

c. Implementing Timing Routines

If a modeler uses FORTRAN or MAD he must write all timing
routines himself. If, however, he uses a simulation language he may not have to write all the timing routines. SIMSCRIPT has an "EVENTS" routine that implements the timing of translations. GPSS has an "overall GPSS scan" routine that implements the timing of translations.

A summary of the algorithm that GPSS uses to control the timing of translations is as follows: Two chains are established in the memory of the computer, a current events chain and a future events chain. The members of these chains are the n-tuples that correspond to the elements that control the timing and flow of the object system. In the GPSS model of the clinical laboratory the majority of the members of these chains correspond to the biological samples. The members on the current events chain are arranged in descending priority classes and, in the case of equal priorities, in the order in which they were linked to this chain. Starting at the highest priority class, the overall GPSS scan moves (translates the scalars characterizing the location of elements in the system) the members of the current events chain through the model (causes the execution of translations) until they are delayed or limited to the future events chain. The simulation clock is then updated to the value at which the first element on the future events chain is due to leave that chain. This element is then linked to the current events chain. The scan again translates the n-tuples on the current events chain as far as they will go. This process is repeated until the simulation is terminated.
IV.5. Other Examples

a. Example of Different Controlling Elements

In many complex systems the timing of all operations is not necessarily controlled by the same element or the same type of element. It is important in the simulation of these systems to recognize all controlling elements. As was previously stated, there are operations in the laboratory which would take place even if no samples arrive, such as lunch and coffee breaks. Therefore, samples do not directly control these operations. In the clinical laboratory, the lunch break does not occur at exactly twelve o'clock but rather some time after twelve o'clock. The factors that govern the time of the lunch break are the urgency of work waiting to be processed by the technologist, or the partial processing of a task that the technologist does not wish to interrupt for lunch.

In the translation $T[i,j,k,S]$ that corresponds to the operation of a lunch break $i$ denotes the subset representing technicians, $j$ denotes the $n$-tuple the particular technician involved, and $k$ denotes the scalar characterizing the state "out to lunch - not out to lunch". The substitution rule is simple "change the value of scalar $x_k$ to the value denoting "not out to lunch".

The problem is of how to control the timing of this translation. The samples do not directly determine the time of the lunch break nor does the technologist. Even if no samples were in the laboratory the lunch break would still occur. To postulate that it is the hunger of the technologists that control the timing of the lunch break, receives the reply that many technologists are hungry as soon
as they arrive at the laboratory. What then controls the timing? A signal from the control information system. This signal is generated at twelve o'clock, and has a designated priority. When nothing else of higher priority is left to be processed and the technician is not engaged in a task, the signal causes the lunch break.

The lunch break for our hungry technician may be simulated in GPSS by the statements below. The control system's signal can be thought of as flowing through these statements. Each statement calls a routine that is an implementation of the translations associated with the operation of a lunch break.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Arguments</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERATE 668</td>
<td>0,0,7200,1,6</td>
<td>Create an n-tuple that corresponds to a signal with priority 6 generated at 12 noon.</td>
</tr>
<tr>
<td>SEIZE 669</td>
<td>1</td>
<td>When nothing of higher priority has seized technologist #1, translate the scalar characterizing the state of technician #1 to a &quot;busy state.</td>
</tr>
<tr>
<td>ADVANCE 700</td>
<td>300</td>
<td>Advance after time for lunch. 300 units of time.</td>
</tr>
<tr>
<td>RELEASE 701</td>
<td>1</td>
<td>Hunger satisfied, let technologist return to system.</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The timing of translations is controlled by the algorithm previously mentioned in Section IV.4.a. Applying this algorithm the following will result. Statement 668 causes the n-tuple representing the signal for a lunch break to be linked to the future events chain, and at simulation time 7200 the n-tuple is linked to the current events chain. The scalar characterizing the n-tuple's position on the current events chain will indicate a priority class 6 on that chain (the highest priority of the system is 7--few n-tuples have a higher priority than the n-tuple representing the controlling element of lunch).

When the simulation clock is updated to 7200, the overall scan will, starting at the highest priority class, give the value of i and j in T[i,j,k,S] (the n-tuple on the current events chain with highest priority) to the statement 689 SEIZE 1. The routine SEIZE 1 implements many translations, as has been discussed (Section IV.11,b). The result of the execution of this statement corresponds to the setting of the scalar $x$, for the technician to a "busy" state. If this scalar already has a value corresponding to "busy" processing of this n-tuple is stopped. Otherwise the GPSS scan will then give the value of i and j of the n-tuple characterizing the signal to the next statement ADVANCE 300. Execution of this routine links the n-tuple for the signal to the future events chain to be removed after 300 units of simulation time. After 300 units of simulation time the GESS scan links the n-tuple characterizing the signal back to the current events chain. Instruction "RELEASE 1"
is then executed resulting in setting the scalar x to the value denoting "not busy".

b. Aspect of System Not Directly Expressible in GPSS

As is the limitation with any simulation language, some important aspects of a system are not directly expressible in the language. The class of operations for manipulating sets is not expressible using GPSS. Among these are operations that manipulate collections of elements (sets, queues, storages, files).

For example, consider a tray brought to a work bench after the samples have been centrifuged. The corresponding translations required are to operate on all n-tuples that are members of the n-tuple representing the collection of samples. The timing of this operation is caused by the number of samples themselves. If the tray is full or if there are no more samples left to process the samples go to the bench.

In GPSS there is no direct way of implementing the translations and timing associated with this operation. The following instructions were used to effect this operation:

<table>
<thead>
<tr>
<th>GPSS Instruction</th>
<th>Label Instruction</th>
<th>Arguments</th>
<th>Corresponding operation in object system</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>324 SEIZE</td>
<td></td>
<td>3</td>
<td>Seize technician who centrifuges</td>
<td></td>
</tr>
<tr>
<td>325 GATE SF</td>
<td></td>
<td>3,NOTEFL</td>
<td>Is tray#3 full? If not go to NOTFL.</td>
<td></td>
</tr>
<tr>
<td>326 ADVANCE</td>
<td></td>
<td>10</td>
<td>If tray is full, bring to bench</td>
<td>(this operation takes 10 units of simulation time)</td>
</tr>
</tbody>
</table>
327  UNLINK  4,BENCH+1,ALL  Cause all samples on chain 4 to go to bench.

329  BENCH  RELEASE 3  Samples now at bench, release technician who centrifuges.

357  NOFUL  TEST NE V5,KOBENCH-1  Are there any more samples in system? If not go to BENCH-1.

358  RELEASE 3  If there are more samples in system release technician who centrifuges.

358  LINK  4,FIFO  Link to chain \( \neq 4 \) and wait.

The instruction in GPSS to test whether a tray (i.e., a GPSS) is full is written "GATE SF 3,NOTFUL." This instruction is read "If gate storage\( \neq 3 \) is not full, go to NOTFL; otherwise execute the next instruction."

The instruction "UNLINK 4,BENCH,ALL" causes all the members of chain \( \neq 4 \) to have the scalars signifying their location changed to the value corresponding to the location BENCH.

The instruction "LINK 4,FIFO" causes the n-tuple executing this instruction to be linked to chain \( \neq 4 \) in the order of first in-first out bases.

V5 in the statements above is set equal to the number of n-tuples corresponding to samples that have not caused the execution of the
routine above but are still in the model.

IV.6 Validity of a Simulation Model

When constructing a model the question of the validity of the model must be dealt with. Can the modeler be absolutely certain that results derived from his model are applicable to the object system?

There is no known method of proving the validity of a model in general. However, two factors ought to be considered. First, the link between the object system and the abstractions may be weak because of exclusion in the abstractions of important elements or characteristics that influence the behavior of the object system. Failure to define all translations or define the controlling elements and timing may introduce errors associated with this link. Second, all n-tuples, scalars, and translations of the set $S_o$ may not be represented among the set $S_m$. This difference will introduce other errors.

The fact that the validity of a model cannot be shown in general does not mean that confidence cannot be placed in a particular model. To give confidence to the validity of a model comparisons between the operation of the object system and the model under known conditions may be made. An example of a set of comparisons to give confidence in the model of the laboratory is shown in Table IV.1. A more complete set is given in Appendix C. The comparisons matched the performance
of the laboratory for a particular day with the performance of the model for that same day. A record was kept of the arrival time and tests requested of samples entering the laboratory that day. Every half-hour a set of statistics of the actual laboratory was obtained. The numbers in the left column correspond to the points in Figure IV-4, indicating the approximate place in the system that these statistics were taken. Above each column the time at which the statistic was observed is given.

The model was given the same inputs as were recorded for a day in the object system. At every half-hour of simulation time the current status of the model was printed. These statistics were compared with statistics taken of the actual laboratory at corresponding times. The simulation results are listed under the actual laboratory observations in Table IV-1.

For example, the number of samples logged by the electrolyte spinner at 10:30 a.m. was 55 in the actual laboratory and 56 in the model.

The excellent correlation between object system data and model data gave confidence in the model.

IV.7 Information System

A typical clinical laboratory, with a given set of facilities, invariably reaches a state beyond which the laboratory becomes incapable of servicing its demands or becomes incapable of supplying the quality of work it is doing. New instruments, automation of instruments, reorganization of the laboratory, automatic data handling equipment, are all possible methods of improving the laboratory. However, incompatibilities and inconsistencies with the idea of maintaining testing standards arise. The simulation model was used to test each of these possible methods of improvement. We will only discuss in this work the practicability
Comparison of actual laboratory events and times of occurrence with those simulated

<table>
<thead>
<tr>
<th>TIME (hours)</th>
<th>a.m.</th>
<th>p.m.</th>
</tr>
</thead>
<tbody>
<tr>
<td>****</td>
<td>8</td>
<td>8.5</td>
</tr>
<tr>
<td>Number of samples</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>given accession</td>
<td>53</td>
<td>97</td>
</tr>
<tr>
<td>logged by nonelectrolyte spinner</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>spun and put in</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>ready rack</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>126</td>
</tr>
<tr>
<td>9.5</td>
<td>17</td>
<td>131</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>53</td>
</tr>
<tr>
<td>11</td>
<td>21</td>
<td>53</td>
</tr>
<tr>
<td>11.5</td>
<td>40</td>
<td>54</td>
</tr>
<tr>
<td>12</td>
<td>53</td>
<td>54</td>
</tr>
<tr>
<td>12.5</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>1</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>1.5</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>2</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>2.5</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>3.5</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>4.5</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>54</td>
</tr>
</tbody>
</table>

**TABLE IV-1**
| Time (hours), topline, refers to the time the count of events was taken. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Table IV-1 continued** | a.m. | p.m. |
| | 8 | 8.5 | 9 | 9.5 | 10 | 10.5 | 11 | 11.5 | 12 | 12.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 |
| Number of samples read at bicarbonate test | 0 | 0 | 0 | 4 | 8 | 15 | 20 | 25 | 34 | 40 | 45 | 52 | 60 | 65 | 73 | 77 | 83 | 86 |
| s | 0 | 0 | 0 | 3 | 7 | 14 | 21 | 25 | 33 | 39 | 47 | 54 | 62 | 67 | 74 | 81 | 84 | 87 |

**See Figure 1 for numbers in this column.**

***Items in this column are descriptions of the number of events.***

- a refers to the number of events observed at the time noted.
- s refers to the number of events determined by simulation.
of introducing data processing equipment into the laboratory. If the
model were used to gain insight to only the introduction of data
processing equipment the level of detail which we have simulated the
laboratory would not be necessary (e.g., all the queues, equipment,
technicians associated with a test bench could be replaced by a delay
function by which the samples were detained). However, we wished to use
the model to gain insight into other questions as well.

The equipment would be introduced into the medical information
system. The present system is inadequate to meet future demands and
automatic data processing seemed an idea worth investigating.

As can be seen from Figure IV-4, the output information (data)
from all test areas goes into the medical information system. With
the proposed data processing equipment the information will go to a
digitizer via a multiplexing unit and then into a computer.

Functions of computer: Some of the functions of the computer
are the following: To collect the information extracted from the
samples and standards; to calculate the results of tests by executing
the appropriate program stored on a memory unit; to continuously
monitor all information received in order to detect when all tests
on a particular sample have been processed; to compile and print
patient and ward reports; to store results on any available storage.
Provision must also be made to process a request for a partial
report of a patient's tests before all tests of that patient are
processed. The computer may also be used to compute other information
such as statistical analysis, or for executing research programs.
IV.8 Simulation of Medical Information System

To simulate the medical information system the six steps, 1. definition of the set $S_0$, 2. definition of translations, 3. timing, steps 4, 5, 6 implement steps 1, 2, 3, are employed.

The set of n-tuples corresponding to all the elements of the objects system (information handled, computer, peripheral equipment, queues) must be defined.

The translations corresponding to the operations of the object system (compiling results of tests, print ward and patient reports and all the operations of the computer) must be defined.

The n-tuples representing information of the biological samples are the controlling elements.

The scalars of the n-tuple representing the information are given values to denote the origin of their test area, which sample the n-tuple represents, the number and kinds of other tests that were requested of the same sample, and other characteristics.

These n-tuples, translations, and timing are implemented on a digital computer in the same manner discussed for the materials handling system.
In the clinical chemistry laboratory, the executive program must perform the following tasks: Control the flow of information in and out of the computer; govern the flow of programs from storage to compute results of test; allow for interruption of activities for emergencies; provide flexibility in the rescheduling of tasks; schedule printing of patient and ward reports.

The assignment of priorities to the tasks of the computer presents a critical problem. For example, if the computer is performing one task of priority 2, then any other task of a higher priority can interrupt the computer and be processed. If too low a priority is given to the task of compiling and printing emergency reports, then too much time might elapse between the request for the report and the printing of the report. If information is ready to enter the computer and low priority has been given to this transaction, then the information may be lost before the executive program allows the computer to accept it.

A digital simulation can be used to explore the assignment of effective priorities. This assignment was implemented into our model by assigning an appropriate value to the priority scalar of the n-tuples characterizing the information of the biological samples. The information n-tuple of the biological sample originates at a test bench which has just processed the biological sample. It is at this point in the model that a priority will be assigned to the information n-tuple. The information from different test benches were given different priorities. Other tasks of the computer such as a doctor's request for a partial patient report were implemented into the model by generating a n-tuple corresponding to the doctor's request.
In the first set of assignments tried, the task of printing a patient report was given highest priority; thereby these results would be obtained immediately after compilation of all tests for that patient. This assignment was found unsatisfactory in that input results from individual test areas were being delayed too long before being accepted by the computer. The outcome was an overflow of the input buffer when other information from test areas was received. This loss of information is intolerable in a hospital.

Assignments that were marginal were found. That is, they would work for the present system but did not allow for an increase of work. These were found by noting the frequency with which the simulated computer was preempted.

After exploring other assignments, the following assignment proved to be satisfactory and allowed for future increase of the laboratory's work load. Highest priority was given to information in the input buffer whenever more information was ready to enter the buffer. Descending priorities were then assigned to the following: A request for a partial patient report, computation of results of a test, compiling and printing of patient reports, and storage of these results on any available storage, accepting data from the buffer of the multiplex unit, compiling and printing ward reports.

The tasks of billing and other low priority work were not considered in the model. However, satisfactory times of day for scheduling these tasks were found using the model. Low utilization times may be seen from Figure IV-6; these are opportune times for scheduling other work. Figure IV-7 shows computer utilization for each hour versus time of day as given by the simulation.
Figure IV-6  Utilization of computer time versus time of day
Note further the utilization never goes above 50 per cent. This demonstrates the system's ability to handle an expansion of the laboratory's work load if a suitable executive system is used, even though the computer simulated is considered to be a small computer.

Summary of Results from the Model:
The simulation model has given insight into the answers to the problems of introducing data processing equipment into the laboratory. It has uncovered problems that were not originally thought of. It has shown that a small computer (IBM 1130) with one disk file and a Dymec data-logger are adequate. It has shown the requirements of the executive program for the laboratory.

The model has shown that the "time slicing mode" mentioned earlier resulted in the shortest external queues of tasks to be performed by the computer. Thus technician waiting time, for the computer, was brought to essentially zero. The model showed that the average per hour utilization of the computer could be made less than fifty percent, thereby giving us further confidence in making the decision of using "time slicing mode" for this mode does use more computer time to maintain internal queues of task request or data.

In the future, the simulation model will be used further to study the efficacy of modifications to the laboratory.
CHAPTER V

CONCLUSIONS AND PROPOSALS FOR CONTINUED RESEARCH

V. 1 Results

This dissertation has been concerned with the application of the procedure depicted in Figure I-1 (page 8) to the digital simulation of complex systems such as a hospital, bank, or factory.

Chapters II and III were concerned with a major problem associated with the syntax of the computer languages used in this procedure, and more generally, with the syntax of any computer language. As evidenced by the works cited in Chapter II there has not been available prior to this writing, a specification that has been used to characterize completely the syntax of computer languages. This investigation has applied known methods of recursive definition to the development of a complete syntactical specification of computer languages.

The specification may be used by a programmer to check the syntactical validity of a string. The specification is more compact, concise, and accurate than a manual.

The specification may be used to define all acceptable strings of a proposed compiler. Hence, decisions affecting membership in the class of legal programs are not made during the construction of the compilers. In this manner the specification may assist the compiler writer.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
A compiler may be viewed as having two phases, a syntax phase and a translation phase. The syntax phase is concerned with recognizing the well formed strings of its source language. The specification may be used in the construction of compilers whose syntax phase is independent of the source language being translated. That is, several source languages may be used with the same compiler.

A specification of the syntax of a limited portion of FORTRAN has been presented. The restriction that all statement labels in the same program must be different and the restriction that all reference labels must correspond to statement labels has been defined using this specification.

A complete syntactical specification of a computer simulation language, GPSS III, has been presented in Chapter III and Appendix A. No other complete or accurate characterization of any form exists for the syntax of GPSS III. The syntax of GPSS is extremely complex as is exhibited by the length of the manuals: [9] is 245 pages and [10] is 103 pages.

It was necessary to construct many experimental programs because many constructions of GPSS were incorrectly or ambiguously stated in the manuals. The major results of these experiments was that the syntax is in general less restrictive than is stated in the manuals. Statement labels, reference labels, arguments, and column format were found to have more flexibility in their form. A GPSS programmer might be
surprised to learn that the statement

\[ A$. )/B \text{ TEST } E \$A$. )/B , $A$. )/B +1, P1 \]

is a legal statement.

It has not been made clear in the manuals for all statements which arguments in these statements must not be void and which may either be void or not void. Each argument was experimentally tested to determine if it may be void. Examples of such statements have been given in Appendix B.

In a few instances it was found that the manuals were not restrictive enough. Some statements (such as, "blank cards or any character in column one are permissible," and "an ORIGINATE block is permissible") made in the manuals are not true, and programs containing the stated constructions will be rejected.

All incorrect or ambiguously stated constructions have been correctly defined by the specification of GPSS III presented in this dissertation. Some of the results of these experiments have been discussed in Section III.12 through III.14 and Appendix B. In Appendix B experimental verification and examples of some of these constructions are also given. In Appendix D more conventional syntactical forms of GPSS III are presented.

In Chapter IV a way of viewing complex systems has been presented. This viewpoint is composed of a set of fundamentals which are basic to complex systems and may be used to facilitate the understanding or
simulation of these systems. These fundamentals form the abstraction depicted in Figure 1-1.

Techniques have been presented in Chapter IV which may be used to implement this set of fundamentals on a digital computer.

These fundamentals and techniques are not dependent on a specific computer language, or system, and so do not suffer the limitations discussed in Section 1.5 associated with such a dependence. Since these fundamentals and techniques are general, a modeler using them need not confront a completely new problem with each system simulated; hence, a saving of man hours can be realized.

A non-trivial, real and important example of the use of digital simulation has been presented in Appendix C. Results of a digital simulation of the Clinical Chemistry Laboratory of Yale-New Haven Hospital have been discussed, and the method used to obtain these results has been discussed. Insight into the many problems of introducing data processing equipment into the laboratory has been obtained from the model.

A complete well documented model of the Clinical Chemistry Laboratory of the Yale-New Haven Hospital has been presented in Appendix D. The model may be used to study further simulation techniques, or to understand the operation and procedures of the laboratory, or as an example of forms of the syntax of GPSS more conventional than these examples given in Appendix B.

A bibliography has been presented which is a survey of the salient
existing works which are related to this dissertation. The bibliography may be used independently of the text of this dissertation. With each related group of references a brief note has been added stating the principal result of those works and their relationship to the work done in this dissertation.

V. 2 Proposals for Continued Research

There are several areas of the work presented in this dissertation which may be extended for further research. These areas will be listed as separate proposals.

a. A fruitful investigation would be the development of a "syntax directed recognizer." That is, a program that would accept any canonic system of a language and then proceed to recognize strings of that language. Implicit in this development is the construction of a set of general decomposition rules. These rules and the canonic system of a language would govern the procedure for decomposing strings of that language into their basic components.

b. A connection should be established between canonic systems and the work of Chomsky. This connection would be a valuable one because Chomsky's work, automata theory, and linguistics have all been linked by recent works [41-61]. This connection would make many of the theorems associated with the linking of Chomsky's work, automata theory, and linguistics applicable to canonic systems. And since canonic systems are the only known specifications that have been
used to characterize completely existing computer languages, this
connection would make some highly theoretical work applicable to
practical computer languages.

One approach to establishing this connection would be finding the
place of canonic systems in Chomsky's hierarchy of languages. One
theorem that should be proven is "A canonic system using predicates of
degree $n$ corresponds to a Turing machine," where $n$ is some number.

c. Canonic systems as presented in Chapter III are ambiguous.
That is, there may be several ways of generating the same string using
a canonic system. A fruitful study would be to make canonic systems
unambiguous. An approach to making canonic systems unambiguous
may be the addition of a set of precedent rules to canonic systems as was
done by Wirth and Weber [64] to their specification.

d. Since a canonic system completely characterizes the syntax of
a computer language it is hoped that canonic systems of computer
languages may be used to gain some insight into the structure of computer
languages.

e. Using the fundamentals and techniques of simulation discussed
in Chapter IV a savings in the number of man hours necessary for the
construction of a model may be obtained. A further savings would be
obtained if results of the model would be put in a more suitable form.
A fruitful study would be an investigation of the feasibility of visual
displays or graphical output to give a closer interaction with the modeler
and the model.
The type of system that this dissertation has been concerned with was a complex system, such as a hospital, bank, or factory. The viewpoint presented in Chapter IV facilitates the understanding and simulation of such systems. A fruitful study would be to apply this viewpoint to other types of systems such as systems described by differential equations.
APPENDIX A

ALPHABETIC LISTING OF THE

SYNTACTICAL SPECIFICATION OF

GPSS III

In this appendix is presented the complete syntactical definition of GPSS III. The canons or rules are listed alphabetically with respect to the name of the strings they define. For motivation and interpretation of the specification consult Chapter III.

alpha numeric

any length card program

arg num or ref label

arg sna or ref label

arg sna* sna or nonrelative ref label

x letter; y digit; z alpha numeric | A, zy, zx alpha numeric

a#b program; s spaces or void | as#b any length card program

n number; a plain ref label base b; c indirect ref label base d | n arg num or ref label A; a arg num or ref label b; c arg num or ref label d

a stand num attrib sna; x ref label base y | a arg sna or ref label A; x arg sna or ref label A

x stand num attrib sna* sna; a, b, c letter; d, e, f, g, h ref label sym | x arg sna* sna or nonrelative ref label A; abcde arg sna* sna or nonrelative ref label abcd; $adefgh arg sna* sna or nonrelative ref label adefg; W$adefgh arg sna* sna or nonrelative ref label adefgh; N$adefgh arg sna* sna or nonrelative ref label adefgh
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arg sna* sna or ref label</code></td>
<td>a stand num attrib sna* sna; b ref label base c</td>
</tr>
<tr>
<td><code>arg void num or ref label</code></td>
<td>a arg num or ref label b</td>
</tr>
<tr>
<td><code>arg void sna or ref label</code></td>
<td>a arg sna or ref label b</td>
</tr>
<tr>
<td><code>arg void sna* sna or ref label</code></td>
<td>a arg sna* sna or ref label b</td>
</tr>
<tr>
<td><code>arg void , sna* sna or ref label</code></td>
<td>x arg void sna* sna or ref label y</td>
</tr>
<tr>
<td><code>cards</code></td>
<td>a_{1}a_{2} \ldots a_{80} sym or space; b cards</td>
</tr>
<tr>
<td><code>comment program</code></td>
<td>a#b program; s spaces; f string</td>
</tr>
<tr>
<td><code>digit</code></td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9 digit</td>
</tr>
<tr>
<td><code>eighteen sym or spaces</code></td>
<td>a_{1}a_{2}a_{3} \ldots a_{18} sym or space</td>
</tr>
<tr>
<td><code>eighty column fnc data card with ref labels</code></td>
<td>a fnc data card with ref labels b; as eighty sym or spaces; s spaces or void; c eighty column fnc data card with ref labels d</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
field with num  abc five sym or spaces; b number; a, c spaces or  
  void | label1 field with number b

five sym or  a, b, c, d, e sym or space | abcde five sym or spaces
spaces

fnc card with  a field with num b; c function code; d arg sna or ref
ref label  label e | a FUNCTION? c, d# fnc card with ref label e

function codes  n number | Cn, Dn, Ln, En, Mn function codes

fnc data card  a fnc data field sna or ref label b; c fnc data field sna
with ref labels  or ref label; d; e fnc data field sna or ref label f;
g fnc data field sna or ref label h; i fnc data card
with ref labels j | acdg fnc data card with ref
labels b, d, f, h, ; iac fnc data card with ref labels jb, d,

fnc data field  first premises  a, c spaces void; s stand num attrib
sna; —

  abc six sym or spaces; b fnc label | abc fnc data field sna

  asc six sym or spaces | asc fnc data field sna

  ae, fc six sym or spaces; e, f number void | ae, fc fnc data field sna
  or ref label

  or ref label

end premises

fnc def with  a fnc card with ref labels b; c eighty column data cards
ref label  with ref labels d | A fnc def with ref labels A ; ac fnc
def with ref labels b, d,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fnc label</td>
<td>x label; a, b, c letter; d, e, f sym or space or void</td>
</tr>
<tr>
<td></td>
<td>x abcd ef fnc label</td>
</tr>
<tr>
<td>gate mnemonic</td>
<td>M, N, U, NU, I, NI, SF, SNF, SE, SNE, LR, LS gate mnemonic</td>
</tr>
<tr>
<td>gate mnemonic field</td>
<td>abc six sym or spaces; b gate mnemonic; a, c spaces or void</td>
</tr>
<tr>
<td></td>
<td>abc gate mnemonic field</td>
</tr>
<tr>
<td>ibsys tape arg</td>
<td>SYSOU2, SYSLB2, SYSPP2 ibsys tape arg</td>
</tr>
<tr>
<td>identify card program</td>
<td>xabsd#y wae#y legal info card program; a1a2a3 \ldots a9 sym space; a eighteen sym spaces; b string; e symbols; s space xabsd#ya2\ldots a9#y identify card program</td>
</tr>
<tr>
<td>indirect ref label base</td>
<td>x label; a letter; b, c, d, e, f ref label sym; n number;</td>
</tr>
<tr>
<td></td>
<td>$x$ indirect ref label base x; abcdef indirect ref label base abcdef; abcdef+n indirect ref label base abcdef; abcdef-n indirect ref label base abcdef</td>
</tr>
<tr>
<td>label</td>
<td>a, b, c letter; d, e label symbol abcde label</td>
</tr>
<tr>
<td>labels among stm labels</td>
<td>a, b list stm labels; c stm label \label labels among stm labels a; \label labels among stm labels a; \label labels among stm labels a; c, \label labels among stm labels a, b</td>
</tr>
<tr>
<td></td>
<td>\label labels among stm labels b; \label labels among stm labels d;</td>
</tr>
<tr>
<td></td>
<td>\label labels among stm labels f \label labels among stm labels bdf</td>
</tr>
<tr>
<td>label symbol</td>
<td>d digit; i letter \label labels among stm labels \label symbol</td>
</tr>
<tr>
<td></td>
<td>[note, {*+- are not}]</td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>legal card program</td>
<td>$p$ identify card program; $p$ cards $\vdash p$ legal card program</td>
</tr>
<tr>
<td>legal info card program</td>
<td>$p$ any length card program; $p$ seventy-one column cards</td>
</tr>
<tr>
<td></td>
<td>$\vdash p$ legal info card program</td>
</tr>
<tr>
<td>link order arg</td>
<td>$n$ number $\vdash P_n$, LIFO, LIFO link order arg</td>
</tr>
<tr>
<td>list stm labels</td>
<td>$a$ stm label; $b$ list stm label $\vdash \Lambda_a$, list stm labels</td>
</tr>
<tr>
<td>loc field with stm label</td>
<td>$abc$ six sym or spaces; $b$ stm label; $a, c$ spaces void</td>
</tr>
<tr>
<td></td>
<td>$\vdash abc$ loc field with stm label $b$</td>
</tr>
<tr>
<td>logic mnemonic</td>
<td>$\vdash R, I, S$ logic mnemonic</td>
</tr>
<tr>
<td>logic mnemonic field</td>
<td>$abc$ six sym or spaces; $b$ logic mnemonic; $a, c$ spaces or void</td>
</tr>
<tr>
<td>number</td>
<td>$d$ digit; $n$ number $\vdash d, n$ number</td>
</tr>
<tr>
<td>number or void</td>
<td>$x$ number $\vdash \Lambda_x$, $x$ number or void</td>
</tr>
<tr>
<td>plain ref label base</td>
<td>$x$ label; $a, b, c$ letter; $d, e$ ref label sym; $n$ number $\vdash x$ plain</td>
</tr>
<tr>
<td></td>
<td>ref label base $x$; $abcd$ plain ref label base $abcde$;</td>
</tr>
<tr>
<td></td>
<td>$abcd+n$ plain ref label base $abcde$; $abcd-n$ plain ref label base $abcde$</td>
</tr>
<tr>
<td>print mnemonic arg</td>
<td>$\vdash , MOV, FUT, CHA, I, N, W, F, S, Q, T, X, A$</td>
</tr>
<tr>
<td>pref stand num attrib</td>
<td>$\vdash W, N, CH, P, MP, F, S, R, V, FN, Q, TB, K$</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
a stm labels in p with ref labels r; r labels among stm labels a \( \vdash \) p program

p comment program \( \vdash \) p program

b fnc def with ref labels c; d var def with ref labels e;
\( \vdots \) storage def with ref label g; h table def with ref labels i;
\( s \) stm labels in mn with ref labels r;
rc, e, g, i labels among stm labels s \( \vdash \) mbdfhn

program

x label; a plain ref label base b; c indirect ref label base d; a letter; b c d e f ref labels sym \( \vdash \) x ref label base x; a ref label base b; c ref label base d;
w$abcdef ref label base abcdef; N$abcdef ref label

base abcdef

x label symbol \( \vdash \) x \text{label} \text{symbol} / \text{ref label sym} \quad \text{[note, + - are not]}

seventy-one column cards

a_1 a_2 a_3 \ldots a_{71} \text{sym or space}; x \text{seventy-one column cards} \quad \text{[x seventy-one column cards]}

simulate card program

a_1 a_2 a_3 \ldots a_{65} \text{sym or space}; p \text{program}
\quad \vdash \#6\text{SIMULATE}a_1 a_2 \ldots a_{65} \# \text{p simulate card program}

six sym or spaces

a_1 b c d e f \text{sym or spaces} \quad \text{[abcdef six sym or spaces]}

spaces

x space \( \vdash \) x \text{1} \text{spaces}

spaces or void

x space \( \vdash \) x \text{1} \text{spaces or void}
\[ \text{stand num} \quad \text{pref} \quad \text{stand num} \quad \text{attrib} \quad \text{sna} \]
\[ \qquad \text{attrib} \quad \text{sna} \quad \text{stand num} \quad \text{attrib} \quad \text{sna} \]
\[ \text{stand num} \quad \text{attrib} \quad \text{sna} \quad \text{*} \quad \text{sna} \]

\[ \text{stm label} \quad \text{x} \quad \text{label} ; \quad \text{a} \quad \text{letter} ; \quad \text{bcde} \quad \text{sym} \quad \text{or} \quad \text{space} \quad \text{or} \quad \text{void} ; \quad \text{f} \quad \text{symbol} \]

\[ \Rightarrow \quad x \quad \text{*} \quad \text{abcdef} \quad \Lambda \quad \text{stm labels} \]

**Statements**

\[ \ldots \quad \text{stm labels in} \quad \text{with ref labels} \]
\[ \quad \Rightarrow \quad \Lambda \quad \text{stm labels in} \quad \Lambda \quad \text{with ref labels} \quad \Lambda \]

**first premises**

\[ \text{s} \quad \text{stm labels in} \quad \text{p} \quad \text{with ref labels} \quad \text{r} ; \quad \text{a} \quad \text{loc field with} \]
\[ \quad \text{label 1} ; \quad \text{-----} \]
\[ \text{s1}, \quad \text{stm labels in} \quad \text{paBUFFER}\text{5#} \quad \text{with ref labels} \quad \text{r} \]
\[ \text{s1}, \quad \text{stm labels in} \quad \text{paTRACE}\text{4#} \quad \text{with ref labels} \quad \text{r} \]
\[ \text{s1}, \quad \text{stm labels in} \quad \text{paUNTRACE}\text{4#} \quad \text{with ref labels} \quad \text{r} \]

\[ \text{t ibsys tape} \quad \text{paWRITE}\text{5#} \quad \text{with ref labels} \quad \text{r} \]

**second premises**

\[ \text{b} \quad \text{arg sna* sna or ref label} \quad \text{c} ; \quad \text{a} \quad \text{number} ; \quad \text{y} \quad \text{arg} \]

\[ \text{void sna* sna or ref label} \quad \text{z} ; \quad \text{-----} \]
\[ \text{s1}, \quad \text{stm labels in} \quad \text{paASSEMBLE}\text{5b#} \quad \text{with ref} \]
\[ \quad \text{labels} \quad \text{rc} , \]
\[ \text{s1}, \quad \text{stm labels in} \quad \text{paEXECUTE}\text{4b#} \quad \text{with ref} \]
\[ \quad \text{labels} \quad \text{rc} , \]
\[ \text{s1}, \quad \text{stm labels in} \quad \text{paGATHER}\text{5b#} \quad \text{with ref labels} \quad \text{rc} , \]

---

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
t logic mnemonic field \( \lnot \text{si, stm labels in paLOGICtb# with ref labels rc}, \)
\( \lnot \text{si, stm labels in paMARKtb# with ref labels rc}, \)
\( \lnot \text{si, stm labels in paMATCH6b# with ref labels rc}, \)
\( \lnot \text{si, stm labels in paPREEMPT4b# with ref labels rc}, \)
\( \lnot \text{si, stm labels in paPRIORITY3y# with ref labels rz}, \)
\( \lnot \text{si, stm labels in paPRIORITY3y, BUFFER# with ref labels rz}, \)
\( \lnot \text{si, stm labels in paRELEASE4b# with ref labels rc}, \)
\( \lnot \text{si, stm labels in paRETURN5b# with ref labels rc}, \)
\( \lnot \text{si, stm labels in paSEIZE6b# with ref labels rc}, \)
\( \lnot \text{si, stm labels in paTERMINATE2y# with ref labels rz}, \)

third premises \( \text{d arg sna* sna or ref label e; f arg void, sna* sna or ref label g} \)
\( \lnot \text{si, stm labels in paCHANGE5b, d# with ref labels rc, e}, \)
\( \lnot \text{si, stm labels in paDEPART5bf# with ref labels rc, g}, \)
\( \lnot \text{si, stm labels in paENTER6bf# with ref labels rc}, \)

t gate mnemonic field \( \lnot \text{si, stm labels in paGATE1tbf# with ref labels rc, g}, \)
\( \lnot \text{si, stm labels in paLEAVE6bf# with ref labels rc, g}, \)
t link order arg | \( s_1 \), stm labels in \( \text{pa}\text{LINK}^7\text{btf}\# \) with ref labels \( rc, g \),
| \( s_1 \), stm labels in \( \text{pa}\text{LOOP}^7\text{b}, d \# \) with ref
| labels \( rc, e \),
| \( s_1 \), stm labels in \( \text{pa}\text{QUEUE}^6\text{bf}\# \) with ref labels \( rc, g \),
| \( s_1 \), stm labels in \( \text{pa}\text{TABULATE}^3\text{bf}\# \) with ref
| labels \( rc, g \),

fourth premises h arg void, sna* sna or ref label i; j arg num or
| ref label k; m arg void number ref label o; q arg void, num or
| ref label u; v arg void num or ref label w; ---
| \( s_1 \), stm labels in \( \text{pa}\text{ADVANCE}^4\text{yq}\# \) with ref
| labels \( rz, u \),
| \( s_1 \), stm labels in \( \text{pa}\text{ADVANCE}^4\text{yn}\# \) with ref
| labels \( rz \),
| \( s_1 \), stm labels in \( \text{pa}\text{ADVANCE}^4\text{yn}, \text{FNn}\# \) with ref
| labels \( rz \),
| \( s_1 \), stm labels in \( \text{pa}\text{ADVANCE}^4\text{yn}, \text{FN*n}\# \) with ref
| labels \( rz \),
| \( s_1 \), stm labels in \( \text{pa}\text{INDEX}^6\text{jq}\# \) with ref labels \( rk, u \),
| \( s \), stm labels in \( \text{pa}\text{INITIAL}^4\text{Xnq}\# \) with ref labels \( ru \),

number void \( s \), stm labels in \( \text{pa}\text{INITIAL}^4\text{Xn-t}\# \) with ref
| labels \( r \)

print mnemonic arg \( s_1 \), stm labels in \( \text{pa}\text{PRINT}^6\text{v}, \text{mt}\# \) with ref labels \( rw, o \),
| \( s_1 \), stm labels in \( \text{pa}\text{SPLIT}^6\text{y}, d \# \) with ref labels \( rz, e \),
\[ t \text{test mnemonic field} \quad \vdash \text{stm labels in paTEST}\_{tyfh} \text{# with ref labels rz, g, i,} \]

\[ t \text{transfer mnemonic arg} \quad \vdash \text{stm labels in paTRANSFER}\_{tfhq}\text{# with ref labels rg, e, u} \]

\[ t \text{xact unlink limit arg sna or ref label f}_1 \text{; f}_2 \text{xact unlinked arg sna or ref label} \quad \vdash \text{stm labels in paUNLINK}\_{fb}\text{, d, tf}_2\text{fh# with ref labels rc, e, f}_1, f_3, g, i, \]

\[ \text{fifth premises a}_1 \text{ arg sna* sna or ref label no offset a}_2; \]

\[ \vdash \text{stm labels in paASSIGN}\_{a}_1\text{ fh# with ref labels ra}_2, g, i, \]

\[ \vdash \text{stm labels in paASSIGN}\_{a}_1+yh\text{# with ref labels ra}_2, z, i \]

\[ \vdash \text{stm labels in paASSIGN}\_{a}_1-yh\text{# with ref labels ra}_2, z, i, \]

\[ \vdash \text{stm labels in paSAVEVALUE}\_{a}_1\text{ fh# with ref labels ra}_2, g, \]

\[ \vdash \text{stm labels in paSAVEVALUE}\_{a}_1+yh\text{# with ref labels ra}_2, z, \]

\[ \vdash \text{stm labels in paSAVEVALUE}\_{a}_1-yh\text{# with ref labels ra}_2, z, \]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ b_1 \text{arg} \text{void, sna* sna or ref label } b_2; b_3 \text{arg} \text{void, sna* sna or ref label } b_4 \]
\[ \vdash s_1, \text{stm labels in } \text{paHELP}_1,yfb_1b_3# \text{ with ref labels } rz, g, i, b_2, b_4, \]
\[ c_1 \text{arg} \text{void sna or ref label } c_2; c_3 \text{arg} \text{void sna or ref label } c_4; c_5 \text{arg} \text{void, num or ref label } c_6; c_7 \text{arg} \text{void, num or ref label } c_8; c_9 \text{arg} \text{void, num or ref label } c_{10}; c_{11} \text{arg} \text{void, sna or ref label } c_{12} \]
\[ \vdash s_1, \text{stm labels in } \text{paGENERATE}_3 c_1# \text{ with ref labels } rc_2, \]
\[ \vdash s_1, \text{stm labels in } \text{paGENERATE}_3 c_1, c_3 c_5 c_7 c_9# \]
\[ \text{with ref labels } rc_2, c_4, c_6, c_8, c_{10}, \]
\[ d_1 \text{arg} \text{void num or ref label } d_2; d_3 \text{arg} \text{void num or ref label } d_4; d_5 \text{arg} \text{void num or ref label } d_6 \vdash s_1, \text{stm labels in } \text{paGENERATE}_3 c_1, c_3, d_1, \]
\[ d_3 d_5 c_{11}# \text{ with ref labels } rc_2, c_4, d_2, d_4, d_6, c_{12}, \]
\[ \text{end premises} \]
Control Cards

alpha numeric x letter; y digit; z alpha numeric | A zy, zx alpha numeric

first premises a field with stm label b; s stm labels in p with ref labels r; ---

| s stm labels in paRESET6# with ref labels r
| s stm labels in pa$6JOB8# with ref labels r
| s stm labels in paEND8# with ref labels r
| s stm labels in paLIST7# with ref labels r
| s stm labels in paUNLIST5# with ref labels r
| s stm labels in paCLEAR6# with ref labels r

second premises c arg num or ref label d; x alpha numeric; ---

| s stm labels in paREAD7c# with ref labels rd,
t ibsys tape | s st m labels in paREWIND5t# with ref labels r
| s stm labels in paSAVE7x# with ref labels r

third premises y arg void num or ref label z; e arg void, num or ref label f; g arg void, num or ref label h;i arg void, num or ref label j; ---

| s stm labels in paJOBTAPE4t, cdg# with ref labels rd, f, h,

| s stm labels in paSTART6yegi# with ref labels rz, f, hj,
| s stm labels in paSTART6y, NP, eg# with ref labels ry, f, h,

end premises
Pseudo-Operations

---

**first premises**

a field with stm label 1; b ref label; b labels among

stm labels sb, s stm labels in p with ref labels r; s spaces

void; bs, ns, ts six sym or spaces; ——

s stm labels in paORG8bs# with ref labels r

n number or void - s stm labels in paICT8ns# with ref labels r

t stm label s1, stm labels in paSYN8bs# with ref labels r

s stm labels in paABS8# with ref labels r

s stm labels in paENDABS5# with ref labels r

---

**end premises**

---

**storage def**

a field with num b; c arg num or ref label d | A storage
def with ref label A; aSTORAGE4c storage def with
ref label d

symbol

| 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z = i, j, k, ...()

- / symbol

symbols

x symbol; y symbols | x, y symbols

string

x sym or space; y string | A, y string

sym or space

x symbol | x, y sym or space

sym or space or void

x sym or space | A, x sym or space or void

---

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
table arg  a stand num attrib sna | a_{1} a_{-1} RT_{1} IA table arg

table def with ref labels  a field with num b; c table arg; n number; d arg void, num

or ref label e; f arg void, num or ref label g; h arg

void, num or ref label i; j arg void, num or ref

label k \mid A  table def with ref labels A ;

a\_TABLE_{\text{g}} cdfhj\#  table def with ref labels e, g, i, k ;

a\_Q\_TABLE_{\text{g}} ndfhj\#  table def with ref labels e, g, i, k ,

test mnemonic  \mid L_{1} LE_{1} G_{1} GE_{1} E_{1} NE test mnemonic

test mnemonic field  abc six sym or spaces; b test mnemonic; a, c spaces or

void \mid abc test mnemonic field

transfer mnemonic arg  n number \mid \cdot n_{1} FN_{1} Pn_{1} BOTH_{1} PICK_{1} SBR_{1} SIM_{1} ALL_{1} A

transfer mnemonic arg

var arg field with ref labels  a var arg sna or ref label b; c variable op; d var arg

field with ref labels e \mid a var arg field with ref

labels b, ; dca var arg field with ref labels eb ,

var arg sna or ref label  a stand num attrib sna; l var arg sna or ref label l ;

N$\text{defgh} var arg sna or ref label defgh; W$\text{defgh}

var arg sna or ref label defgh; $\text{defgh} var arg sna

or ref label defgh; l var arg sna or ref label l

var def with ref labels  a var arg field with ref labels b; c field with num d

\mid CVARIABLE_{3a}# var def with ref labels b; A var

def with ref labels

var labels sym  x label symbol \mid x_{1} , var label sym  [note (+ - */ are not]

variable op  - +_{1} -^{*}_{1} /_{1} ( variable op

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
APPENDIX B

EXPERIMENTAL VERIFICATION OF

SYNTAX OF GPSS III

B.1 Summary

This appendix is concerned with that portion of the syntax of GPSS III that is ambiguously or incorrectly defined in the GPSS III manuals.

Many of the constructions defined by Appendix A were found to be legal formations by experiment. That is, test programs were written in GPSS III and executed on an IBM 7094/7040 DCS. For each construction that was executed, enough variations were written to find a pattern.

The major conclusion of this experimentation was that the syntax of GPSS III is in general less restrictive than is stated in the manuals. Statement labels, reference labels, arguments, and column format were found to have more flexibility in their form. A GPSS programmer might be surprised to learn that the statement

$$Z*( )$= TEST E $Z*( )$=, $Z*( )$=+50, 4$$

is a legal statement.

It has not been made clear in the manuals for all statements which arguments in these statements must not be void and which may either be void or not void. Each argument was experimentally tested to determine if it may be void. Examples of such statements are given later in this appendix.

In a few instances it was found that the manuals were not restrictive
enough. Some statements (e.g., blank cards or any character in
column one are permissible, and an ORIGINATE block is permissible)
which the manuals state to be legal constructions are not, and will cause
a program containing these constructions to be rejected.

Some of the results of these experiments have been discussed in
Section III. 12 through III. 14.

In Section B.2.a of this appendix is presented a listing of a program
that passed through all phases of GPSS III preprocessing and was executed.
This program contains examples of constructions that are legal but were
stated or implied to be illegal in the manuals. There has been no attempt
to include all examples of legal constructions. For examples of more
conventional constructions consult Appendix D.

In Section B.2.b is presented the symbol table of the above-mentioned
program. The reader may use this to locate the block numbers that
were assigned to each statement label.

In Section B.2.c are presented the assembled statements of the
above-mentioned program. In the assembled program all symbols
(labels) of each statement have been given their assigned block numbers.
The reader should use this assembled program to learn how the labels,
numbers, and strings were recognized by GPSS III. For example the
reference label ABC+1 is recognized as two different labels: as the
statement label ABC+1, and as the relative addressed label ABC plus
one. Very large numbers may be permissible but they are not always
recognized as the numbers they denote. In Section B.2.c it is demonstrated the recognition of the large numbers in the above-mentioned program by GPSS.

The execution of the program will cause the contents of "savevalue" locations 1, 2, 3, 10 and 15 to be printed if they contain any number other than a zero. Locations 1, 2, 3 contain the numbers recognized by GPSS from a void B field in an INDEX, ASSIGN, and SAVEVALUE block (statement). Locations 10 and 15 contain the numbers recognized by GPSS from very large numbers used in the INITIAL blocks in the above-mentioned program.

It is important that a programmer study the assembled program thoroughly before using void arguments or very large numbers or complex reference labels. A void argument may denote different values to GPSS III. E.g. void arguments in an ADVANCE block denote zero, void arguments in a TRANSFER block denote next block number, void arguments in A field of a GENERATE or SPLIT block denote one. Reference labels may denote different numbers depending on their context. Large numbers may denote different numbers. E.g. the number 50044444444555566666222 used as an A field of a TRANSFER block is recognized as 222.

A listing of a program that has a syntax error in nearly every statement is presented in Section B.2.d. In this program there are examples of constructions that have been stated or implied in the manuals to be legal but are not. In this program there are also examples of
constructions that from the results of the experimental programs might be thought to be legal but are not.

B.2 Experimental Verification of Ambiguously or Incorrectly Stated Constructions in the GPSS Manuals
<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>LOC</th>
<th>NAME</th>
<th>A, B, C, D, E</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**SIMULATE TEST SYNTAX**

*EVERY STATEMENT BELOW IS SYNTACTICALLY LEGAL*

THIS PROGRAM WAS COMPILED AND EXECUTED ON AN IBM 7094/7040 DCS BY GPSS III

---

**PERMISSIBLE STATEMENTS·LABELS**

IT IS STATED IN THE GPSSIII MANUALS (PP. 19 AND 224 OF REF NO. 9) THAT A BLOCK SYMBOL (STM LABEL) MUST CONSIST OF FROM THREE TO FIVE ALPHAMERIC, NON BLANK CHARACTERS, THE FIRST THREE OF WHICH MUST BE LETTERS. AN ADDITIONAL RESTRICTION IS THAT NONE USED IN A BLOCK SYMBOL. \(+ - \$/(-,\) IN ADDITION, IT IS STATED IN THE MANUAL (P. 225, REF NO. 9) THAT DOLLAR SIGNS MAY NEVER OCCUR IN THE LOCATION FIELD. IN THE MANUALS IT IS STATED THAT LABELS MUST APPEAR IN COLUMNS 2 THROUGH 7 (E.G. P. 224 OF REF 9) NOTE THE VARIETY OF PERMISSIBLE STM LABELS FOUND BY EXPERIMENT BELOW. FOR PRECISE DEFINITION OF PERMISSIBLE STM LABELS CONSULT APPENDIX A.

---

1. \(Z*()\$= \) SEIZE 1 STM LABEL IS \(Z*()\$= \) AND IS REFERENCED BY ENTER STM BELOW

2. \(X=,+/ \) SEIZE 1

3. \(Q^*,+- \) SEIZE 1

4. \(ABC)\) SEIZE 1

5. \(ABC$\, \) SEIZE 1

6. \(ABC(1 \) SEIZE 2 LABELS START IN COLUMN NUMBER 1

7. \(ABC1 \) SEIZE 2

8. \(ABC*1 \) SEIZE 2

9. \(ABC\) SEIZE 2

10. \(ABC(1 \) SEIZE 2

11. \(ABC\) SEIZE 2

12. \(ABC\) SEIZE 2

13. \(ABC\) SEIZE 2

14. \(ABC\) SEIZE 2

15. \(ABC\) SEIZE 2

16. \(ABC\) SEIZE 2

17. \(ABC\) SEIZE 2

18. \(ABC\) SEIZE 2

19. \(ABC\) SEIZE 2

20. \(ABC\) SEIZE 2

21. \(ABC\) SEIZE 2

22. \(ABC\) SEIZE 2

23. \(ABC\) SEIZE 2

24. \(ABC\) SEIZE 2

25. \(ABC\) SEIZE 2

---

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
A1111 SEIZE 1
A*+ SEIZE 1
ABC/1 SEIZE 5
ABC- SEIZE 5
ABC+1 SEIZE 5
A SEIZE 1
AA SEIZE 2
A=BC SEIZE 1
AABC SEIZE 1
ABC+5 SEIZE 1
A,,, A$ SEIZE 4
ABC$ SEIZE 4
ABC$ SEIZE 1
A12 SEIZE 1
ABCDE SEIZE 1
A PP SEIZE 2
HELP SEIZE 10
A*() SEIZE 1
ABC12 INITIAL X10,10
* NOTE STM LABEL IS PERMISSIBLE IN INITIAL CARD
* BUT IT IS NEVER ADDED TO THE LIST OF SYMBOLS
* THEREFORE IT MAY NEVER BE REFERENCED
TERMINATE
*
* --------------- PERMISSIBLE FUNCTION DATA CARD ARGUMENTS
* FOR PRECISE DEFINITION OF ALL PERMISSIBLE ARGU-
* MENTS OF FUNCTION DATA CARDS CONSULT APPENDIX A.
* NOTE THE READER SHOULD CONSULT SECTION B.2.B AND
* B.2.C , THE SYM TABLE AND THE ASSEMBLED PROGRAM,
* TO SEE HOW GPSS II RECOGNIZED THESE REFERENCE LABELS
*
1 FUNCTION V10,D2
ABC+1 ABC=*(ABC$* ABC(*)
* LABELS REFERENCED IN THE ABOVE FNC DATA CARD ARE
* ABC+1 ABC=*(ABC$* ABC(*), ABC+1 IS NOT A REL-
* ATIVE LABEL.
2 FUNCTION $ABC+1,D2 ABC IS A RELATIVE LABEL
ABC+1 ABCD+ ABCDE ABC(*)
55 FUNCTION V1,D2
ABC=*. 12 50 90
6 FUNCTION V1,E2
ABCDEF 12 ABC123 19
7 FUNCTION V1,E2
ABC=*(12 ABC(*) 50
88 FUNCTION P1,C2
ABC 12 120 123 400
* NOTE THE LABEL ABC 12 IS REFERENCED IN THE ABOVE
89 FUNCTION S10,D2
ABC=*(12 ABC$. 14
8 FUNCTION V1,E2
ABC$, ABC(1 ABC1 ABC()
66 FUNCTION X10,D2
ABC- 30 40 50
15 FUNCTION V12,E2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Q10  ABC  ABC$*  ABC==
10  FUNCTION  V10,E2
ABC  ABCD+  A=BC  ABCD

*------------PERMISSIBLE VARIABLE ARGUMENTS
*
* NOTE THE READER SHOULD CONSULT SECTION B.2.B AND
* B.2.C, THE SYM TABLE AND THE ASSEMBLED PROGRAM,
* TO SEE HOW GPSSIII RECOGNIZED THESE REFERENCE LABELS
*
1  VARIABLE  ABC/1*$ABC/1+1-3  REFERENCE LABEL IS ABC
2  VARIABLE  2*$A12
3  VARIABLE  ABC*Q10  LABEL IN VARIABLE
4  VARIABLE  2*$ABC==
6  VARIABLE  $AB+1  NOTE REF LABEL IS AB AND NOT AB+1
5  VARIABLE  $A,+,+1  LABEL REFERENCED IS A,+,+
13  VARIABLE  5+$ABC$+3  LABEL REFERENCED IS ABC
14  VARIABLE  $ABC/12
12  VARIABLE  ABC=,*+1
22  VARIABLE  ABC$,+1
23  VARIABLE  ABC==++,2
33  VARIABLE  1+$A111+$ABC==
43  VARIABLE  1+$A1111
53  VARIABLE  ABC1.++1

*------------PERMISSIBLE ARGUMENTS
*
* STRICT DEFINITION OF PERMISSIBLE ARGUMENTS IS IN
* APPENDIX A. NOTE VARIETY OF PERMISSIBLE REF
* LABELS IN ARGUMENTS OF STATEMENTS
*
45 ENTER  $Z$,,$+21
*
* SEE SECTION B.2.B AND SECTION B.2.C, SYM TABLE
* AND ASSEMBLE PROGRAM FOR GPSSIII RECOGNITION OF ABOVE
46 A12345 ENTER  $A12345
47 ABC133 PREEMPT  $ABC133  REF LABEL IS 6 CHARACTERS
48 ABC13* EXECUTE  $ABC12*
49 EXECUTE  $ABCDDEF
50 EXECUTE  $ABC12**,1
51 ALL RELEASE  ABC
52 TRANSFER  ,ABC*1-1
53 TRANSFER  ,ABCDDEF  LABEL MORE THAN 6 CHARACTERS
54 TRANSFER  ,ABC*1
55 TRANSFER  ,ALL+9
56 TRANSFER  ,ALL+10
57 TRANSFER  ,ALL+20
58 TRANSFER  BOTH, ALL+25, ALL
59 TRANSFER  ,ALL+25
60 TRANSFER  ALL, ALL+25, ALL, 5
*
* THE FIRST ALL IN ARG ABOVE IS NOT A LABEL
61 ASS111 TRANSFER  ALL,ABC+,25,ABC,0
62 TRANSFER  ALL,ABC,ABC+,25,5
63 TRANSFER  ALL,ALL,ALL+25,5
64 TRANSFER  ALL,ALL,ABC,1
65 TRANSFER  ,ALL+1
66 TRANSFER  ALL,$ALL+1,ALL,1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
67 Transfer All,$ABC,All,1
68 Transfer All+8
69 Transfer All,ABC+1,ABC+2,1
70 Transfer ,ABC12*
71 Transfer ,All+5
72 Transfer ,All+4
73 Transfer ,All+3
74 Transfer ,All-8
75 Transfer ,AllLL+1
76 Allll Transfer ,$ALLll+1
77 Transfer ,$ABC12*
78 Transfer All,All,All,5
79 Transfer BOTH,All,All+3
80 Transfer ,TRAN TRANSFER TO SAME STATEMENT
81 Transfer ,$HELP TEST $ IN FRONT OF LABEL
82 Transfer ,ABC*1
83 Transfer ,$ABC+1
84 Transfer ,ABC
85 Transfer ,ABC+1
86 Transfer ,ABC+1 SEE SEC. B.2.C FOR RECOGNITION
87 SaveValue $ABC,-2
88 SaveValue ABC+1
89 SaveValue $ABC+1
90 SaveValue $ABC+1
91 Assign N$ABC+ LABEL IS ABC
92 Assign ABC+,1
93 Assign $ABC+,1
94 Assign $ALL,1 SYMBOLOIC BLOCK
95 Assign P10,1
96 Assign $ABC,1
97 Assign 1+,1
98 Assign ABC+ P10+,1
99 Print ABC,ABC-1, NOTE COMMA IS THERE
100 Xs*1 Assemble ABC
101 All Seize $ALL $ALL IS RECOGNIZED AS A
102 Seize ABC+1 REF LABELS
103 ABD+ Seize 2
104 ABC Seize ABC
105 ABC Seize ABC=
106 ABC$ Seize ABC$*
107 ABC Seize ABC.1
108 ABC/ Seize ABC/
109 A123 Seize $A123 REFERENCE LABEL
110 ABC-1 Seize $ABC+1
111 ABCD Seize All
112 Seize ABCD
113 Seize ABC+1-1
114 Seize $A
115 Seize ALL+25
116 Seize ALL+9
117 Seize $ABCDEF REF LABEL IS 6 CHARACTERS
118 Seize ABC+1
119 Seize $A=BC $ IN REF LABEL
120 Queue $ABCDE
121  QUEUE $ABC+1
122  QUEUE W$ABCDE
123  QUEUE N$ABCDE
124  RELEASE $ABC

*--------- PERMISSIBLE VOID ARGUMENTS
* NOTE A VOID ARG MAY BE RECOGNIZED AS DENOTING
* SEVERAL DIFFERENT VALUES. CONSULT SECTION B.2.C
*
125  ADVANCE 1, WHERE CAN VOID ARGUMENTS BE
126  ASSIGN 4, WHERE CAN VOID ARGUMENTS BE
127  DEPART 4, WHERE CAN VOID ARGUMENTS BE
128  GATE LS 4, WHERE CAN VOID ARGUMENTS BE
129  PRINT 0, MOV PRINT FUTURE EVENTS CHAIN
130  PRINT P1, FUT PRINT FUTURE EVENTS CHAIN
131  PRIORITY WHERE CAN VOID ARGUMENTS BE
132  SAVEVALUE 1, WHERE CAN VOID ARGUMENTS BE
133  TEST GE WHERE CAN VOID ARGUMENTS BE
134  TEST NE WHERE CAN VOID ARGUMENTS BE
135  TRANSFER WHERE CAN VOID ARGUMENTS BE
136  TRANSFER WHERE CAN VOID ARGUMENTS BE

* IMPORTANT - A VOID ARGUMENT MAY DENOTE DIFFERENT
* VALUES TO GPSS III E.G. VOID ARG IN ADVANCE STM
* DENOTES ZERO, VOID ARG IN TRANSFER STM DENOTES ONE
* TEST IF BLANKS EXECUTE CORRECTLY
* GENERATE 1,1,1,1,1 THIS WILL CAUSE THE
* THE RECOGNITION OF VOID ARG IN INDEX AND ASSIGN STM
* TO BE PRINTED. SEE END OF SEC B.2.C
*
137  ADVANCE 1, BLANK TIME
138  ASSIGN 1,5 P1=5
139  INDEX 1, IS BLANK PERMISSIBLE
140  SAVEVALUE 1,P1
141  PRINT 1,1 PRINT VALUE OF SAVE 1 OR P1
142  ASSIGN 2, BLANK IN ASSIGN
143  SAVEVALUE 2,P2
144  PRINT 2,2 PRINT P2
145  SAVEVALUE 3,
146  PRINT 3,3 BLANK IN SAVEVALUE
147  TERMINATE 1

*--------- PERMISSIBLE COLUMN FORMAT
*--------- PERMISSIBLE COLUMN FORMAT FOR LOGIC ATTRIB
* OR RELATIONAL MNEMONICS. IT IS STATED IN THE
* MANUALS (E.G. P.224, OF REF NO.9) THAT WHEN A LOGIC
* ATTRIB OR RELATIONAL MNEMONIC IS NEEDED IN THE
* AUX TEST, GATE, AND LOGIC BLOCKS IT MUST BEGIN IN
* COLUMN 13 FOR GATE AND TEST BLOCKS AND IN
* COLUMN 14 FOR LOGIC BLOCKS.

149  LOGICR 1 TEST IF MNEMONIC IN COL 13
150  LOGIC R 1 TEST IF MNEMONIC IN COL 14
151  LOGIC R 1
152  LOGIC R 1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
153 LOGIC R1
154 TEST GE 12,12,1 TEST WHERE GE GOES
155 TEST GE 12,12,1 TEST WHERE GE GOES
156 TEST GE 12,12,1 TEST WHERE GE GOES
157 VARIABLE VI TEST RIGHT JUSTIFICATION
158 STORAGE 20 TEST RIGHT JUST OF STORAGE

* TEST HOW LONG NUMBER CAN BE
* NOTE THE FORM OF NUMBERS PERMISSIBLE IN THE
* FOLLOWING STM, THESE NUM ARE NOT ALWAYS RECOGNIZED
* BY GPSS III AS THE NUM THEY DENOTE

159 TRANSFER 1111111111111111111111111, ALL, ABCDE
160 TRANSFER 50044444444445555566666222, TES1, TES2

* SEE SEC. B.2.C FOR RECOGNITION OF ABOVE NUMBERS
161 INITIAL X10,1233333333333333333333333
162 INITIAL X15,12345678912345678 LIMIT ON LENGTH OF N
163 GENERATE 1,1,1,1,1 THIS WILL CAUSE THE
164 GENERATION OF THESE LONG NUMBERS IN INITIAL STM

165 PRINT 15,15 PRINT NUMBER IN INITIAL ABOVE
166 PRINT 10,10 PRINT SAV NO 10
167 TERMINATE 1

*------------------ PERMISSIBLE ARGUMENTS FOR GENERATE STATEMENT
* GENERATE WHERE CAN VOID ARGUMENTS BE

168 TERMINATE
169 GENERATE 2,1,,10
170 TERMINATE
171 GENERATE ,,,13
172 TERMINATE
173 GENERATE 10,2
174 TERMINATE
175 GENERATE 8,2,,10
176 TERMINATE
177 GENERATE 12,3
178 TERMINATE
179 GENERATE V1,ABC
180 TERMINATE
181 GENERATE 10,3,ABC,ABC

*----------------- PSEUDO OPERATIONS
* IT IS STATED IN THE MANUALS THAT LOCATION FIELD
* MUST BE LEFT BLANK OF PSEUDO OPS ICT, ABS, ENDABS
* THIS STM IS NOT CORRECT. HOWEVER IF A STM LABEL
* IS USED IN THE LOC FIELD IT IS NOT ADDED TO THE
* LIST OF SYM E.G. ACFV IS NOT IN LIST OF SYM SEC-
* TION B.2.B. HENCE MAY NOT BE REFERENCED

182 GGGG SEIZE 1
183 ORG ABCDE PSUEDO OP NEXT BLOCK = ABCDE
184 SSS SEIZE 6
MULTIPLE DEFINITION OF SYMBOL IN ABOVE CARD

53
500 SEIZE 46
499 499 = 4ASS

6
6DSS SEIZE

6

5
5ICT SEIZE

6
6AX SEIZE

NOTE BLOCK NUMBER OF ABOVE BLOCK
b. Symbol Table of Legal Program

<table>
<thead>
<tr>
<th>*</th>
<th>SYMBOLS AND CORRESPONDING BLOCK NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>*</td>
</tr>
<tr>
<td>20</td>
<td>A111</td>
</tr>
<tr>
<td>26</td>
<td>A1111</td>
</tr>
<tr>
<td>39</td>
<td>A12</td>
</tr>
<tr>
<td>109</td>
<td>A123</td>
</tr>
<tr>
<td>46</td>
<td>A12345</td>
</tr>
<tr>
<td>33</td>
<td>A=BC</td>
</tr>
<tr>
<td>32</td>
<td>AA</td>
</tr>
<tr>
<td>34</td>
<td>AABC</td>
</tr>
<tr>
<td>17</td>
<td>A3</td>
</tr>
<tr>
<td>19</td>
<td>AB123</td>
</tr>
<tr>
<td>24</td>
<td>ABC</td>
</tr>
<tr>
<td>13</td>
<td>ABC123</td>
</tr>
<tr>
<td>48</td>
<td>ABC12*</td>
</tr>
<tr>
<td>47</td>
<td>ABC133</td>
</tr>
<tr>
<td>105</td>
<td>ABC=</td>
</tr>
<tr>
<td>25</td>
<td>ABC==</td>
</tr>
<tr>
<td>11</td>
<td>ABC= '</td>
</tr>
<tr>
<td>14</td>
<td>ABC='(</td>
</tr>
<tr>
<td>9</td>
<td>ABC='</td>
</tr>
<tr>
<td>104</td>
<td>ABC='1</td>
</tr>
<tr>
<td>8</td>
<td>ABC='1</td>
</tr>
<tr>
<td>98</td>
<td>ABC+</td>
</tr>
<tr>
<td>30</td>
<td>ABC+1</td>
</tr>
<tr>
<td>35</td>
<td>ABC+5</td>
</tr>
<tr>
<td>111</td>
<td>ABCD</td>
</tr>
<tr>
<td>103</td>
<td>ABCD+</td>
</tr>
<tr>
<td>40</td>
<td>ABCDE</td>
</tr>
<tr>
<td>12</td>
<td>ABC=DEF</td>
</tr>
<tr>
<td>107</td>
<td>ABC.()</td>
</tr>
<tr>
<td>7</td>
<td>ABC.1</td>
</tr>
<tr>
<td>4</td>
<td>ABC.</td>
</tr>
<tr>
<td>29</td>
<td>ABC−</td>
</tr>
<tr>
<td>110</td>
<td>ABC−1</td>
</tr>
<tr>
<td>16</td>
<td>ABC$</td>
</tr>
<tr>
<td>38</td>
<td>ABC$</td>
</tr>
<tr>
<td>37</td>
<td>ABC$</td>
</tr>
<tr>
<td>106</td>
<td>ABC$</td>
</tr>
<tr>
<td>5</td>
<td>ABC$</td>
</tr>
<tr>
<td>23</td>
<td>ABC*$1</td>
</tr>
<tr>
<td>18</td>
<td>ABC 12</td>
</tr>
<tr>
<td>28</td>
<td>ABC/1</td>
</tr>
<tr>
<td>108</td>
<td>ABC//</td>
</tr>
<tr>
<td>6</td>
<td>ABC(1</td>
</tr>
<tr>
<td>10</td>
<td>ABC()</td>
</tr>
<tr>
<td>15</td>
<td>ABC(*)</td>
</tr>
<tr>
<td>51</td>
<td>ALL</td>
</tr>
<tr>
<td>76</td>
<td>ALLLLL</td>
</tr>
<tr>
<td>27</td>
<td>A*:*</td>
</tr>
<tr>
<td>43</td>
<td>A*()</td>
</tr>
<tr>
<td>101</td>
<td>A LL</td>
</tr>
<tr>
<td>41</td>
<td>A PP</td>
</tr>
<tr>
<td>46</td>
<td>ASSS</td>
</tr>
<tr>
<td>Line</td>
<td>Label</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>61</td>
<td>ASS</td>
</tr>
<tr>
<td>41</td>
<td>AZX</td>
</tr>
<tr>
<td>36</td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>DDS</td>
</tr>
<tr>
<td>179</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>HELP</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>SSS</td>
</tr>
<tr>
<td>21</td>
<td>TES1</td>
</tr>
<tr>
<td>22</td>
<td>TES2</td>
</tr>
<tr>
<td>80</td>
<td>TRAN</td>
</tr>
</tbody>
</table>
| 2    |       | X=S,+/
| 100  |       | X$*(   |
| 1    |       | Z*( )$= |
| 180  | ZZ    |       |
c. Assembled Legal Program

* SIMULATE TEST SYNTAX
* ------------------------------ LEGAL PROGRAM-------------------------------
* EVERY STATEMENT BELOW IS SYNTACTICALLY LEGAL
* THIS PROGRAM WAS COMPILED AND EXECUTED ON AN IBM 7094/7040 DCS BY GPSS III
*
*---------------------- PERMISSIBLE STATEMENTS LABELS
* IT IS STATED IN THE GPSSIII MANUALS (PP. 19 AND 224 OF REF NO. 9) THAT A BLOCK SYMBOL (STM LABEL) MUST
* CONSIST OF THREE TO FIVE ALPHAMERIC, NON BLANK
* CHARACTERS, THE FIRST THREE OF WHICH MUST BE
* LETTERS. AN ADDITIONAL RESTRICTION IS THAT NONE
* USED IN A BLOCK SYMBOL $/\{=, \} \quad \text{IN}
* ADDITION, IT IS STATED IN THE MANUAL (P. 225, REF NO. 9) THAT DOLLAR SIGNS MAY NEVER OCCUR IN
* THE LOCATION FIELD. IN THE MANUALS IT IS
* STATED THAT LABELS MUST APPEAR IN COLUMNS 2
* THROUGH 7 (E.G. P. 224 OF REF 9)
* NOTE THE VARIETY OF PERMISSIBLE STM LABELS FOUND
* BY EXPERIMENT BELOW. FOR PRECISE DEFINITION OF
* PERMISSIBLE STM LABELS CONSULT APPENDIX A.
*
* 1 \textbf{SEIZE 1} \quad \textbf{REFERENCED BY ENTER STM BELOW}
*
* 2 \textbf{SEIZE 1}
* 3 \textbf{SEIZE 1}
* 4 \textbf{SEIZE 1}
* 5 \textbf{SEIZE 1}
* 6 \textbf{SEIZE 2}
* 7 \textbf{SEIZE 2}
* 8 \textbf{SEIZE 2}
* 9 \textbf{SEIZE 2}
* 10 \textbf{SEIZE 2}
* 11 \textbf{SEIZE 2}
* 12 \textbf{SEIZE 2}
* 13 \textbf{SEIZE 2}
* 14 \textbf{SEIZE 2}
* 15 \textbf{SEIZE 2}
* 16 \textbf{SEIZE 2}
* 17 \textbf{SEIZE 2}
* 18 \textbf{SEIZE 2}
* 19 \textbf{SEIZE 1} \quad \textbf{NIZED AS THE SAME LABEL AS ABC ABOVE EVEN THOUGH}
* THIS LABEL DOES NOT OCCUR IN COLUMNS 2 - 7
* 20 \textbf{ENTER 1}
* 21 \textbf{ENTER 1}
* 22 \textbf{ENTER 1}
* 23 \textbf{SEIZE 33}
* 24 \textbf{SEIZE 1}
* 25 \textbf{SEIZE 1}
* 26 \textbf{SEIZE 1}
* 27 \textbf{SEIZE 1}
* 28 \textbf{SEIZE 5}
* 29 \textbf{SEIZE 5}
* 30 \textbf{SEIZE 5}
* 31 \textbf{SEIZE 1}
* 32 \textbf{SEIZE 2}
* 33 \textbf{SEIZE 1}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
34 SEIZE 1
35 SEIZE 1
36 SEIZE 4
37 SEIZE 4
38 SEIZE 1
39 SEIZE 1
40 SEIZE 1
41 SEIZE 2
42 SEIZE 10
43 SEIZE 1

ABC12 INITIAL X10 10
* NOTE STM LABEL IS PERMISSIBLE IN INITIAL CARD
* BUT IT IS NEVER ADDED TO THE LIST OF SYMBOLS
* THEREFORE IT MAY NEVER BE REFERENCED
44 TERMINATE

* ------------ PERMISSIBLE FUNCTION DATA CARD ARGUMENTS
* FOR PRECISE DEFINITION OF ALL PERMISSIBLE ARGU-
* MENTS OF FUNCTION DATA CARDS CONSULT APPENDIX A.
* NOTE THE READER SHOULD CONSULT SECTION B.2.B AND
* B.2.C, THE SYM TABLE AND THE ASSEMBLED PROGRAM,
* TO SEE HOW GPSS II RECOGNIZED THESE REFERENCE LABELS
*
1 FUNCTION V10 D2
30 14 106 15
* LABELS REFERENCED IN THE ABOVE FNC DATA CARD ARE
* ABC+1 ABC="/ ( ABC#* ABC{*) , ABC+1 IS NOT A REL-
* ATIVE LABEL.
2 FUNCTION 25 D2
30 103 40 107
55 FUNCTION V1 D2
9 12 50 90
6 FUNCTION V1 E2
12 12 13 19
7 FUNCTION V1 E2
14 12 15 50
88 FUNCTION P1 C2
18 120 123 400
* NOTE THE LABEL ABC 12 IS REFERENCED IN THE ABOVE
89 FUNCTION S10 D2
14 12 38 14
8 FUNCTION V1 E2
5 6 8 10
66 FUNCTION X10 D2
29 30 40 50
15 FUNCTION V12 E2
Q10 24 106 25
10 FUNCTION V10 E2
24 103 33 111

* ------------ PERMISSIBLE VARIABLE ARGUMENTS
* NOTE THE READER SHOULD CONSULT SECTION B.2.B AND
* B.2.C, THE SYM TABLE AND THE ASSEMBLED PROGRAM,
* TO SEE HOW GPSSIII RECOGNIZED THESE REFERENCE LABELS
*
1 VARIABLE 24/1*24/1+1-3
2 VARIABLE 2*39
3 VARIABLE 24*Q10
4 VARIABLE 2*25
6 VARIABLE 17+1
VARIABLE 36+1
VARIABLE 5+37+3
VARIABLE 24/12
VARIABLE 11+1
VARIABLE 5+1
VARIABLE 25+2
VARIABLE 1+20+25
VARIABLE 1+26
VARIABLE 4+1

*--------------------- PERMISSIBLE ARGUMENTS
* PRICISE DEFINITION OF PERMISSIBLE ARGUMENTS IS IN
* APFENDIX A, NOTE VARIETY OF PERMISSIBLE REF.
* LABELS IN ARGUMENTS OF STATEMENTS
*
45 ENTER 22
* SEE SECTION B.2.B AND SECTION B.2.C, SYM TABLE
* AND ASSEMBLE PROGRAM FOR GPSSIII RECOGNITION OF ABOVE
46 ENTER 46
47 PREEMPT 47
48 EXECUTE 48
49 EXECUTE 12
50 EXECUTE 49
51 RELEASE 24
52 TRANSFER 22
53 TRANSFER 12
54 TRANSFER 23
55 TRANSFER 60
56 TRANSFER 61
57 TRANSFER 71
58 TRANSFER BOTH 76 51
59 TRANSFER 76
60 TRANSFER ALL 76 51 5
* THE FIRST ALL IN ARG ABOVE IS NOT A LABEL
61 TRANSFER ALL 49 24 0
62 TRANSFER ALL 24 49 5
63 TRANSFER ALL 51 76 5
64 TRANSFER ALL 51 24 1
65 TRANSFER 52
66 TRANSFER ALL 52 51 1
67 TRANSFER ALL 24 51 1
68 TRANSFER 59
69 TRANSFER ALL 25 26 1
70 TRANSFER 48
71 TRANSFER 56
72 TRANSFER 55
73 TRANSFER 54
74 TRANSFER 43
75 TRANSFER 77
76 TRANSFER 77
77 TRANSFER 77
78 TRANSFER ALL 51 51 5
79 TRANSFER BOTH 51 54
80 TRANSFER 80
81 TRANSFER 42
82 TRANSFER 23
83 TRANSFER 25
84 TRANSFER 24
85 TRANSFER 25
86 TRANSFER 25

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SAVEVALUE 24 2
SAVEVALUE 24 1
SAVEVALUE 111 1
SAVEVALUE 127 1
ASSIGN N24
ASSIGN 24 1
ASSIGN 24 1
ASSIGN 51 1
ASSIGN P10 1
ASSIGN 24 1
ASSIGN 1+ 1
ASSIGN P10+ 1
PRINT 24 23
ASSEMBLE 24
SEIZE 31
SEIZE 25
SEIZE 2
SEIZE 104
SEIZE 105
SEIZE 106
SEIZE 107
SEIZE 108
SEIZE 109
SEIZE 25
SEIZE 51
SEIZE 111
SEIZE 22
SEIZE 31
SEIZE 76
SEIZE 60
SEIZE 12
SEIZE 25
SEIZE 33
QUEUE 40
QUEUE 25
QUEUE W40
QUEUE N40
RELEASE 24

*------------------- PERMISSIBLE VOID ARGUEMENTS
* NOTE A VOID ARG MAY BE RECOGNIZED AS DENOTING
* SEVERAL DIFFERENT VALUES. CONSULT SECTION B.2.C
*
ADVANCE
ASSIGN 1
DEPART 4
GATE LS 4
PRINT MOV
PRINT FUT
PRIORITY
SAVEVALUE 1
TEST GE
TEST NE
TRANSFER 136
TRANSFER 137

* IMPORTANT – A VOID ARGUMENT MAY DENOTE DIFFERENT
* VALUES TO GPSS III E.G. VOID ARG IN ADVANCE STM
* DENOTES ZERO, VOID ARG IN TRANSFER STM DENOTES
* NEXT BLOCK NUMBER, VOID ARG IN SPLIT STM DENOTES ONE
* TEST IF BLANKS EXECUTE CORRECTLY

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
137 GENERATE 1 1 1 1 1
* THE RECOGNITION OF VOID ARG IN INDEX AND ASSIGN STM
* TO BE PRINTED. SEE END OF SEC B.2.C
138 ADVANCE
139 ASSIGN 1 5
140 INDEX 1
141 SAVEVALUE 1 P1
142 PRINT 1 1
143 ASSIGN 2
144 SAVEVALUE 2 P2
145 PRINT 2 2
146 SAVEVALUE 3
147 PRINT 3 3
148 TERMINATE 1

*------------------ PERMISSIBLE COLUMN FORMAT
*------------------ PERMISSIBLE COLUMN FORMAT FOR LOGIC ATTRIB
* OR RELATIONAL MNEMONICS. IT IS STATED IN THE
* MANUALS (E.G. P.224 OF REF NO.9) THAT WHEN A LOGIC
* ATTRIB OR RELATIONAL MNEMONIC IS NEEDED IN THE
* AUX TEST, GATE, AND LOGIC BLOCKS IT MUST BEGIN IN
* COLUMN 13 FOR GATE AND TEST BLOCKS AND IN
* COLUMN 14 FOR LOGIC BLOCKS.
* 149 LOGIC R 1
150 LOGIC R 1
151 LOGIC R 1
152 LOGIC R 1
153 LOGIC R 1
154 TEST GE 12 12 1
155 TEST GE 12 12 1
156 TEST GE 12 12 1
11 VARIABLE VI
11 STORAGE 20
* TEST HOW LONG NUMBER CAN BE
* NOTE THE FORM OF NUMBERS PERMISSIBLE IN THE
* FOLLOWING STM, THESE NUM ARE NOT ALWAYS RECOGNIZED
* BY GPSS III AS THE NUM THEY DENOTE
* 157 TRANSFER 1 51 40
158 TRANSFER 222 21 22
* SEE SEC. B.2.C FOR RECOGNITION OF ABOVE NUMBERS
  INITIAL X10  123333333333333333333333333333333
  INITIAL X15  12345678912345678
159 GENERATE 1 1 1 1 1
* RECOGNITION OF THESE LONG NUMBERS IN INITIAL STM
* TO BE PRINTED. SEE END OF SEC. B.2.C
160 PRINT 15 15
161 PRINT 10 10
162 TERMINATE 1
*------------------ PERMISSIBLE AGUMENTS FOR GENERATE STATEMENT
* 163 GENERATE
164 TERMINATE
165 GENERATE 2 1 10
166 TERMINATE
167 GENERATE
168 TERMINATE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>169</td>
<td>GENERATE</td>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>TERMINATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>GENERATE</td>
<td>8</td>
<td>2</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>172</td>
<td>TERMINATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>GENERATE</td>
<td>12</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>TERMINATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>175</td>
<td>GENERATE</td>
<td>V1</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>TERMINATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>GENERATE</td>
<td>10</td>
<td>3</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>178</td>
<td>TERMINATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*----------- PSEUDO OPERATIONS *
* IT IS STATED IN THE MANUALS THAT LOCATION FIELD *
* MUST BE LEFT BLANK OF PSEUDO OPS ICT, ABS, ENDABS *
* THIS STM IS NOT CORRECT. HOWEVER IF A STM LABEL *
* IS USED IN THE LOC FIELD IT IS NOT ADDED TO THE *
* LIST OF SYM. E.G. ACFV IS NOT IN LIST OF SYM SEC-
* TION B.2.B. HENCE MAY NOT BE REFERENCED *

  179 SEIZE 1  
  40  SEIZE 6  

* NOTE BLOCK NUMBER OF ABOVE BLOCK  
  41 SEIZE 6  
  46 SEIZE 6  
  179 SEIZE 6  
  46 SEIZE 53  
  180 SEIZE 1  

* START 2  

<table>
<thead>
<tr>
<th>SAVEX NR,</th>
<th>VALUE</th>
<th>NR,</th>
<th>VALUE</th>
<th>NR,</th>
<th>VALUE</th>
<th>NR,</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>20377574019</td>
<td>NR,</td>
<td>VALUE</td>
<td>NR,</td>
<td>VALUE</td>
<td>NR,</td>
</tr>
<tr>
<td>10</td>
<td>20254118229</td>
<td>NR,</td>
<td>VALUE</td>
<td>NR,</td>
<td>VALUE</td>
<td>NR,</td>
</tr>
</tbody>
</table>

The above six lines are output resulting from the execution of this program. This output contains the values of "savevalue" locations initialized or indexed by the above program. Note, the numbers recognized by GPSS are not necessarily the numbers mentioned in the INITIAL cards.
* SIMULATE TEST SYNTAX

*---------------------------------------------------------------------ILLEGAL PROGRAM:---------------------------------------------------------------------

* EVERY STATEMENT BELOW (EXCEPT THOSE SO NOTED ) IS SYNTACTICALLY AN ILLEGAL STATEMENT
* THE READER MAY CONSULT APPENDIX A FOR CORRECT DEFINITIONS OF LEGAL CONSTRUCTIONS, AND TO LEARN WHY THE FOLLOWING CONSTRUCTIONS ARE ILLEGAL

*---------------------------------------------------------------------ILLEGAL STATEMENT

* ORIGINATE
* GPSS III WILL NOT RECOGNIZE THE ABOVE AS A STATEMENT. GPSS III MANUALS
STATE ORIGINATE IS A STATEMENT (E.G. PP.9 AND 54 OF REF NO.9)
* IT IS STATED IN THE MANUAL REF NO. 10 , P.76 THAT BLANK CARDS ARE PERMISSIB
**----------------------- BLANK CARD IS ILLEGAL @ FOLLOWING CARD IS BLANK

TEST GE 12,12,1 WHERE GE GOES
* THE MNEMONIC GE CANNOT START IN COLUMN 12 THE MANUALS IMPLY THAT THIS CONSTRUCTION IS NOT LEGAL. BUT THEY ALSO IMPLY THAT THE MNEMONIC CAN NOT START IN COLUMN 14,-18 WHERE IT WAS FOUND TO BE LEGAL
SEIZE 2 TEST IF BLANK SAACE IN OP FIELD
* BLANKS ARE NOT PERMISSIBLE IN THE OPERATION

IT IS STATED IN THE MANUALS THAT ANY CHARACTER IN COLUMN 1 WILL SUPRESS BLOCK COUNT STATISTICS ,P. 46 OF REF NO. 9 WILL CAUSE THE ENTIRE PROGRAM TO BE REJECTED

$ SEIZE 1 $ IN COLUMN 1
SEIZE 1

* ILLEGAL STATEMENT LABELS

* 123A SEIZE 1 TEST FORM OF STATEMENT LABELS
1 SEIZE 1 PERMISSIBLE STATEMENT LABELS
ABCDEFGSEIZE 1 LABEL 7 CHARACTERS LONG
ABCD2SEIZE 2 LABEL IN COLUMN 7

ABC123:RELEASE ABC123
* LABELS CAN NOT BE 7 CHARACTERS LONG
ILLEGAL VOID ARGUMENTS

JOBTAPE   VOID ARGUMENT
TEST      VOID TEST RELATIONAL MNEMONIC
SEIZE     WHERE CAN VOID ARGUMENTS BE
SAVEVALUE 1, WHERE CAN VOID ARGUMENTS BE
SAVEVALUE WHERE CAN VOID ARGUMENTS BE
RETURN    WHERE CAN VOID ARGUMENTS BE
RELEASE   WHERE CAN VOID ARGUMENTS BE
QUEUE     WHERE CAN VOID ARGUMENTS BE
PREEMPT   WHERE CAN VOID ARGUMENTS BE
LOOP      WHERE CAN VOID ARGUMENTS BE
LEAVE     WHERE CAN VOID ARGUMENTS BE
GATHER    WHERE CAN VOID ARGUMENTS BE
GATE LS   WHERE CAN VOID ARGUMENTS BE
CHANGE 4, WHERE CAN VOID ARGUMENTS BE
CHANGE WHERE CAN VOID ARGUMENTS BE
ASSEMBLE WHERE CAN VOID ARGUMENTS BE
LINK      WHERE CAN VOID ARGUMENTS BE
ASSIGN    WHERE CAN VOID ARGUMENTS BE
INDEX     WHERE CAN VOID ARGUMENTS BE

ILLEGAL ARGUMENTS

SPLIT 2,ABC,X1,X1 CAN C FIELD BE A SNA
SPLIT 3,ABC,X7
SPLIT 3,ABC,X1,15 SNA IN C FIELD
SPLIT 2,ASS,X*2
SPLIT 3,ABC,X1,15 SNA IN C FIELD
SPLIT 4,ASS,X*2
SPLIT 3,ABC,X1,15 SNA IN C FIELD
SPLIT 4,ASS,X*2
SPLIT 3,ABC,X1,15 SNA IN C FIELD
SPLIT 4,ASS,X*2

3 VARIABLE $ABCDEF+1
3 VARIABLE 1*AB123+1
2 VARIABLE 3 LABEL IN COLUMN ONE
UNLINK 1,1,ABC,ABC,ABC,ABC LABELS CAN NOT APPEAR IN C AND D FIELDS
* MAY NOT USE REFERENCE LABELS IN C AND D FIELDS OF UNLINK STATEMENT
INITIAL Q10,1 ANY SNA
INITIAL 1,3 NO PREFIX
INITIAL Q10,123456789
INITIAL X10,V1
ABC INITIAL V1,ABC
A-+12 SEIZE A-+12
A+++B SEIZE A+++B
A,, SEIZE $A,,
A$+$+ SEIZE A$+$+
SEIZE $A$+$+
A++++ SEIZE $A++++
A-+12 SEIZE $A-+12
ABC12* EXECUTE ABC12*
SEIZE ASSS
QUEUE $S$ABCDE
A=BC SEIZE P$ABC
ASSS INITIAL X12,1 LABEL ON INITIAL CARD GET REF

* THE ABOVE TWO STM USE THE SAME CONSTRUCTION AS W$ABC OR N$ABC WHICH
* ARE LEGAL. IT IS ILLEGAL TO USE THIS FORM WITH ANY OTHER STAN NUM ATTRIB
* OTHER THAN W OR N.
* THE ABOVE STM IS CORRECT BUT THE STM LABEL MAY NEVER BE REFERENCED.

TRANSFER ,ABCDEFG LABEL 7 CHARACTERS LONG
TRANSFER ,ABC+1-1
TRANSFER BOTH,ALL,3+ALL ADD IN REVERSE
TRANSFER BOTH,ALL,ALL+ALL ADD LABLES
TRANSFER ,A LL TEST IF BLANK IS PERMISSIBLE IN REFERENCE LABEL

* NOTE BLANK IS RECOGNIZED AS START OF COMMENT THEREFORE THE LABEL RECOGNIZE
* IS SIMPLY A WHICH DOES NOT OCCUR IN THIS PROGRAM

A LL SEIZE 2 THIS STM IS CORRECT - IT IS USED TO SHOW ABOVE IS INCORRECT
JOBTAPE SYSOU2,Q10,1,1 SNA IN 2 ARG
INDEX Q10,1 STANDARD NUMERICAL ATTRIBUTE PERRMISIBLE
TRANSFER ,U
ADVANCE 1,Q10 STANDARD A NUMERICAL ATTRIBUTE PERRMISSIBLE - NO
ADVANCE 1,RN1 RN1 PERMISSIBLE - NO
ASSIGN 1,NU STANDARD LOGICAL ATTRIBUTE PERRMISSIBLE - NO
ADVANCE *12 INDIRECT ADDRESS PERRMISSIBLE - NO

66 FUNCTION RN1,C1
12 30 COMMENTS ON DATA CARD
* THE ONLY CARD ON WHICH COMMENTS ARE NOT PERMITTED
* FUNCTION NUMBER IN COLUMN ONE NOT LEGAL
<table>
<thead>
<tr>
<th></th>
<th>FUNCTION RN1,C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>$ABC(</td>
</tr>
<tr>
<td>*</td>
<td>DOLLAR SIGN IN FRONT OF LABEL NOT PERMISSIBLE IN FNC DATA CARDS</td>
</tr>
<tr>
<td>3</td>
<td>FUNCTION RN1,C2</td>
</tr>
<tr>
<td>$A1111</td>
<td>$ABC(</td>
</tr>
<tr>
<td>5</td>
<td>FUNCTION RN1,C4</td>
</tr>
<tr>
<td>12</td>
<td>30 1 201A1</td>
</tr>
<tr>
<td></td>
<td>TRANSFER .11111111111444444,123ALLLLLLL,PPPPUT012345</td>
</tr>
<tr>
<td>*</td>
<td>IS COLUMN 72 USED</td>
</tr>
<tr>
<td></td>
<td>TRANSFER .111111111111111111111111111111111,ABC,ABCDE</td>
</tr>
<tr>
<td>*</td>
<td>CONTINUOUS ARGUMENT THROUGH COLUMN 72 IS NOT PERMISSIBLE</td>
</tr>
<tr>
<td>START</td>
<td>V1</td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
## Comparison of actual laboratory events and times of occurrence with those simulated

<table>
<thead>
<tr>
<th>Time (hours)</th>
<th>a.m.</th>
<th>p.m.</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>8.5</td>
<td>8.5</td>
<td>12.5</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>9.5</td>
<td>9.5</td>
<td>13.5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>10.5</td>
<td>10.5</td>
<td>14.5</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>11.5</td>
<td>11.5</td>
<td>15.5</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>12.5</td>
<td>12.5</td>
<td>16.5</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>13.5</td>
<td>13.5</td>
<td>17.5</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>14.5</td>
<td>14.5</td>
<td>18.5</td>
</tr>
</tbody>
</table>

### 3. Number of samples given accession

<table>
<thead>
<tr>
<th>Time (hours)</th>
<th>8</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
<th>10.5</th>
<th>11</th>
<th>11.5</th>
<th>12</th>
<th>12.5</th>
<th>13</th>
<th>13.5</th>
<th>14</th>
<th>14.5</th>
<th>15</th>
<th>15.5</th>
<th>16</th>
<th>16.5</th>
<th>17</th>
<th>17.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>22</td>
<td>41</td>
<td>53</td>
<td>65</td>
<td>97</td>
<td>124</td>
<td>129</td>
<td>134</td>
<td>137</td>
<td>141</td>
<td>142</td>
<td>144</td>
<td>152</td>
<td>157</td>
<td>164</td>
<td>166</td>
<td>166</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>22</td>
<td>40</td>
<td>53</td>
<td>65</td>
<td>97</td>
<td>126</td>
<td>131</td>
<td>134</td>
<td>137</td>
<td>141</td>
<td>141</td>
<td>142</td>
<td>152</td>
<td>159</td>
<td>163</td>
<td>166</td>
<td>166</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3. Number of samples logged by nonelectrolyte spinner

<table>
<thead>
<tr>
<th>Time (hours)</th>
<th>8</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
<th>10.5</th>
<th>11</th>
<th>11.5</th>
<th>12</th>
<th>12.5</th>
<th>13</th>
<th>13.5</th>
<th>14</th>
<th>14.5</th>
<th>15</th>
<th>15.5</th>
<th>16</th>
<th>16.5</th>
<th>17</th>
<th>17.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>11</td>
<td>17</td>
<td>18</td>
<td>21</td>
<td>40</td>
<td>53</td>
<td>53</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>11</td>
<td>17</td>
<td>19</td>
<td>22</td>
<td>41</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td>54</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3. Number of samples logged by electrolyte spinner

<table>
<thead>
<tr>
<th>Time (hours)</th>
<th>8</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
<th>10.5</th>
<th>11</th>
<th>11.5</th>
<th>12</th>
<th>12.5</th>
<th>13</th>
<th>13.5</th>
<th>14</th>
<th>14.5</th>
<th>15</th>
<th>15.5</th>
<th>16</th>
<th>16.5</th>
<th>17</th>
<th>17.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>11</td>
<td>24</td>
<td>35</td>
<td>44</td>
<td>55</td>
<td>71</td>
<td>76</td>
<td>80</td>
<td>85</td>
<td>87</td>
<td>88</td>
<td>90</td>
<td>98</td>
<td>103</td>
<td>110</td>
<td>112</td>
<td>112</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>11</td>
<td>23</td>
<td>34</td>
<td>43</td>
<td>56</td>
<td>72</td>
<td>77</td>
<td>80</td>
<td>83</td>
<td>87</td>
<td>87</td>
<td>88</td>
<td>98</td>
<td>105</td>
<td>109</td>
<td>112</td>
<td>112</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 5. Number of samples spun and put in ready rack

<table>
<thead>
<tr>
<th>Time (hours)</th>
<th>8</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
<th>10.5</th>
<th>11</th>
<th>11.5</th>
<th>12</th>
<th>12.5</th>
<th>13</th>
<th>13.5</th>
<th>14</th>
<th>14.5</th>
<th>15</th>
<th>15.5</th>
<th>16</th>
<th>16.5</th>
<th>17</th>
<th>17.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>4</td>
<td>29</td>
<td>43</td>
<td>52</td>
<td>65</td>
<td>85</td>
<td>106</td>
<td>117</td>
<td>121</td>
<td>121</td>
<td>132</td>
<td>144</td>
<td>155</td>
<td>165</td>
<td>166</td>
<td>166</td>
<td>166</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>4</td>
<td>29</td>
<td>45</td>
<td>52</td>
<td>64</td>
<td>84</td>
<td>103</td>
<td>117</td>
<td>121</td>
<td>121</td>
<td>131</td>
<td>142</td>
<td>155</td>
<td>163</td>
<td>163</td>
<td>166</td>
<td>166</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

APPENDIX C
<table>
<thead>
<tr>
<th>TIME (hours)</th>
<th>8</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
<th>10.5</th>
<th>11</th>
<th>11.5</th>
<th>12</th>
<th>12.5</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3</th>
<th>3.5</th>
<th>4</th>
<th>4.5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GLUCOSE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of glucose samples given accession number by glucose technologist</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>20</td>
<td>27</td>
<td>45</td>
<td>65</td>
<td>65</td>
<td>71</td>
<td>74</td>
<td>75</td>
<td>75</td>
<td>80</td>
<td>84</td>
<td>84</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>Number of glucose in incubating rack</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>31</td>
<td>43</td>
<td>66</td>
<td>66</td>
<td>70</td>
<td>72</td>
<td>72</td>
<td>72</td>
<td>80</td>
<td>80</td>
<td>84</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>Number of glucose samples that have been read</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>51</td>
<td>51</td>
<td>54</td>
<td>58</td>
<td>65</td>
<td>70</td>
<td>73</td>
<td>75</td>
<td>81</td>
<td>62</td>
</tr>
<tr>
<td><strong>UREA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of samples at urea bench</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>21</td>
<td>40</td>
<td>43</td>
<td>59</td>
<td>90</td>
<td>96</td>
<td>100</td>
<td>101</td>
<td>102</td>
<td>106</td>
<td>108</td>
<td>109</td>
<td>110</td>
<td>114</td>
</tr>
<tr>
<td>Number of urea samples given accession number</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>21</td>
<td>38</td>
<td>43</td>
<td>56</td>
<td>67</td>
<td>85</td>
<td>99</td>
<td>102</td>
<td>103</td>
<td>105</td>
<td>108</td>
<td>109</td>
<td>110</td>
<td>113</td>
</tr>
<tr>
<td>TIME (hours)</td>
<td>a.m.</td>
<td>p.m.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.5</td>
<td>8.5</td>
<td>8.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.5</td>
<td>9.5</td>
<td>9.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.5</td>
<td>10.5</td>
<td>10.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.5</td>
<td>11.5</td>
<td>11.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.5</td>
<td>12.5</td>
<td>12.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>3.5</td>
<td>3.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Number of samples | a | 0 | 15 | 57 | 74 | 110 | 120 | 170 | 172 | 200 | 214 | 250 | 270 | 280 | 282 | 285 | 285 | 285 |
| and standards    |   |   |    |    |    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| entered hydrolysis | s | 0 | 16 | 58 | 78 | 120 | 126 | 172 | 172 | 206 | 214 | 258 | 274 | 280 | 282 | 284 | 290 | 290 |

<table>
<thead>
<tr>
<th>TOTAL PROTEIN</th>
<th>ALBUMEN, GLOBULIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of samples</td>
<td>a</td>
</tr>
</tbody>
</table>

reached total protein,

albumen, globulin

bench

s | 0 | 0 | 0 | 0 | 24 | 28 | 35 | 40 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 |

| Number total      | 0 | 0 | 0 | 0 | 4 | 24 | 29 | 38 | 43 | 43 | 43 | 43 | 43 | 43 | 43 | 43 | 43 | 43 |

proteins waiting
to be read

s | 0 | 0 | 0 | 0 | 4 | 24 | 30 | 40 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 |

| Number of total   | a | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 82 | 84 |

protein, albumen

globulin read including
duplicates

s | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 83 | 84 | 84 | 84 |
<table>
<thead>
<tr>
<th>Time (hours)</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
<th>10.5</th>
<th>11</th>
<th>11.5</th>
<th>12</th>
<th>12.5</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3</th>
<th>3.5</th>
<th>4</th>
<th>4.5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.m.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p.m.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electrolytes: Sodium, Potassium, Chloride</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number at sodium,</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>24</td>
<td>32</td>
<td>45</td>
<td>50</td>
<td>57</td>
<td>62</td>
<td>62</td>
<td>65</td>
<td>66</td>
<td>75</td>
<td>80</td>
<td>83</td>
<td>83</td>
<td>84</td>
<td>87</td>
</tr>
<tr>
<td>Potassium chloride</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bench</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>25</td>
<td>33</td>
<td>45</td>
<td>48</td>
<td>58</td>
<td>62</td>
<td>62</td>
<td>65</td>
<td>65</td>
<td>74</td>
<td>80</td>
<td>83</td>
<td>84</td>
<td>84</td>
<td>87</td>
</tr>
<tr>
<td>Number read at</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>64</td>
<td>72</td>
<td>92</td>
<td>114</td>
<td>124</td>
<td>125</td>
<td>130</td>
<td>130</td>
<td>134</td>
<td>146</td>
<td>166</td>
<td>168</td>
<td>168</td>
<td>174</td>
</tr>
<tr>
<td>Sodium and potassium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(one term for Na, one for K)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>66</td>
<td>74</td>
<td>90</td>
<td>112</td>
<td>124</td>
<td>124</td>
<td>130</td>
<td>130</td>
<td>132</td>
<td>144</td>
<td>164</td>
<td>168</td>
<td>168</td>
<td>174</td>
</tr>
<tr>
<td>Number read at</td>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>52</td>
<td>80</td>
<td>94</td>
<td>114</td>
<td>124</td>
<td>124</td>
<td>124</td>
<td>130</td>
<td>130</td>
<td>134</td>
<td>160</td>
<td>166</td>
<td>168</td>
</tr>
<tr>
<td>Chloride test including</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duplicates</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>53</td>
<td>79</td>
<td>96</td>
<td>115</td>
<td>124</td>
<td>124</td>
<td>130</td>
<td>130</td>
<td>134</td>
<td>159</td>
<td>166</td>
<td>168</td>
<td>168</td>
<td>174</td>
</tr>
<tr>
<td>Number of samples</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>22</td>
<td>32</td>
<td>40</td>
<td>50</td>
<td>57</td>
<td>62</td>
<td>62</td>
<td>65</td>
<td>66</td>
<td>70</td>
<td>80</td>
<td>83</td>
<td>83</td>
<td>84</td>
<td>87</td>
</tr>
<tr>
<td>at bicarbonate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bench</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>22</td>
<td>33</td>
<td>41</td>
<td>48</td>
<td>58</td>
<td>62</td>
<td>62</td>
<td>65</td>
<td>65</td>
<td>69</td>
<td>80</td>
<td>83</td>
<td>84</td>
<td>84</td>
<td>87</td>
</tr>
<tr>
<td>TIME (hours)</td>
<td>8</td>
<td>8.5</td>
<td>9</td>
<td>9.5</td>
<td>10</td>
<td>10.5</td>
<td>11</td>
<td>11.5</td>
<td>12</td>
<td>12.5</td>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>2.5</td>
<td>3</td>
<td>3.5</td>
<td>4</td>
<td>4.5</td>
</tr>
<tr>
<td>-------------</td>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>-----</td>
<td>----</td>
<td>------</td>
<td>----</td>
<td>------</td>
<td>----</td>
<td>------</td>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>-----</td>
<td>---</td>
<td>-----</td>
</tr>
<tr>
<td>a.m.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of samples</td>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>34</td>
<td>40</td>
<td>45</td>
<td>52</td>
<td>60</td>
<td>65</td>
<td>73</td>
<td>77</td>
<td>83</td>
</tr>
<tr>
<td>read at bicarbonate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>s</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>25</td>
<td>33</td>
<td>39</td>
<td>47</td>
<td>54</td>
<td>62</td>
<td>67</td>
<td>74</td>
<td>81</td>
<td>84</td>
</tr>
</tbody>
</table>

*Time (hours), topline, refers to the time the count of events was taken.

**See Figure 1 for numbers in this column.

***Items in this column are descriptions of the number event**.

a refers to the number of events observed at the time noted.

s refers to the number of events determined by simulation.
APPENDIX D
GPSS MODEL OF CLINICAL LABORATORY

In this appendix is presented a version of the GPSS program that was used to implement a model of the Clinical Chemistry Laboratory of Yale-New Haven Hospital on a digital computer. The program is well documented. The program may be used to study further the simulation techniques presented in Chapter IV; or to study the organization and procedures of a clinical chemistry laboratory as was briefly discussed in Appendix C; or to be used as a reference of more conventional examples of the syntax of GPSS than were presented in Appendix B. The form of all of the constructions used in the program given in this section have been defined in Chapter III and Appendix A.

The version of the program given in this appendix implements a model of the clinical laboratory for a particular day. The results of the execution of this program were used to give confidence in the validity of the model (see Section IV. 14). The model generates (GENERATE section of program) the same inputs as were recorded for a day in the laboratory. At every half-hour of simulation time the program will cause a status of the model to be given. These statistics were compared with statistics taken of the actual laboratory at corresponding times. The results of this comparison are presented in Table IV-1. This program was also used to observe the response of a model of a particular data processing configuration to conditions prevailing in the laboratory for a particular day.

An overall flow chart of the model of the laboratory is depicted in
Figure D-1. The program is organized into sections corresponding to the blocks of Figure D-1.
Figure D-1 Flow chart of a simulation of the clinical chemistry laboratory.
* SIMULATE HOSPITAL

* THE FOLLOWING CARDS ARE COMMENTS IDENTIFYING THE VARIOUS ENTITIES USED

* PARAMETER 12 NUMBER OF TUBES WITH SAMPLE
* PARAMETER 13 NUMBER OF STORAGE READY RACK OR NO. OF AUTOCLAVE RACK
* PARAMETER 14 NO OF UREA TEC SEIZED BY SET
* PARAMETER 15 NO OF TECHNITION LOGING FOR BUN OR GLU TEST
* PARAMETER 16 EMERGENCY IF EQUAL TO FIVE
* PARAMETER 17 NO. OF SPINNER TEC, NO OF CENT SPIN
* PARAMETER 18 NO. OF WAITING RACK WAITING TO ENTER CENT.
* PARAMETER 20 = 25 IF SAMP HAS BEEN PROCESSED BY GLU TEC
* PARAMETER 21 TIME FOR COMPUTER TO COMPUTE RESULT OF TEST ON SAMP
* PARAMETER 22 = 5 IF CONTROL FOR GLU 6 FOR UREA, 7 IF STANDARD FOR UR
* PARAMETER 24 NO OF LOGIC SWITCH PREVENTING SAMP ENTER SPINNING CENT
* QUEUE 2 WHOLE BLOOD LINE AT PUNCH BENCH
* QUEUE 3 SPINNER TO PROCESS - NON ELEC
* QUEUE 4 SPINNER TO PROCESS - ELECTROLYTE
* QUEUE 5 SAMP WAIT FOR COMPUTER TO COMPUTE RESULTS OF TEST
* QUEUE 6 SAMP WAIT FOR COMPUTER TO PRINT SAMP REPORT
* QUEUE 7 COMPUTER QUE WAIT TO PRINT WARD REPORT
* QUEUE 8 COMPUTER QUE WAIT TO PRINT SECTION REPORT
* QUEUE 9 DOCTOR REQUEST FOR PARTIAL REPORT QUEUE
* QUEUE 12 MICROTUBES
* QUEUE 13 WHOLE BLOOD LINE AT BENCH
* QUEUE 14 GLUCOSE LINE BEFORE PIPETING
* QUEUE 15 UREA LINE BEFORE LOGGED IN
* QUEUE 16 UREA WAITING TO BE PROCESSED
* LOGIC SWITCH 3 1ST. SAMP GETS LOG REMAINING MUST SEIZE SAME TEC.
* LOGIC SWITCH 4 LET NEXT SAMP SEIZE TEC ABOVE
* LOGIC SWITCH 5 WAIT TO INCUBATE AT GLU
* LOGIC SWITCH 6 LET REST OF SET SEIZE SAME TEC FOR UREA LOG
* LOGIC SWITCH 11 FIRST OF SET SEIZE TEC FOR UREA LOG
* LOGIC SWITCH 12 FIRST OF SET SEIZE TEC FOR UREA PIPETING
* LOGIC SWITCH 13 LET REST OF SET SEIZE SAME TEC FOR UREA PIPETING
* LOGIC SWITCH 14 FIRST OF SET SEIZE UREA TEC FOR ALKALINIZATION
* LOGIC SWITCH 15 LET REST OF SET SEIZE SAME TEC FOR ALKALINIZATION
* LOGIC SWITCH 16 FIRST OF SET SEIZE UREA TEC TO REMOVE FROM WHEEL
* LOGIC SWITCH 17 LET REST OF SET SEIZE SAME TEC FOR REMOVAL FROM WHEEL
* LOGIC SWITCH 18 FIRST OF SET SEIZE TEC FOR UREA READ
* LOGIC SWITCH 19 LET REST OF SET SEIZE SAME TEC FOR UREA READ
* LOGIC SWITCH 20 LET FIRST URIC ACID SAMP OF BATCH TEST FOR TOO LONG WAIT
* LOGIC SWITCH 21 MAKE SURE FLAME PHOTOMETER IS CALIBRATED FOR NA K
* LOGIC SWITCH 22 MAKE SURE GLU COLORIMETER IS CALIBRATED FOR BATCH
* LOGIC SWITCH 23 MAKE TOT PROT TEC PREPARE SODIUM SULFATE
* LOGIC SWITCH 24 GLU MUST MAKE STANDARDS WITH EACH INCUBATION
* LOGIC SWITCH 25 UREA MUST MAKE CONTROLS WITH EACH WHEEL
* LOGIC SWITCH 26 PREVENT SAMP FROM ENTERING SPINNING CENT -NON ELEC
* LOGIC SWITCH 27 PREVENT SAMP FROM ENTERING SPINNING CENT -ELEC
* SAVEVALUE 1 NO. OF TEC LOGGING
* SAVEVALUE 2 NUMBER OF INCUBATING RACK
* SAVEVALUE 3 PROPER READY RACK FOR NON ELEC SPIN
* SAVEVALUE 4 PROPER READY RACK FOR ELEC SPIN
* SAVEVALUE 5 NO. OF UREA TEC SEIZED FOR LOG SET
* SAVEVALUE 6 NO. OF UREA TEC SEIZED FOR PIPETTING SET
* SAVEVALUE 7 NO. OR UREA TEC SEIZED FOR ALKALINIZATION SET
* SAVEVALUE 8 NO. OF UREA TEC SEIZED FOR REMOVAL FROM WHEEL
* SAVEVALUE 9 NO. OF UREA TEC SEIZED FOR READ SET
* SAVEVALUE 10 PRIORITY OF SAMPLE
* SAVEVALUE 11 PRIORITY OF SAMPLE
* SAVEVALUE 12 NO. OF SAMP COMPUTER HANDLES TODAY
* SAVEVALUE 13 =0 WHEN EVER STANDARDS MUST BE MAKE FOR GLU TEST
* SAVEVALUE 14 KEEP TRACK OF NUMBER TO BE PROCESSED BY NA K TEC
* SAVEVALUE 15 KEEP TRACK OF NO. OF URIC ACID SAMP
* SAVEVALUE 16 NUMBER OF ELECTROLITES LOGGED IN BY SPINNER
* STORAGE 2 TIME PUNCH TRAY - UNLIMITED
* STORAGE 5-14 READY RACK .13 CAP. FOR ELEC SPIN
* STORAGE 15 MAGNETIC STIRRER FOR GLUCOSE - CAP. 8
* STORAGE 16 CENTRIFUGE FOR GLU - CAP. 8
* STORAGE 17 WAITING PLACE FOR CENTRIFUGE FOR GLU - UNLIMITED
* STORAGE 18,19,20 GLU RACK WAITING TO INCUBATE CAP. 114
* STORAGE 29 AUTOCLAVE FOR GLUCOSE
* STORAGE 30 WAITING RACK TO PIPET UREA - UNLIMITED
* STORAGE 31 UREA WHEEL NO.1- CAPACITY 30
* STORAGE 32 UREA WHEEL NO.2- CAPACITY 30
* STORAGE 33 WAITING RACK FOR UREA WHEEL FREE - UNLIMITED
* STORAGE 34 Nessler Bottle Box For Urea - UNLIMITED
* STORAGE 35-44 READY RACK .13 CAP. FOR NON ELEC SPIN
* STORAGE 45 URIC ACID RACK AT URIC ACID BENCH
* STORAGE 46 URIC ACID REACTION RACK
* STORAGE 47 CHLORIDE RACK AT BENCH
* STORAGE 48 NA K CHLORIDE COLLECTING RACK
* STORAGE 49 RACK AT NA K BENCH
* STORAGE 50 CHLORIDOMETER - CAPACITY 1
* STORAGE 51 CHLORIDE WAITING FOR CHLORIDOMETER - CAPACITY 1
* STORAGE 52 WAITING RACK FOR CENT OF SPINNER-NON ELECT
* STORAGE 53 WAITING RACK FOR CENT OF SPINNER-ELECT
* STORAGE 54 CHLORID WAITING RACK FOR MACHINE TO EMPTY
* STORAGE 55 POSITION IN READY RACK FOR BICARBONATE
* STORAGE 56 WAIT RACK FOR POSITION IN AIRATOR
* STORAGE 57 BICARBONATE AIRATOR
* STORAGE 58 POSITION IN READY RACK FOR PROT ALBUMIN GLOBIUM
* STORAGE 59 TOTAL PROTEIN ALBUMIN GLOBIUM RACK AT BENCH
* STORAGE 60 TOT PROT ALBUMIN GLOBIUM FILTER RACK
* STORAGE 61 SODIUM POTASSIUM RACK AT BENCH
* STORAGE 62 URIC ACID CENT
* FACILITY 1 TEC WHO TIME PUNCHES
* FACILITY 2 TIME PUNCH MACHINE
* FACILITY 3 NON ELEC SPINNER
* FACILITY 4 ELEC SPINNER
* FACILITY 5 CENTRIFUGE FOR SPINNER
* FACILITY 7 GLU TEC, NORMALLY INCUBATE
* FACILITY 8 GLU TEC, NORMALLY PIPETING
* FACILITY 9 UREA TEC, NORMALLY FOR LOGGING
* FACILITY 10 UREA TEC, NORMALLY OFR PIPETING
* FACILITY 11 COLORIMETER FOR GLUCOSE
* FACILITY 12 CENT FOR URIC ACID
* FACILITY 13 CHLORIDE TEC
* FACILITY 14 SODIUM POTASSIUM TEC
* FACILITY 15 COMPUTER
* FACILITY 16 BICARBONATE TEC WHO TITRATES AND READS
* FACILITY 17 TOT PROT TEC AND ALBUMIN GLOBIUM TEC
* FACILITY 18 BICARBONATE READ OUT MACHINE
* FACILITY 19 COLORIMETER FOR URIC ACID AND TOT PROT A G
* FACILITY 20 BICARBONATE TEC WHO PIPETS AND BRINGS TO BENCH
* USER CHAIN 1  SAMPLES IN GLU SAKER
* USER CHAIN 2  SAMP WAIT TO BE TAKEN TO SPIN FROM TIME PUNCH
* USER CHAIN 3  SAMP IN NON ELEC CENT WAITING TO SPIN
* USER CHAIN 4  SAMP IN ELEC CENT WAITING TO SPIN
* USER CHAIN 5  URIC ACID WAITING TO BE PROCESSED
* USER CHAIN 6  WAIT FOR COMPUTER FREE OR DOING A LOS PRIORITY JOB
* USER CHAIN 7  SAMP WAIT TO GO TO WHOLE BLD BENCH FROM TIME PUNCH
* USER CHAIN 8  NON ELEC SAMP WAIT IN READY RACT TO BE BROUGHT TO BENCH
* USER CHAIN 9  ELEC SAMP WAIT IN READY RACT TO BE BROUGHT TO BENCH
* USER CHAIN 10  WAITING PLACE FOR TOT PROT AG TO BE READ ON MACHINE
* USER CHAIN 11  WAITING FOR UREA WHEEL FREE
* USER CHAIN 12  ALBUMIN GLOBLUM WAITING TO BE READ

UNIT OF TIME ONE TENTH OF A MINUTE

GENERATE BLOOD SAMPLES


FOLLOWING EACH GENERATE STATEMENT IS A TRANSFER STATEMENT. THIS TRANSFERS TO A ROUTINE BELOW THAT Assigns THE CORRECT VALUES TO THE SCALARS OF THE N-TUPLE GENERATED.

*LOC NAME A,B,C,D,E
GENERATE 0,0,4900,5,5,30  8-10AM, 5 SAMP ARV, PRIORITY 5, 30 PARAM
SAVEVALUE 2,K18  INITIAL VALUE OF INCUBATING RACK FOR GLU
SAVEVALUE 3,K5  INITIAL NO. OF READY RACK FOR NON-ELEC
SAVEVALUE 4,K35  INITIAL NO. OF READY RACK FOR ELEC
SAVEVALUE 6, K18
TRANSFER UEL
GENERATE 0, 0, 4950, 3, 5, 30
TRANSFER GUE
GENERATE 0, 0, 4950, 6, 5, 30
TRANSFER UEL
GENERATE 0, 0, 4925, 1, 5, 30
TRANSFER ALL
GENERATE 0, 0, 4950, 1, 5, 30
TRANSFER UAGUE
GENERATE 0, 0, 4950, 1, 5, 30
TRANSFER UAUET
GENERATE 0, 0, 4950, 2, 5, 30
TRANSFER UREO
GENERATE 0, 0, 4950, 1, 5, 30
TRANSFER GLUO
GENERATE 0, 0, 5000, 1, 5, 30
TRANSFER TPGU
GENERATE 0, 0, 5000, 1, 5, 30
TRANSFER UAGU
GENERATE 0, 0, 4950, 3, 5, 30
TRANSFER TPA
GENERATE 0, 0, 4950, 2, 5, 30
TRANSFER GUE
GENERATE 0, 0, 4950, 2, 5, 30
TRANSFER ELE
GENERATE 0, 0, 4950, 2, 5, 30
TRANSFER TPGUE
GENERATE 0, 0, 5000, 9, 5, 30
TRANSFER OTHER
GENERATE 0, 0, 4950, 4, 5, 30
TRANSFER GUR
GENERATE 0, 0, 4950, 3, 5, 30
TRANSFER URE
GENERATE 0, 0, 4920, 2, 5, 30
TRANSFER GLU
GENERATE 0, 0, 5400, 4, 5, 30
TRANSFER UEL

INITIAL NO. OF INCUBATE RACK
8-15AM GO WITH THE TWO BELOW
8.20
8.15
8.15
8.15
8.15 - WB
8.15 - WB
8.12 - WB
9.00
| GENERATE  | 0,0,5400,1,5,30 | 9.00  |
| TRANSFER | ,GUE          |       |
| GENERATE  | 0,0,5400,2,5,30 | 9.00  |
| TRANSFER | ,ÉLÉ          |       |
| GENERATE  | 0,0,5400,1,5,30 | 9.00  |
| TRANSFER | ,TPGUE        |       |
| GENERATE  | 0,0,5400,1,5,30 | 9.00  |
| TRANSFER | ,TPUE         |       |
| GENERATE  | 0,0,5600,2,5,30 | 9.20  |
| TRANSFER | ,GUR          |       |
| GENERATE  | 0,0,5600,1,5,30 | 9.20  |
| TRANSFER | ,ÉLÉ          |       |
| GENERATE  | 0,0,5600,1,5,30 | 9.20  |
| TRANSFER | ,TPA          |       |
| GENERATE  | 0,0,5600,1,5,30 | 9.20  |
| TRANSFER | ,OTHER        |       |
| GENERATE  | 0,0,5860,7,5,30 | 9.46  |
| TRANSFER | ,GUE          |       |
| GENERATE  | 0,0,6300,9,5,30 | 9.59  |
| TRANSFER | ,UEL          |       |
| GENERATE  | 0,0,5990,4,5,30 | 9.59  |
| TRANSFER | ,ALL          |       |
| GENERATE  | 0,0,5990,2,5,30 | 9.59  |
| TRANSFER | ,ÉLÉ          |       |
| GENERATE  | 0,0,5990,3,5,30 | 9.50  |
| TRANSFER | ,TPUE         |       |
| GENERATE  | 0,0,5900,2,5,30 | 9.50  |
| TRANSFER | ,TPGUE        |       |
| GENERATE  | 0,0,5900,1,5,30 | 9.50  |
| TRANSFER | ,ALL          |       |
| GENERATE  | 0,0,5900,1,5,30 | 9.50  |
| TRANSFER | ,ÉLÉ          |       |
| GENERATE  | 0,0,5950,2,5,30 | 9.55  |
| TRANSFER | ,GLUO         |       |
| GENERATE  | 0,0,5950,1,5,30 | 9.55  |
| TRANSFER | ,UAG          |       |
| GENERATE  | 0,0,5950,1,5,30 | 9.55  |
| TRANSFER | GENERATE | 0.0, 6300, 8, 5, 30 | 10.30 - WB |
| TRANSFER | GENERATE | 0.0, 6300, 2, 5, 30 | 10.30 - WB |
| TRANSFER | GENERATE | 0.0, 6900, 3, 5, 30 | 11.30 |
| TRANSFER | GENERATE | 0.0, 6900, 1, 5, 30 | 11.30 - WB |
| TRANSFER | GENERATE | 0.0, 7360, 1, 5, 30 | 12.16 |
| TRANSFER | GENERATE | 0.0, 7470, 1, 5, 30 | 12.27 |
| TRANSFER | GENERATE | 0.0, 7470, 4, 5, 30 | 12.27 |
| TRANSFER | GENERATE | 0.0, 7470, 1, 5, 30 | 12.27 - WB |
| TRANSFER | GENERATE | 0.0, 7950, 1, 5, 30 | 1.15 - WB |
| TRANSFER | GENERATE | 0.0, 7760, 1, 5, 30 | 6 |
| TRANSFER | GENERATE | 0.0, 8250, 1, 5, 30 | 1.45 |
| TRANSFER | GENERATE | 0.0, 8250, 1, 5, 30 | 1.45 |
| TRANSFER | GENERATE | 0.0, 8250, 1, 5, 30 | 1.45 - WB |
| TRANSFER | GENERATE | 0.0, 8250, 1, 5, 30 | 1.45 - WB |
| TRANSFER | GENERATE | 0.0, 8250, 1, 5, 30 | 1.45 - WB |
| TRANSFER | GENERATE | 0.0, 8130, 1, 5, 30 | 1.30 |
| TRANSFER | GENERATE | 0.0, 8540, 9, 5, 30 | 2.14 |
| TRANSFER | GENERATE | 0.0, 8540, 1, 5, 30 | 2.14 |
TRANSFER, ELE
GENERATE 0,0,8740,1,5,30  2.34
TRANSFER, UEL
GENERATE 0,0,8920,1,5,30  2.52
TRANSFER, GUE
GENERATE 0,0,8920,1,5,30  2.52
TRANSFER, TPUAU
GENERATE 0,0,8920,2,5,30  2.52
TRANSFER, UAC
GENERATE 0,0,8920,2,5,30  2.52
TRANSFER, TPUA
GENERATE 0,0,8920,1,5,30  2.52
TRANSFER, OTHER
GENERATE 0,0,9120,1,5,30  3.12
TRANSFER, OTHER
GENERATE 0,0,9120,1,5,30  3.12 - WB
TRANSFER, URE
GENERATE 0,0,9120,4,5,30  3.12
TRANSFER, GLU
GENERATE 0,0,9400,4,5,30  3.40 - WB
TRANSFER, GLU
GENERATE 0,0,9520,2,5,30  3.52
TRANSFER, ELE
GENERATE 0,0,9520,1,5,30  3.52
TRANSFER, TPE

ALL ASSIGN 5,1       URIC ACID
TPGUE ASSIGN 7,1     TOT. PROT ABIUM AND GLOBIUM
GUE ASSIGN 1,1       GLU
UEL ASSIGN 2,1       UREA
ELE ASSIGN 3,1       ELECTROLITE
TRANSFER, ENTER
TPUE ASSIGN 7,1     TOT. PROT ABIUM AND GLOBIUM
TRANSFER, UEL
UAGUE ASSIGN 5,1     URIC ACID
TRANSFER, GUE
UAUET ASSIGN 5,1     URIC ACID
ASSIGN 7,1       TOT. PROT ABIUM AND GLOBIUM
<table>
<thead>
<tr>
<th>Test</th>
<th>Assignment</th>
<th>Tube</th>
</tr>
</thead>
<tbody>
<tr>
<td>GURO</td>
<td>4,1</td>
<td>UEL</td>
</tr>
<tr>
<td>UREO</td>
<td>4,1</td>
<td>URE</td>
</tr>
<tr>
<td>TPGU</td>
<td>7,1</td>
<td>GUR</td>
</tr>
<tr>
<td>GUR</td>
<td>1,1</td>
<td>GUR</td>
</tr>
<tr>
<td>URE</td>
<td>2,1</td>
<td>URE</td>
</tr>
<tr>
<td>GLUO</td>
<td>4,1</td>
<td>ENTER</td>
</tr>
<tr>
<td>GLU</td>
<td>1,1</td>
<td>GLU</td>
</tr>
<tr>
<td>UAGU</td>
<td>5,1</td>
<td>ENTER</td>
</tr>
<tr>
<td>TPE</td>
<td>3,1</td>
<td>GUR</td>
</tr>
<tr>
<td>TPA</td>
<td>7,1</td>
<td>GUR</td>
</tr>
<tr>
<td>OTHER</td>
<td>4,1</td>
<td>ENTER</td>
</tr>
<tr>
<td>UAG</td>
<td>5,1</td>
<td>GLU</td>
</tr>
<tr>
<td>TPUAU</td>
<td>7,1</td>
<td>GLU</td>
</tr>
<tr>
<td>UAU</td>
<td>5,1</td>
<td>GLU</td>
</tr>
<tr>
<td>TPUA</td>
<td>7,1</td>
<td>URE</td>
</tr>
<tr>
<td>UAC</td>
<td>5,1</td>
<td>URE</td>
</tr>
<tr>
<td>ENTER</td>
<td>ENTER+1</td>
<td></td>
</tr>
<tr>
<td>ASU12</td>
<td>12, FN60</td>
<td></td>
</tr>
</tbody>
</table>

* TIME PUNCHED *

- TIMPU ENTER 2
- TIMPU PRIORITY 4, BUFFER
- TIMPU SEIZE 1
- TIMPU SEIZE 2
- TIMPU LEAVE 2

Wait in tray to be time punched
Wait for all samp this delivery
Technition that time punches
Time punch
Leave tray and be punched

**Number of Tubes:**

- TOT. PROT ABIUM AND GLOBIUM
- GLU
- UREA
- ELECTROLITE
- TOT. PROT ABIUM AND GLOBIUM
- OTHER TEST REQUESTED
- URIC ACID
- TOT. PROT A G
- URIC ACID
- TOT. PROT A G
- URIC ACID
- NUMBER OF TUBES
ADVANCE 2
RELEASE 1
RELEASE 2
TEST E V1,K0,SPINR
PRIORITY 0,BUFFER
TRANSFER ALL,SE11Z,LINK2,3
SEIZ SEIZE 1
ASSIGN 15,1
TRANSFER WBLD1
SEIZE 7
ASSIGN 15,7
TRANSFER WBLD1
SEIZE 8
ASSIGN 15,8
TRANSFER WBLD1
SEIZE 9
ASSIGN 15,9
TRANSFER WBLD1
SEIZE 10
ASSIGN 15,10
TRANSFER WBLD1

LINK2 LINK 7,FIFO
WBLD1 ADVANCE 10
RELEASE P15
UNLINK 7,WBLD,ALL
WBLD QUEUE 2
PRIORITY 4
PRIORITY 3,BUFFER
TRANSFER WBLDB
SPINR TEST LE V10,K6600,SPIN1
SPIN2 TEST E P3,K0,SPIN1
ASSIGN 17,3
ASSIGN 18,52
ASSIGN 24,26
*
TRANSFER SEIZ1-1
SPIN1 ASSIGN 17,4

TIME IT TAKES TO PUNCH AND STORE
RELEASE TEC. THAT TIME PUNCHES
RELEASE TIME PUNCH
TEST TO SEE IF GO TO SPINNER
LET ANY OTHER IN SYSTEM SEIZE TEC@S
SEIZE ANY ONE OF 5 TEC
TEC WHO TIME PUNCHES
PAR 15 = NO. OF TEC SEIZED

GLU TEC NORMALLY INCUBATE
PAR 15 = NO. OF TEC SEIZED

GLU TEC NORMALLY PIPETING
PAR 15 = NO. OF TEC SEIZED

UREA TEC NORMALLY FOR LOGGING
PAR 15 = NO. OF TEC SEIZED

UREA TEC NORMALLY FOR PIPETING
PAR 15 = NO. OF TEC SEIZED

WAIT HERE TIL TEC FREE
TIME TO BRING TO WB BENCH
TEC SEIZED
ALL IN LINK ABOVE GO TO WB BENCH
WHOLE BLOOD WAIT TO BE PROCESSED
TREAT AS SET
GO TO WHOLE BLOOD BENCH
IF CLOCK TIME LESS THAN 11 2 SPI
TEST IF ELECTROLYTES
NO. OF NON ELECT SPINNER
NO. OF WAIT RACK NONELEC CENT
NO. OF SWITCH PREVENTING SAMP FROM
ENTERING SPINNING CENT

NO. OF ELECT SPINNER
ASSIGN  18, 53
ASSIGN  24, 27
PRIORITY 0, BUFFER
TRANSFER ALL, SEIZ1, LINK1, 3
SEIZ1 SEIZE 1
ASSIGN 15, 1
TRANSFER QUP16
SEIZE 3
ASSIGN 15, 3
TRANSFER QUP16
SEIZE 4
ASSIGN 15, 4
TRANSFER QUP16
LINK1 LINK 2, FIFO
QUP16 ADVANCE 2
UNLINK 2, QUP17, ALL
RELEASE P15
QUP17 QUEUE P17
ASSIGN 15, P17
PRIORITY 4

* SAMPLE GETS SPINED
*
* NOTE PARAMETER 17 IS EITHER 3 FOR NONELEC OR 4 FOR ELEC
PRIORITY 3, BUFFER
SEIZE P17
DEPART P17
ADVANCE FN72
TEST E P3, K1, NOTE+1
NOTE SAVEVALUE 164, 1
TEST G V2, KO, NOBG1
ADVANCE 5
SPLIT 1, WBPR
NOBG1 TEST E P11, KO, BABY
TRANSFER BOTH, INCEN, INUSE
INCEN GATE LR P24
TRANSFER BOTH, ENTP7, INUS1

NO. OF WAIT RACK ELEC CENT
NO. OF SWITCH PREVENTING SAMP FROM LET ANY OTHER IN SYSTEM SEIZE TEC@S SEIZE ANY ONE OF 3 TEC
TEC WHO TIME PUNCHES PAR 15 = NO. OF TEC SEIZED
NON ELEC SPINNER PAR 15 = NO. OF TEC SEIZED
ELECTROLYTE SPINNER PAR 15 = NO. OF TEC SEIZED
WAIT HERE TIL TEC FREE
TIME TO BRING TO SPINNING BENCH
TAKE ALL WAITING IN LINK TO BENCH
TEC SEIZED
ENTER EITHER QUEUE NO. 4 OF 3
PAR 15 = NO. OF SPIN TEC

NOT PARAMETER 17 IS EITHER 3 FOR NONELEC OR 4 FOR ELEC
TREAT AS SET
SEIZE EITHER ELEC OR NON ELEC TEC
EITHER ELEC OR NONELEC QUE
TIME TO MAKE SEPARATE TUBES, LOG IN
KEEP TRACK OF HOW MANY ELEC ARE LOGGED
NUMBER OF ELEC LOGGED
FAIL GO TO NO BUN OR GLU
YES, THAN TIME TO MAKE WB TUBE
SEND ONE TUBE TO WHOLE BLOOD
TEST FAIL GO TO MICROTUBE
IS CENTRIFUGE IN USE
IF CENT IS SPINNING DONOT ALLOW SAMP ENTE
WILL SAMP FIT IN CENT
YES, THEN ENTER CEN ALL TUBES OF SAMP TIME TO ENTER CENT.
IS CENTRIFUGE FULL
IS ONLY ONE SPINNER WORKING
YES, SPIN NON ELEC SAMP IN WITH ELEC SAMP

ELECT SPINNER NOW HANDELS THESE AMP SPIN THESE IN CENT
SPINNING TIME
SPINNER MUST UNLOAD BEFORE ANYTHING ELSE
SEIZE PROPER SPINNER -ELEC OR NON ELEC
LET SAMP ENTER CENT

SEIZE PROPER SPINNER -ELEC OR NON ELEC
LEAVE CEN NO TUBES WITH SAMPLE
INITIAL NUMBER OF READY RACK
ENTER PROPER READY RACK
TIME TO PUT SAMP IN RACK

IS AN ELECTROLYTE TEST REQUESTED
PART OF SAMPLE FOR NA,K,CL.

IS URIC ACID REQUESTED
PART OF SAMPLE FOR URIC ACID

IS IT AFTER 1200 YES - DO TOMORROW
IS IT AFTER 11.50 O'CLOCK

IS READY RACK FULL
YES, TIME TO BRING RACK TO BENCH
MAKE SAMP LEAVE PROPER WAIT PLACE
IS ONLY ONE SPINNER WORKING
SEND ALL SAMP IN NON ELEC READY RACK TO B
SEND ALL SAMP IN RACK TO BENCH

MAKE NEW READY RACK
SAVEVALUE P17, P13
ATBEN TRANSFER ,TERM
DTOMO TRANSFER ,TERM
BABY QUEUE 12
RELEASE P17
TERM TERMINATE

* CENTRIFUGE IN USE

INUSE1 RELEASE P17
PRIORITY 6
TEST E V21, KO

* ENTER P18, P12
TRANSFER , WAI17+2
INUSE ADVANCE FN50
PRIORITY 6
WAI17 ENTER P18, P12
RELEASE P17
GATE SE P17
PRIORITY 5, BUFFER

* SEIZE P17
LEAVE P18, P12
TRANSFER , INCEN-1

* CENTRIFUGE NOT FULL

NOFUL TEST E V5, KO, WATE
TRANSFER , SPIN-1
WATE RELEASE P17
LINK P17, FIFO

* IS READY RACK NOT FULL

RNOFU RELEASE P17

VALUE OF PROPER READY RACK
PART OF SAMP LEFT OVER
DO THESE TOMORROW
LINE OF MICROTES
RELEASE PROPER SPINNER
TERM ONE BUT DON'T HAVE IT COUNT IN TERM!

SPINNER

WAIT HERE TIL TEC TRIES ALL ENTER SAMP
CENT
PROPER WAIT RACK FOR CENT

TIME TO PREPARE WAITING RACK
TO GO WITH BUFFER BELOW
SPINNER
PROPER WAIT RACK FOR CENT
WAIT FOR CENT EMPTY
ALL REACH THIS POINT
HIGH PRIORITY TO SEIZE TEC,
SPINNER
PROPER SPINNER
PROPER WAIT RACK FOR CENT

ARE SAMPLES WAITING
START CENTRIFUGE
SPINNER
WAIT TILL CENT FULL OR NO SAMP
ASSIGN 15+,5
LINK P15,FIFO
* READY TRANSFER ,READY+1
     ADVANCE 10
     TRANSFER ,ATBEN
* INITIALIZE GLUCOSE AND UREA
* WBPR ADVANCE FN73 TIME SPENT WAIT AT SPIN BENCH TO GO TO WB
     TRANSFER ,WBPR1
WBLDB DEPART 2
* NOTE ONLY GET A LOGGER THAT IS NOT DOING ANY IMPORTANT SINCE PRI 3
WBPR1 QUEUE 13
     TRANSFER BOTH,BELOG,CNLOG LINE OF WHOLE BLOOD AT BENCH
     PRIORITY 6
     SEIZE X1
     ASSIGN 15,X1
     TRANSFER ,DEPAR LOGING TEC
     PAR. 15 = NO. OF LOGGING TEC
BELOG GATE LR 3
     TRANSFER ALL,GLBAK,BNPIP,3 ADMIT ONLY THE BEGIN LOG SAMP.
     SEIZE ANY ONE OF 4 TEC
GLBAK SEIZE 7
     ASSIGN 15,7
     TRANSFER ,CNTN1 GLUCOSE TEC WHO AUTO CLAVES
     PAR. 15 NO. OF TEC SEIZED
BNLOG SEIZE 9
     ASSIGN 15,9
     TRANSFER ,CNTN1 BUN TEC WHO NORMALLY LOGS
     PAR. 15 NO. OF TEC SEIZED
GLPIP SEIZE 8
     ASSIGN 15,8
     TRANSFER ,CNTN1 GLUTEC WHO NORMALLY PIPETS
     PAR. 15 NO. OF TEC SEIZED
BNPIP SEIZE 10
     ASSIGN 15,10
     TRANSFER ,CNTN1 BUN TEC WHO NORMALLY PIPETS
     PAR. 15 NO. OF TEC SEIZED
CNTN1 SAVEVALUE 1,P15
     LOGICS 3
     LOGICS 4
     DEPAR DEPART 13 DEPART WHOLE BLOOD QUE AT BENCH
PRIORITY 2, BUFFER
TEST E P1,K1,UREA
ADVANCE 5
RELEASE P15
TEST E P2,K1,LGLIN
TEST GE V10,K5040,URQUE
TEST G Q14,K54,LGLIN
TEST L Q15,K15,LGLIN
TRANSFER , URQUE
LGLIN TEST E Q13,K0,NOEM1
LOGICR 3
LOGICR 4

* GLUCOSE TEST *

NOEM1 TEST GE V10,K5030
QUEUE 14
GATE LR 24, SEIZE
LOGICS 24
TEST LE V10,K7200, SEIZE
SPLIT 6, COTGL
TRANSFER , SEIZE
COTGL ASSIGN 22,5
QUEUE 14
SEIZE 8
DEPART 14
ADVANCE 10
TRANSFER , NOBN1
SEIZE 8
DEPART 14
ADVANCE 6
ASSIGN 20,25
TEST NE P22,K5,NOBN1
TEST E P2,K1,NOBN1
TEST NE P19,K20,NOBN1
SPLIT 1, URQUE
NOBN1 ENTER 15,2

TEST IF ONLY UREA TEST REQ.
TIME TO LOG IN GLU BOOK AND MAKE DECISION
RELEASE LOGGING TEC
TEST TO SEE IF UREA TEST REQUESTED
IF BEFORE 8-40 UREA TEC GETS FIRST
IS GLUCOSE LINE LONG
IS UREA LINE SHORT
IF SO, SEND TO UREA PEOPLE FIRST
HAVE LOGGING LINE EMPTY
IF LOGGING LINE EMPTY
HAVE NEXT GROUP OF SAMP SEIZE PROPER TEC.
LET REST OF NEXT GROUP SEIZE SAME TEC.

GLU TEC READY FOR DAY
WAIT IN LINE TO BE PIPETED
MUST MAKE STANDARDS
BUT ONLY WITH EACH INCUBATION
IF AFTER 12 NO CONTROLS WITH EVERY ENCUBE
6 STANDARDS - 5 AND 1 BLANK
PAR 22 = 5 IF SAMP IS A CONTROL FOR GLU
LINE TO BE PIPETED
GLU TEC WHO PIPETS
LINE TO BE PIPETED
TIME TO PREPARE STANDARDS
GLUCOSE TEC WHO PIPETS
LEAVE PIPETING LINE
TIME DECISION IF GLU, PIPET AND PLACE IN
NOTE THAT SAMP BEEN PROCESSED BY GLU
IS THIS A STANDARD -YES GO TO STIRRER
IS UREA TEST REQUESTED
HAS SAMP ALREADY GONE TO UREA
REMAINING SAMPLES TO UREA LINE
TAKE UP TWO SPACES IN STIRRER (DUPLICATE)
GATE SNF 15, YESF1
TEST NE V8, KO, YESF1
RELEASE 8
ASSIGN 15, 2
LINK 1, FIFO
* YESF1 UNLINK 1, ASS50, ALL
ASS50 ADVANCE 15
* TEST NE P15, K2, RELA1
RELEASE 8
RELA1 LEAVE 15, 2
TRANSFER BOTH, INCN1, WAIT1
WAIT1 ENTER 17, 2
PRIORITY 5
GATE SE 16
PRIORITY 2, BUFFER
LEAVE 17, 2
TRANSFER RELA1 + 1, INCN1
INCN1 ENTER 16, 2
GATE SNF 16, SPINT
TRANSFER BOTH, TES21, TES22
TES21 TEST E V17, KO
TRANSFER SPINT
TES22 GATE SF 16
SPINT ADVANCE 40
PRIORITY 4
TEST GE V10, K5550
SEIZE 7
LEAVE 16, 2
ADVANCE 3
ASSIGN 13, X2
ENTER X2, 2
ASSIGN 26, P13
PRIORITY 5
TEST NE P16, K5, INCUB
GATE SNF P13, INCUB
TEST G V10, K7200, RELEA

IS SHAKER FULL
ANY SAMP IN WB OR PIPITING LINE
GLU TEC WHO PIPETS
NOTE THESE SAMP RELEASED TEC
WAIT ON USER CHAIN 1 UNTIL SHAKER
FULL OR NO MORE IN SYSTEM
TAKE ALL SAMP OFF USER CHAIN 1
TIME TO ADD ZINK TO ALL SAMP IN STIRRER
AND PLACED IN CENT
DID SAMP ALREADY RELEASE TEC NO 8
RELEASE GLU TEC THAT PIPETS
LEAVE STIRER
IS CENTRIFUGE FILLED
WAITING PLACE FOR GLU CENTRIFUGE EMPTY
HAVE ALL TRANS LEAVE CENT TOGETHER
WAIT UNTIL CENTRIFUGE IS EMPTY
TECHNIQUE USED TO TREAT AS SET
LEAVE WAITING PLACE
DUPLICATES ENTER CENT.
IS CENTRIFUGE FULL
ANY MORE OT BE PROCESSED
WAIT UNTIL GLU CENT FULL
TIME TO SPIN
HIGHER PRIOR. THAN WHEN LOGGED
WAIT ARRIVAL OF GLU TEC THAT AUTOCLAVES
SEIZE GLU TEC WHO AUTO CLAVES
LEAVE CENTRIFUGE
TIME TO PIPE AND MIX COPPER
NO OF INCUBATE RACK NOW IN USE
ENTER RACK WAIT TO INCUBATE
PAR 26 = NO OF INCUB RACK ENTERED
DON@T INTERRUPT PREP FOR INCUBATION
TEST FOR EMERGENCY
IS INCUBATE RACK FULL
IS IT AFTER 12 O'CLOCK
TEST NE V7, KO, INCUB
RELEA RELEASE 7  GLU TEC WHO AUTOCLAVES
GATE LS 5  WAIT TO INCUBATE
ASSIGN 15, 4  MAKE NOTE SAMP RELEASED TEC.
TRANSFER , MARLE
INCUB LOGICS 5  READY TO INCUBATE ALL SAMPLES IN RACK
MARLE ADVANCE V9  MAKE TEC MAKE NEW CONTROLS FOR NEXT INCUB
* 
LOGICR 5  TOTAL TIME TO PUT MARBLES ON ALL SAMP. IN RACK
ASSIGN 13+, K1  PREVENT FUTURE SAMPLES FROM INCUBATING
SAVEVALUE 2, P13  MAKE NEW INCUBATE RACK
TEST NE P15, K4, ADVA  MAKE NEW INCUBATE RACK
RELEASE 7  HAS SAMP RELEASED TEC
ADVA ENTER 29  GLU TEC WHO AUTOCLAVES
ADVANCE 250  AUTOCLAVE
SEIZE 7  AUTOCLAVE
LEAVE 29  TOTAL TIME TO INCUBATE
ADVANCE FN51  TIME TO REMOVE MARBLES, COLD WATER BATH, AD REAGENT AND BE SHAKEN
* 
SEIZE 11  COLORIMETER FOR GLUCOSE
GATE LR 22, READG  FIRST SAMP CAUSES CALIBRATION
LOGICS 22  MAKE NEXT SAMP JUST BE READ AND NOT CALIB
ADVANCE 50  TIME TO CALIBRATE INSTRUMENT
READG ADVANCE 4  TIME TO READ A SAMPLE AND RECORD RESULT
LEAVE P26, 2  INCUBATE RACK
RELEASE 11  GLU COLORIMETER
RELEASE 7  GLU TEC WHO AUTOCLAVES,
TEST E S*13, KO, ASCOM  HAVE ALL BEEN READ
LOGICR 22  MAKE TEC RECALIBRATE COLORIMETER FOR NEXT
ASCOM ASSIGN 16, 4  PAR 16 = PRIORITY OF SAMP
ASSIGN 21, 1  TIME FOR COMPUTER TO COMPUT RESULT OF THIS TEXT
TRANSFER , COMPT  GO TO THE COMPUTER

* 
* 
UREA
UREA ADVANCE FN51 TIME TO MAKE DECISION
RELEASE P15 RELEASE LOGGING TEC
UREQA QUEUE 16 UREA QUEUE

* NOTE PRIORITY OF TRANSACTION IS 3
UREAB TRANSFER BOTH,BEULO,COULO BEGIN LOG OR CONT LOG
COULO GATE LS 6 MAKE NEXT SAMP IN BATCH SEIZE SAME TEC
PRIORITY 5 DONAT INTERRUPT TEC
SEIZE X5 PROPER UREA TEC
ASSIGN 14,X5 PAR 14 NO. OF LOG TEC SEIZED
TRANSFER ,DEPAU

BEULO GATE LR 11 FIRST IN SET GO HERE
TRANSFER BOTH,TEC1A,TEC2A

TEC1A SEIZE 9 UREA TEC NORMALLY LOGS
ASSIGN 14,9 NO. OF TEC SEIZE
TRANSFER ,CONT2

TEC2A SEIZE 10 UREA TEC NORMALLY PIPETS
ASSIGN 14,10 NO. OF TEC SEIZE
CONT2 SAVEVALUE 5,P14 SAVE VALUE OF UREA TEC, SEIZED
LOGICS 6 LET REST GO TO CONTINUE LOG
LOGICS 11 MAKE REST GO TO CONTINUE LOG

DEPAU DEPART 16 UREA QUE
ADVANCE 3 TIME TO GET LOGGED IN
ENTER 30.2 RACK WAITING TO BE PIPETED
TEST E Q16,K0,REP14 ARE THERE ANY MORE SAMPLES TO LOG
LOGICR 6 LET FUTURE BATCHES SEIZE FREE TEC
LOGICR 11 LET FIRST OF NEXT SET SEIZE FREE TEC

P14 RELEASE P14 RELEASE TEC SEIZED

* * UREA GET PIPETED *

GATE LR 25,URGPI MAKE CONTROLS FOR UREA
LOGICS 25 BUT ONLY 2 CONTROLS WITH EACH WHEEL
TEST LE V10,K6300,URGPI AFTER 10.30 DONAT MAKE STANDARDS EVERY 3 WHEE
SPLIT 2,COTUR 2 CONTROLS
TEST E X13,K0,URGPI ARE STANDARDS NEEDED
SAVEVALUE 13,1 YES
SPLIT 6,STAND MAKE 6 STANDARDS WITH EVERY 3 RD WHEEL
TRANSFER URGPI
STAND ASSIGN 22,7
TRANSFER COTUR+1
COTUR ASSIGN 22,6
PRIORITY 5
SEIZE X6
ASSIGN 14,X6
ADVANCE 10
TRANSFER DEPAP+1
URGPI PRIORITY 1
TRANSFER BOTH, BEUPI, COUPI BEGIN PIPETING OR CONTINUE PIPING
COUPI GATE LS 13
PRIORITY 4
SEIZE X6
ASSIGN 14,X6
TRANSFER DEPAP
BEUPI GATE LR 12
TRANSFER BOTH, TEC1P, TEC2P ATTEMPT TO SEIZE EITHER OF TWO TEC
TEC1P SEIZE 10
ASSIGN 14,10
TRANSFER CONT3
TEC2P SEIZE 9
ASSIGN 14,9
CONT3 SAVEVALUE 6,P14
LOGICR 13
LOGICR 12
DEPAP LEAVE 30,2
ADVANCE 10
TEST E S30,K0,ADVCO IS WAITING RACK FOR PIPETING EMPTY
LOGICR 13
LOGICR 12
ADVCO TEST E P1,K1,WHEEL TEST FOR GLU
TEST NE P20,K25,WHEEL HAS SAMP BEEN PROCESSED BY GLU TEC.
TEST NE P22,K7,WHEEL IS THIS A STANDARD - YES GO TO WHEEL
TEST NE P22,K6,WHEEL IS THIS A CONTROLD - YES GO TO WHEEL
PRIORITY 3
ASSIGN 19,20
NOTE THAT SAM HAS GONE TO UREA

PAR 22 = 7 IF SAMP IS A UREA STANDARD
PAR 22 = 6 IF SAMP IS A UREA CONTROL
HIGH TO SEIZE TEC
SEIZE PIPETING TEC
PAR 14 = NO OF TEC SEIZED
PREPARE CONTROLS
LOW PRIORITY DON'T INTERRUPT LOGGER
MAKE NEXT SAMP IN BATCH SEIZE SAME TEC
DON'T INTERRUPT TEC SEIZED
PROPER UREA TEC
PAR 14 NO. OF READING TEC
FIRST SAMPLE GETS PIPETED
ATTEMPT TO SEIZE EITHER OF TWO TEC
UREA TEC NORMALLY PIPETS
NO. OF TEC SEIZE
UREA TEC NORMALLY LOGS
NO. OF TEC SEIZE
SAVE NO. OF UREA TEC, SEIZED
LET FUTURE BATCHES SEIZE FREE TEC,
LET FIRST OF NEXT SET SEIZE FREE TEC
WAITING RACK TO BE PIPETED
TIME TO BE PIPETED AND PUT IN WHEEL OR WAIT RACK
IS WAITING RACK FOR PIPETING EMPTY
LET REMAINING OF BATCH GET TO CONTINUE
LET FIRST OF NEXT SET SEIZE TEC
TEST FOR GLU
HAS SAMP BEEN PROCESSED BY GLU TEC.
IS THIS A STANDARD - YES GO TO WHEEL
IS THIS A CONTROLD - YES GO TO WHEEL
GIVE GLU PART-SAME PRIORITY AS OTHER GLU
NOTE THAT SAM HAS GONE TO UREA
SPLIT 1,NOEM1 SEND A PORTION TO GLU
WHEEL TRANSFER ALL,ENWH1,ENWAT,3 ENTER WHEEL 1 OR 2 OR WAITING RACK
ENWH1 ENTER 31,2 UREA WHEEL NO ENTER TWO SAMPLES
ASSIGN 15,31 NO. OF WHEEL ENTERED
TRANSFER ,WHEFU
ENTER 32,2 UREA WHEEL NO 2 TWO SPACES
ASSIGN 15,32 NO. OF WHEEL ENTERED
TRANSFER ,WHEFU
ENWAT ENTER 33,2 WAIT FOR FREE WHEEL
RELEASE P14 PIPIETING TEC
PRIORITY 5 HIGH PRIORITY SO THAT GET IN WHEEL FIRST
LINK 11,FIFO WAIT HERE FOR URIA WHEEL FREE
UNL17 SEIZE X8 TEC THAT JUST EMPITIED WHEEL
ASSIGN 14,X8 NO. OF TEC JUST SEIZED
ADVANCE 3 TIME TO GET PLACED IN WHEEL
LEAVE 33,2 WAITING RACK FOR WHEEL
TRANSFER ,WHEEL
WHEFU TEST NE P16,K5,TURN TEST FOR EMERGENCIES
GATE SNF P15,TURN IS WHEEL FULL
TEST G V10,K7200,RELE1 NO IS IT AFTER 12 O'CLOCK
TEST NE V12,K0,TURN ANY MORE IN SYSTEM
RELE1 RELEASE P14 TEC THAT PIPIET
GATE LS P15 WAIT TO TURN IN PROPER WHEEL
ASSIGN 17,10 NOTE SAMP. RELEASED P P TEC
TRANSFER ,TURN2
TURN LOGICS P15 LET REST IN THIS WHEEL TURN
LOGICCR 25 MAKE TEC MAKE CONTROLS FOR NEST WHEEL
SAVEVALUE 13+,1 NOTING ONE WHEEL IS COMPLETE
TEST E X13,K4,TURN2 ARE STANDARDS NEEDED
SAVEVALUE 13,K0 YES, MAKE TEC MAKE STANDARDS WITH EVERY 3 RD WHEEL
TURN2 PRIORITY 0,BUFFER WAIT FOR ALL BEFORE CONTINUING
LOGICCR P15 MAKE INCOMING SAMP WAIT TO TURN
TEST NE P17,K10,ADV11 HAS SAMP. RELEASE PIPIETING TEC,
RELEASE P14 RELEASE PIPIETING TEC
ADV11 ADVANCE 160 TIME TO TURN ON WHEEL

* GET ALKALINIZED

★
TRANSFER BOTH,BESQU,COSQU BEGIN ALKALINIZATION OR CONTINUE
COSQU GATE LS 15 MAKE NEXT SAMP IN BATCH SEIZE SAME TEC
ASSIGN 14,30 NOTE THAT SAMP DID NOT SEIZE TEC
TRANSFER ,ADVQ
BESQU GATE LR 14 HAVE FIRST SAMPLE ENTER HERE
TRANSFER BOTH,TEC1S,TEC2S
TEC1S SEIZE 9 UREA TEC NORMALLY LOGS
ASSIGN 14,9 NO. OF TEC SEIZE
TRANSFER ,CONT4
TEC2S SEIZE 10 UREA TEC NORMALLY PIPETS
ASSIGN 14,10 NO. OF TEC SEIZE
CONT4 SAVEVALUE 7,P14 SAVE NO. OF UREA TEC SEIZED
LOGICS 15 LET REST GO TO CONTINUE ALKALIN
LOGICS 14 MAKE REST GO TO CONTINUE ALKALIN
ADVQ ADVANCE V13 TOTAL TIME TO GET ALKALI. IN ALL SAMP
LOGICR 15 LET FUTURE PATCHES SEIZE FREE TEC.
LOGICR 14 LET FIRST OF NEXT SET SEIZE FREE TEC
TEST NE P14,K30,NOTSZ DID SAMP SEIZE TEC
RELEASE P14 RELEASE TEC SEIZED
NOTSZ ADVANCE 250 TURNING TIME

* GET TAKEN OUT OF WHEEL
*

PRIORITY 6 MUST EMPTY WHEEL
TRANSFER BOTH,BETKO,COTKO TAKE OUT OF WHEEL
COTKO GATE LS 17 MAKE NEXT SAMP IN BATCH SEIZE SAME TEC.
ASSIGN 14,34 NOTE THAT SAMP DID NOT SEIZE TEC
TRANSFER ,ADVTK
BETKO GATE LR 16 HAVE FIRST SAMPLE ENTER HERE
TRANSFER BOTH,TEC1T,TEC2T SEIZE EITHER OF 2 TEC
TEC1T SEIZE 9 TEC NORMALLY LOGS
ASSIGN 14,9 NO. OF TEC SEIZE
TRANSFER ,CONT5
TEC2T SEIZE 10 UREA TEC NORMALLY PIPETS
ASSIGN 14,10 NO. OF TEC SEIZED
CONT5 SAVEVALUE 8,P14 SAVE NO. OF UREA TEC SEIZED
LOGICS 17 LET REST GO TO CONTINUE TKO
LOGICS 16 MAKE REST GO TO CONTINUE TKO
ADVTK LEAVE P15,2 LEAVE WHEEL ENTERED
ADVANCE V14 TOTAL TIME TO TAKE OUT ALL SAMPLES
LOGICR 17 MAKE NEXT XACTO OF NEXT SET NOT SEIZE TEC
LOGICR 16 LET FIRST OF NEXT SET SEIZE TEC
UNLINK 11,UNL17,ALL LET SAMP WAIT FOR WHEEL ENTER WHEEL
ENTER 34,2 NESSLER BOTTLE RACK
TEST NE P14,K34,NOSZ1 DID SAMP SEIZE TEC
RELEASE P14 RELEASE TEC SEIZED
NOSZ1 ADVANCE 50 WAIT AT LEAST 5 MIN BEFORE PROCEEDING

* * GET READ ON COLORIMETER *
* *
TRANSFER BOTH,BERD,CORD BEGIN TO GET READ OR CONTINUE READ
CORD GATE LS 19 MAKE NEXT SAMP IN BATCH SEIZE SAME TEC.
PRIORITY 6 DON'T INTERRUPT TEC
SEIZE X9 PROPER UREA TEC
SEIZE 12 COLORIMETER UREA
ASSIGN 14,X9 PAR 14 NO. OF READING TEC
TRANSFER ,ADVRD
BERD GATE LR 18 HAVE FIRST SAMPLE ENTER HERE
TRANSFER BOTH,TEC1R,TEC2R SEIZE FREE TEC.
TEC1R SEIZE 9 UREA TEC NORMALLY LOGS
ASSIGN 14,9 NO. OF TEC SEIZED
TRANSFER ,CONT6
TEC2R SEIZE 10 UREA TEC NORMALLY PIPETS
ASSIGN 14,10 NO. OF TEC SEIZED
CONT6 SAVEVALUE 9,P14 SAVE NO. OF UREA TEC, SEIZED
LOGICS 19 LET REST GO TO CONTINUE READ
LOGICS 18 MAKE REST GO TO CONTINUE READ
SEIZE 12 COLORIMETER FOR UREA
ADVANCE 20 TIME TO CALIBRATE INSTRUMENT
ADVRD ADVANCE 2 TIME TO READ A SAMPLE - NOT DUPLICATE
TEST E S34,K0,REL14 IS ARE MORE MORE SAMP OT BE READ
LOGICR 19 LET FUTURE BATCHES SEIZE FREE TEC
LOGICR 18 LET FIRST OF NEXT SET SEIZE FREE TEC
REL14 RELEASE P14 RELEASE TEC SEIZED
RELEASE 12 COLORIMETER UREA
ASSIGN 16,4, PAR 16 = PRIORITY OF SAMP
ASSIGN 21,1, TIME FOR COMPUTER TO COMPUT RESULT OF THIS TEXT
TRANSFER ,COMPT, GO TO THE COMPUTER

* * * LUNCH ROUTINE TO SIMULATE TEC LUNCH PERIOD * * *

GENERATE 0,0,7200,1,6, SIGNAL FROM CONTROL SYST. TO SIGNIFY LUNCH -PR 6
SPLIT 1,LUSE7
SEIZE 3, NON ELECT SPINNER - SEIZE WHEN AVAILABLE
ADVANCE 30, TIME FOR LUNCH
RELEASE 3, NON ELECT SPINNER
TRANSFER ,LUADV

LUSE7 SPLIT 1,LUSE9
SEIZE 7, GLU TEC WHO NORMALLY INCUBATES
ADVANCE 30, TIME FOR LUNCH
RELEASE 7, GLU TEC WHO NORMALLY INCUBATES
TRANSFER ,LUADV

LUSE9 SPLIT 1,LUS13
SEIZE 9, UREA TEC WHO NORMALLY LOGGING
ADVANCE 30, TIME FOR LUNCH
RELEASE 9, UREA TEC WHO NORMALLY LOGGING
TRANSFER ,LUADV

LUS13 SPLIT 1,LUS16
SEIZE 13, CHLORIDE TEC
ADVANCE 30, TIME FOR LUNCH
RELEASE 13, CHLORIDE TEC
TRANSFER ,LUADV

LUS16 SEIZE 16, BICARBONATE TEC WHO TITRATES
ADVANCE 30, TIME FOR LUNCH
RELEASE 16, BICARBONATE TEC WHO TITRATES
TRANSFER ,LUADV
GENERATE 0,0,7500,1,6, HUNGER SECOND PERIOD
SPLIT 1,LUSE8
SEIZE 4, ELECTROLITE TEC
ADVANCE 30, TIME FOR LUNCH
<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELEASE</td>
<td>4</td>
<td>ELECTROLITE TEC</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>,LUADV</td>
<td></td>
</tr>
<tr>
<td>LUSE8</td>
<td>SPLIT 1,LUS10</td>
<td>GLU TEC WHO NORMALLY PIPETS</td>
</tr>
<tr>
<td>SEIZE</td>
<td>8</td>
<td>TIME FOR LUNCH</td>
</tr>
<tr>
<td>RELEASE</td>
<td>8</td>
<td>GLU TEC WHO NORMALLY PIPETS</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>,LUADV</td>
<td></td>
</tr>
<tr>
<td>LUS10</td>
<td>SPLIT 1,LUS20</td>
<td>UREA TEC WHO NORMALLY PIPETS</td>
</tr>
<tr>
<td>SEIZE</td>
<td>10</td>
<td>TIME FOR LUNCH</td>
</tr>
<tr>
<td>RELEASE</td>
<td>10</td>
<td>UREA TEC WHO NORMALLY PIPETS</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>,LUADV</td>
<td></td>
</tr>
<tr>
<td>LUS20</td>
<td>SEIZE 20</td>
<td>BICARBONATE TEC WHO PIPETS AND BRINGS TO BENCH</td>
</tr>
<tr>
<td>ADVANCE</td>
<td>30</td>
<td>TIME FOR LUNCH</td>
</tr>
<tr>
<td>RELEASE</td>
<td>20</td>
<td>BICARBONATE TEC WHO PIPETS AND BRINGS TO BENCH</td>
</tr>
<tr>
<td>LUADV</td>
<td>TRANSFER ,TERM</td>
<td></td>
</tr>
<tr>
<td>**</td>
<td>**</td>
<td>**URIC ACID</td>
</tr>
<tr>
<td>**</td>
<td>**</td>
<td>**GENERATE 0,0,6600,31,0,30 SAMPS LEFT OVER FROM PREVIOUS DAY</td>
</tr>
<tr>
<td>URIAC</td>
<td>SAVEVALUE 15+,1</td>
<td>SAVE NO. OF URIC ACID SAMPLES</td>
</tr>
<tr>
<td>LINK</td>
<td>5,FIFO,TES11</td>
<td>FIRST Samp TEST 11, NEXT GOES ON CHAIN 6</td>
</tr>
<tr>
<td>TES11</td>
<td>TEST GE V10,K6600</td>
<td>IS IT PAST 11 OCLOCK—NO, THEN WAIT</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>3</td>
<td>DON'T INTERUPT SPINNER IF BUSY</td>
</tr>
<tr>
<td>SEIZE</td>
<td>3</td>
<td>SEIZE THE NON ELEC SPINNER WHO DOES THE URIC ACID</td>
</tr>
<tr>
<td>ADVANCE</td>
<td>30</td>
<td>AVG TIME TO FIND AND COLLECT ALL U.A. SAMP</td>
</tr>
<tr>
<td>RELEASE</td>
<td>3</td>
<td>RELEASE THE NON ELEC SPINNER</td>
</tr>
<tr>
<td>UNLINK</td>
<td>5,ENTUA,ALL AKE SAMP OFF THE USER CHAIN 6 AND SEND IT TO</td>
<td></td>
</tr>
<tr>
<td>ENTUA</td>
<td>ENTER 45</td>
<td>URIC ACID RACK WAITING TO START PROCESSING</td>
</tr>
<tr>
<td>SAVEVALUE</td>
<td>15-,1</td>
<td>NO. OF SAMPLES YET TO FIND</td>
</tr>
<tr>
<td>TEST E</td>
<td>X15,K0</td>
<td>ARE ALL URIC ACID SAMPLES COLLECTED</td>
</tr>
<tr>
<td>SEIZ3</td>
<td>SEIZE 3</td>
<td>URIC ACID TEC</td>
</tr>
<tr>
<td>ADVANCE</td>
<td>10</td>
<td>TIME TO PIPET AND MIX PLACE IN RACK</td>
</tr>
<tr>
<td>LEAVE</td>
<td>45</td>
<td>URIC ACID RACK</td>
</tr>
<tr>
<td>RELEASE</td>
<td>3</td>
<td>URIC ACID TEC</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>2,BUFFER</td>
<td></td>
</tr>
</tbody>
</table>

* * *
ENTER 45        URIC ACID RACK
SEIZE 3        URIC ACID TEC.
ADVANCE 10     TIME TO PIPE T
LEAVE 45       URIC ACID RACK
RELEASE 3      URIC ACID TEC.
GATE SE 45     WAIT UNTIL ALL ARE DONE
ENTER 62       URIC ACID CENT
TRANSFER BOTH, SEZ12, ADSUA LET BATCH SEIZE TEC
SEZ12 SEIZE 12 CENT FOR URIC ACID
SEIZE 3        SEIZE URIC ACID TEC
ADVANCE 300     TIME TO SPIN AND LOAD AND UNLOAD
RELEASE 3      RELEASE U.A. TEC
RELEASE 12     CENT URIC ACID
TRANSFER , SE133
ADSUA ADVANCE 300 TIME TO SPIN BATCH - WHILE SPIN THE TEC IS
* PREPARING RACK ADDING CARBONATE TO RECEPTOR TEST T U
SE133 SEIZE 3   URIC ACID TEC
ADVANCE 7      TIME TO PIPE T ADD CARB AND SHAKE
ENTER 46,2     URIC ACID RECEPTOR RACK
LEAVE 62       URIC ACID CENT
RELEASE 3      RELEASE URIC ACID TEC.
GATE SE 62     WAIT UNTIL ALL ARE TAKEN OUT OF CENT
SEIZE 3        SEIZE URIC ACID TEC
ADVANCE 7      TIME TO ADD COLOR REAGENT
LEAVE 46,2     URIC ACID RECEPTOR RACK
RELEASE 3      RELEASE URIC ACID TEC
GATE SE 46     WAIT UNTIL ALL ARE PROCESSED
MARK           SET TRANSIT TIME TO ZERO
ADVANCE 300     SIT FOR HALF HOUR
SEIZE 19       COLORIMETER
GATE LR 20, SEZ31 LET FIRST SAMPLE TEST TIME
TEST LE  M1,K4500,PANIC  HAS IT BEEN WAITING MORE THAN 45 MINUTES
LOGICS 20     PREVENT FUTURE SAMP FROM TEST TIME
SEZ31 SEIZE 3  SEIZE URIC ACID TEC
ADVANCE 4      TIME TO READ
RELEASE 19     COLORIMETER
RELEASE 3      RELEASE URIC ACID TEC
ASSIGN 16,4  PAR 16 = PRIORITY OF SAMP
ASSIGN 21,1 TIME FOR COMPUTER TO COMPUT RESULT OF THIS TEXT
TRANSFER ,COMPT GO TO THE COMPUTER

PANIC TRANSFER ,TERM
*PANIC MEANS SAMPLE HAS BEEN WAITING MORE THAN 45 MINUTES
*
*
** ELECTROLYTES - SODIUM - POTASSIUM - CHLORIDE **
**

NAPCL ENTER 48 SODIUM POTASSIUM AND CHLORIDE
PRIORITY 5 MUST COLLECT ALL SAMP BEFORE CONTINUE
TEST GE V10,K5120 NA K CL TEC ARRIVE AT 0.32
SEIZE 14 SODIUM POTASSIUM TEC
ADVANCE 5 TIME TO FIND AND COLLECT A SAMPLE FOR CL, NA, P
LEAVE 48 SODIUM POTASSIUM AND CHLORIDE
RELEASE 14 SODIUM POTASSIUM TEC
GATE SE 48 WAIT UNTIL ALL ARE COLLECTED IN THIS SET
SPLIT 1,BICBE SEND TUBE TO BARCARBONATE TEST
SPLIT 1,CHLOR SEND PART OF SAMPLE TO CHLORIDE RACK
ENTER 49 SODIUM AND POTASSIUM RACK AT BENCH
PRIORITY 3,BUFFER HAVE ALL IN SET ARRIVE AT THIS POINT
PRIORITY 6 ONCE TEC AT BENCH KEEP HIM THERE
SEIZE 14 SODIUM POTASSIUM TEC
ADVANCE 2 TIME TO PI PET AND SEND REMAINING TO CHLORIDE
LEAVE 49 SODIUM AND POTASSIUM RACK AT BENCH

** SODIUM POTASSIUM **

RELEASE 14 SODIUM POTASSIUM TEC
GATE SE 49 WAIT UNTIL ALL PI PETED
SAVEVALUE 14+1 ADD ONE TO SAVEVALUE 14
TEST LE V10,K6600, IGNOR IS IT AFTER 12 OCLOCK ,YES -GO TO IGNOR
TEST GE X14,K3 DON@T CONTINUE UNTIL AT LEAST 3 SAMP HERE
IGNOR ENTER 61,2 SODIUM AND POTASSIUM RACK AT BENCH
PRIORITY 2,BUFFER HAVE ALL IN SET ARRIVE AT THIS POINT
SAVEVALUE 14-,1 SUB ONE TO SAVEVALUE 14
SEIZE 14 SODIUM POTASSIUM TEC
GATE LR 21,READF FIRST SAMP, CAUSES CALIBRATION
LOGICS 21 MAKE NEXT SAMP JUST BE READ ON FLAME PHOTOMETER
ADVANCE 50 TIME TO CALIBRATE FLAME PHOTOMETER
READF ADVANCE 6 TIME TO READ ON FLAME PHOTOMETER
TEST NE V15,K0,RELSP ANY MORE NA P CL TO PROCESS
ADVANCE 60 TIME TEC SPENDS TO CHECK ALL ELEC RESULTS
RELSP RELEASE 14 SODIUM AND POTASSIUM TEC
LEAVE 61,2 SODIUM AND POTASSIUM RACK AT BENCH
SPLIT 1,NAK MACHINE GIVES INFORMATIN TO COMPUTER
* ABOUT BOTH SODIUM AND POTASSIUM FROM ONE SAMP
NAK TEST E S61,K0,COMPS HAVE ALL BEEN READ
LOGICR 21 MAKE TEC RECAL FLAME FOR NEXT SET
COMPS ASSIGN 16,4 PAR 16 = PRIORITY OF SAMP
ASSIGN 21,1 TIME FOR COMPUTER TO COMPUT RESULT OF THIS TEXT
TRANSFER .COMPT GO TO COMPUTER
*
*CHLORIDE
*
CHLOR ENTER 47 CHLORIDE RACK AT BENCH
TEST GE V10,K3480 TEC NOT AVAILABLE TIL 98.58
PRIORITY 0 LOW PRIORITY
TRANSFER SIM, GAT54, GAT54 SET DELAY INDICATOR TO ZERO
GAT54 GATE SE 54 NO SAMP WAITING IN WAIT RACK FOR MACHINE
GATE NU 13 DON'T CONTINUE IF TEC IS BUSY
GATE SE 50 WAIT UNTIL TEC HAS EMPTIED MACHINE
TRANSFER SIM, SEICH, GAT54 SIMULTANIOUS STORAGE MACH, WAIT MACH EMPTI
SEICH SEIZE 13 CHLORIDE TEC
LEAVE 47 CHLORIDE RACK AT BENCH
ADVANCE 2 TIME TO PIPEIT AND MAKE DUPLICATE PLACE IN MACHINE
SPLIT 1, DUPWA MAKE DUPLICATE AND PUT ASIDE
ENT50 ENTER 50,1 CHLORIDE MACHINE
RELEASE 13 CHLORIDE TEC
ADVANCE 8 TIME SPENT ON MACHINE
SEIZE 13 CHLORIDE TEC
ADVANCE 2 TIME TAKEN OUT OF MACHINE AND RECORD
<table>
<thead>
<tr>
<th>LEAVE</th>
<th>50,1</th>
<th>CHLORIDE MACHINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELEASE</td>
<td>13</td>
<td>CHLORIDE TEC</td>
</tr>
<tr>
<td>ASSIGN</td>
<td>16,4</td>
<td>PAR 16 = PRIORITY OF SAMP</td>
</tr>
<tr>
<td>ASSIGN</td>
<td>21,1</td>
<td>TIME FOR COMPUTER TO COMPUT RESULT OF THIS TEST</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>5,ENT50</td>
<td>GO TO COMPUTER</td>
</tr>
<tr>
<td>DUPWA PRIORITY</td>
<td>4</td>
<td>HIGH PRIORITY - NEXT SIEZED BY TEC</td>
</tr>
<tr>
<td>ENTER</td>
<td>54,1</td>
<td>CHLORIDE WAITING RACK FOR MACHINE TO EMPTY</td>
</tr>
<tr>
<td>GATE SE</td>
<td>50</td>
<td>WAIT UNTIL CHLORIDE MACHINE IS EMPTY</td>
</tr>
<tr>
<td>SEIZE</td>
<td>13</td>
<td>CHLORIDE TEC</td>
</tr>
<tr>
<td>LEAVE</td>
<td>54,1</td>
<td>CHLORIDE WAITING RACK FOR MACHINE TO EMPTY</td>
</tr>
<tr>
<td>ADVANCE</td>
<td>FN71</td>
<td>TIME TO GET PLACED IN MACHINE</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>50</td>
<td>ENTER MACHINE</td>
</tr>
</tbody>
</table>

**BICARBONATE**

| BICAR ENTER | 55 | POSITION IN READY RACK FOR BICARBONATE SAMP |
| PRIORITY 1 | | |
| SEIZE 20 | BICARB TEC WHO PIPETS AND BRINGS SAMP TO BENCH |
| LEAVE 55 | POSITION IN READY RACK FOR BICARBONATE SAMP |
| ADVANCE 30 | TIME TO COLLECT SAMP AND BRING TO BENCH |
| RELEASE 20 | BICARB TEC WHO PIPETS AND BRINGS SAMP TO BENCH |
| PRIORITY 0, BUFFER | LET NEXT SAMP COME TO BENCH |
| PRIORITY 2 | | |
| BICBE SEIZE 20 | BICARB TEC WHO PIPETS AND BRINGS SAMP TO BENCH |
| ADVANCE 10 | TIME TO PIPET |
| ENTER 56 | WAIT RACK FOR POSITION IN AIRATOR |
| RELEASE 20 | BICARB TEC WHO PIPETS AND BRINGS SAMP TO BENCH |
| PRIORITY 5 | | |

* PRIORITIES ARE TO INSURE PROPER ORDER OF EVENTS

<p>| GATE SNF | 57 | WAIT FOR POSITION IN AIRATOR |
| LEAVE 56 | WAIT RACK FOR POSITION IN AIRATOR |
| SEIZE 16 | BICARBONATE TEC WHO TITRATES AND READS |
| ENTER 57,2 | BICARBONATE AIRATOR SAMP NAD ITS DUPLICATE |
| ADVANCE 2 | TIME TO PUT IN AIRATOR |
| RELEASE 16 | BICARBONATE TEC WHO TITRATES AND READS |</p>
<table>
<thead>
<tr>
<th>ADVANCE</th>
<th>30</th>
<th>AIRATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEAVE</td>
<td>57,2</td>
<td>BICARBONATE AIRATOR SAMP AND ITS DUPLICATE</td>
</tr>
<tr>
<td>SEIZE</td>
<td>16</td>
<td>BICARBONATE</td>
</tr>
<tr>
<td>ADVANCE</td>
<td>1</td>
<td>TAKEN OUT OF AIRATOR</td>
</tr>
<tr>
<td>RELEASE</td>
<td>16</td>
<td>BICARBONATE TEC WHO TITRATES AND READS</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>3, BUFFER</td>
<td>ALLOW TEC TO PUT A NEW SAMP IN AIRATOR</td>
</tr>
<tr>
<td>SEIZE</td>
<td>16</td>
<td>BICARBONATE TEC WHO TITRATES AND READS</td>
</tr>
<tr>
<td>ADVANCE</td>
<td>10</td>
<td>TIME TO TITRATE AND READ</td>
</tr>
<tr>
<td>ADVANCE</td>
<td>30</td>
<td>TIME TEC TAKES TO CALCULATES RESULTS</td>
</tr>
<tr>
<td>RELEASE</td>
<td>16</td>
<td>BICARBONATE TEC WHO TITRATES AND READS</td>
</tr>
<tr>
<td>ASSIGN</td>
<td>16,4</td>
<td>PAR 16 = PRIORITY OF SAMP</td>
</tr>
<tr>
<td>ASSIGN</td>
<td>21,1</td>
<td>TIME FOR COMPUTER TO COMPUT RESULT OF THIS TEST</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>, COMPT</td>
<td>GO TO COMPUTER</td>
</tr>
</tbody>
</table>

**TOTAL PROT9 ALBUMIN GLOBIUM**

| GENERATE | 0,0,5700,13,0,30 SAMPS LEFT OVER FROM PREVIOUS DAY |
| TRANSFER | , TPGAT |
| TPRAG     | ENTER | 58 | POSITION IN READY RACK FOR PROT A G |
| TEST GE   | V10, K5390 | MUST WAIT FOR TEC TO ARRIVE IN LAB |
| PRIORITY  | 5 |
| SEIZE     | 17 | TOT PROT AND ALBUMIN GLOBIUM TEC |
| LEAVE     | 58 | POSITION IN READY RACK FOR PROT A G |
| ADVANCE   | 20 | TIME TO COLLECT SAMP AND BRING TO BENCH |
| RELEASE   | 17 | TOT PROT AND ALBUMIN GLOBIUM TEC |
| TPGAT     | GATE SE | 58 | LET TEC GET ALL SAMP |
| PRIORITY  | 4 | HIGH TO KEEP TEC AT BENCH |
| ENTER     | 59 | TOTAL PROT A B RACK AT BENCH |
| GATE LR   | 23, PIPAG | LET FIRST THROUGH HERE—MUST PREPARE NA SULFATE |
| LOGICS    | 23 | MAKE REST GO TO PIPAG |
| SEIZE     | 17 | TOT PROT AND ALBUMIN GLOBIUM TEC |
| ADVANCE   | 50 | TIME TO PREPARE SODIUM SULFATE |
| RELEASE   | 17 | TOT PROT AND ALBUMIN GLOBIUM TEC |
| PIPAG     | SEIZE | 17 | TOT PROT AND ALBUMIN GLOBIUM TEC |
LEAVE 59 TOTAL PROT A G RACK AT BENCH
ADVANCE 5 TIME TO PIPEIT INTO TUBE OF NA SULFATE-HAND PIPET
RELEASE 17 TOT PROT AND ALBUMIN GLOBIUM TEC
GATE SE 59 WAIT UNTIL ALL AT BENCH ARE PIPETED
SEIZE 17 TOT PROT AND ALBUMIN GLOBIUM TEC
ADVANCE 5 TIME TO PIPEIT PART OFF FOR TOT PROT REMAINDER FOR A
SPLIT 1,TRO SEND PART TO TOTAL PROTINE
ENTER 60 BICARBONATE FILTER RACK
RELEASE 17 TOT PROT AND ALBUMIN GLOBIUM TEC
PRIORITY 0 DON'T CONT UNTIL ALL OTHER WORK DONE
ASSIGN 15,2 NO, OF TIMES NEED TO ADD MORE SAMP TO FILTER
FILTE ADVANCE 300 BE FILTERED
SEIZE 17 TOT PROT AND ALBUMIN GLOBIUM TEC
ADVANCE 3 TIME TO ADD MORE SAMP TO SAMP IN FILTER RACK
RELEASE 17 TOT PROT AND ALBUMIN GLOBIUM TEC
LOOP 15,FILTE ADD MORE SAMP TO FILT AGAIN
TEST GE V10,K6900 WAIT UNT JUST BEFORE LUNCH
ADVANCE 300 BE FILTERED
SEIZE 17 TOT PROT AND ALBUMIN GLOBIUM TEC
LEAVE 60 BICARBONATE FILTER RACK
ADVANCE 20 TIME TO TAKE OUT OF FILTER RACK AND PEP CLEAR PORT1
RELEASE 17 TOT PROT AND ALBUMIN GLOBIUM TEC
ADVANCE 300 WAIT 1/2 HOUR
TEST E V6,K0,LIN11 ARE THERE ANY SAMP WAITING
UNLINK 10,SE119,ALL NO, THAN START READING SAMP WAITING
LIN11 LINK 12,FIFO YES, THAN WAIT HERE TO BE READ
TPRO TEST NE V6,K0,UNLTP ARE MORE SAMP STILL TO BE PROCESSED BY TEC
LINK 10,FIFO YES, WAIT HERE TO BE READ
UNLTP UNLINK 10,SE119,ALL NO, THAN ALL SAMP ARE TO BE READ
SE119 SEIZE 17 TOT PRO AG TEC
ADVANCE 5 TIME TO ADD BIURET
RELEASE 17 TOT PRO AG TEC
ADVANCE 300 WAIT 1/2 HOUR
PRIORITY 7 HIGH PRIORITY TO KEEP TEC READING
SEIZE 19 TOT PRO AG READ MACHINE
SEIZE 17 TOT PRO AG TEC
TRANSFER ,ADRR
SEITP SEIZE 19 TOT PRO AG READ MACHINE
SEIZE 17 TOT PRO AG TEC
ADRR ADVANCE 5 TIME TO READ AND RECORD
UNLINK 12,SEITP,ALL WHEN ALL TOT PRO AG READ - READ AG
RELEASE 17 TOT PROT AND ALBUMIN GLOBIUM TEC
RELEASE 19 TOT PRO AG READ MACHINE
ASSIGN 16,4 PAR 16 = PRIORITY OF SAMP
ASSIGN 21,1 TIME FOR COMPUTER TO COMPUT RESULT OF THIS TEST
TRANSFER ,COMPT GO TO COMPUTER

* * * COMPUTER COMPUTES RESULTS OF TESTS * * *

COMPT QUEUE 5 COMPUTER QUEUE WAIT FOR COMPUTE RESULTS OF TESTS
SAVEVALUE 12+,K1 ADD ONE EACH TIME SAMP ENTERS HERE
ASSIGN 16,6 PAR 16 = PRIORITY OF SAMP
PRIORITY 6 INFORMATION MUST ENTER COMP OR BE LOST
COMP2 GATE NU 15,YESCU IS COMPUTER IN USE
SEIZE 15 NO THAN SEIZE IT
SAVEVALUE 10,P16 SAVE PRIORITY OF SAMPLE
SAVEVALUE 11,X10 SAVE PRIORITY OF SAMP
DEPART 5 COMPUTER QUE FOR COMPUTING RESULTS OF TESTS
COMPU ADVANCE P21 TIME TO COMPUT AND STORE RESULT OF TEST
ADVANCE FN70 TIME TO CHECK IF ALL TEST ON SAMP ARE DONE
RELEASE 15 COMPUTER
TRANSFER ,UNL16

YESCU TEST L P16,X10,NOWAC IS PRI. LESS THAN THAT OF SAMP THAT HAS COMPUTER
LINK 6,P16 YES, WAIT HERE FOR COMP FREE - IN ORDER OF PRI
NOWAC PREEMPT 15 NO, PREEMPT COMPUTER
DEPART 5 COMPUTER QUEUE FOR COMPUTING RESULTS OF TESTS
SAVEVALUE 10,P16 SAVE VALUE OF PRI IN SAVEVALUE 10
ADVANCE 1 TIME TO UNLOAD CORE
ADVANCE P21 TIME TO COMPUTE AND STORE RESULT OF TEST
ADVANCE FN70 TIME TO CHECK IF ALL TEST ON SAMP ARE DONE
RETURN 15 RELEASE COMPUTER
SAVEVALUE 10,X11 RESTORE PRIORITY OF PREEMPTED SAMP
UNLI 6,COMP2,1 SEND WAITING SAMP TO COMPUTER FOR PROCESSING
PRIORITY 5 HIGH PRI MUST PRINT REPORT WHEN READY
TEST NE V3,K0,TERM DOES MODEL PROCESS THIS SAMP — CONTINUE IF IT DOES
ASSEMBLE V3 WAIT HERE TIL ALL TEST OF SAMP ARE DONE

** COMPUTER PRINTS PATIENT REPORT **

QUEUE 6 COMPUTER Q WAITING FOR PRINTSAMP REPORT
SEIZE 15 COMPUTER
SAVEVALUE 10,K4 ONLY LET COMP BE INTURUPT TO ACCEPT RESULTS OF TEST
ADVANCE V4 TIME TO PRINT PATIENT REPORT
RELEASE 15 COMPUTER

** COMPUTER COMPUTES AND PRINTS WARD REPORT **

PRIORITY 3 WAIT TO PRINT WARD REPORT
QUEUE 7 LET FIRST SAMP THROUGH — REST GO TO TERM
GATE LR 22,TERM PREVENT REST FROM COMMING HERE
LOGICS 22 TEST G V10,9600 IS IT AFTER 4 OCLOCK
SEIZE 15 YES, THAN SEIZE COMPUTER
DEPART 7 WAIT TO PRINT WARD REPORT
ADVANCE V19 TIME TO PRINT WARD REPORTS
RELEASE 15 COMPUTER
PRIORITY 1,BUFFER LET COMPUTER BE SEIZED BY ANY WAITING SAMP

** COMPUTER COMPUTES AND PRINTS SECTION REPORT **

QUEUE 8 COMPUTER QUE WAIT TO PRINT SECTION REPORT
SEIZE 15 COMPUTER
DEPART 8 COMPUTER QUE WAIT TO PRINT SECTION REPORT
ADVANCE V19 TIME TO PRINT SECTION REPORT
RELEASE 15 COMPUTER
TRANSFER ,TERM

** DOCTOR NEEDS REPORT BEFORE ALL TEST ON SAMP ARE DONE **

GENERATE 300,100,4800,,6 REQUEST FOR REPORTS — ONE EVERY 1/2 HOUR
QUEUE 9  DOCTOR REQUEST FOR PARTIAL REPORT QUEUE
GATE NU 15, PRE1C  IS COMPUTER BUSY
SEIZE 15  NO, THAN SEIZE IT
DEPART 9  DOCTOR REQUEST FOR PARTIAL REPORT QUEUE
ADVANCE 1  TIME TO GATHER INFO FROM STORAGE
ADVANCE 30  AVG TIME TO PRINT RESULTS ON AN AVG SAMP

* BASED ON PRINTING 3 LINES FOR A PARTIAL REPORT AT 1 LINE PER MINUTE
RELEASE 15  COMPUTER
TERMINATE  TERMINATE REQUEST FOR RESULTS
PRE1C PREEMPT 15  PREEMPT COMPUTER
DEPART 9  DOCTOR REQUEST FOR PARTIAL REPORT QUEUE
ADVANCE 1  TIME TO INTERRUPT
ADVANCE 30  AVG TIME TO PRINT RESULTS ON AN AVG SAMP

* BASED ON PRINTING 3 LINES FOR A PARTIAL REPORT AT 1 LINE PER MINUTE
RETURN 15  RELEASE COMPUTER
TERMINATE  TERMINATE REQUEST FOR RESULTS OF TEST

1 VARIABLE P3+P4+P5+P6+P7+P8+P9+P10+P11 INDICATES ANY TEST EXCEPT URE GLU
2 VARIABLE P1+P2  INDICATES UREA OR GLU TEST
3 VARIABLE P1+P2+P3*4+P5+P7*2-1 N OF PARTS OF SAMP COMP ASSEMBLES FOR REP
* EITHER — GLU, UREA, CL, NA, K, URIC ACID, TOT PROT, AG, BICAR
4 VARIABLE V1*10  TIME TO PRINT RESULTS OF TEST - 1 LINE PER INIT
* V4= NO. OF TEST OF SAMP * TIME TO PRINT ONE LINE
5 VARIABLE Q13+Q14+Q15+Q16+Q17+Q18  Q WAV SPIN+WAV STORAGE FOR CENT+
6 VARIABLE S13+Q14  NO OF SAMP THAT TOT PROT A G TEC YET TO PROCESS
7 VARIABLE Q13+Q14+Q15+Q16+Q17  NO OF SAMP THAT TOT PROT A G TEC YET TO PROCESS
8 VARIABLE Q13+Q14  NO OF XACT IN WB AT BEN AND PIP GLU LINE
9 VARIABLE S*13*K1/K2  TIME TO PROCESS MARBLES N RACK
10 VARIABLE C1+4600  CLOCK TIME
11 VARIABLE V1+V2  SUM OF PAR FROM 1 TO 11
12 VARIABLE S33+Q16+Q13+Q14  CONTENTS OF WAITING RACK UREA QUE ETC
13 VARIABLE 1*S*15  TIME TO SQURT WHEEL TOTAL
14 VARIABLE 1*S*15  TIME TO TAKE OUT AND SWIRL ROD
15 VARIABLE S49+S48  NO. OF NA P ,CL SAMPLES LEFT IN SYSTEM
16 VARIABLE V8+S17+S15  V8+GLU CENT + GLU SHAKER
17 VARIABLE V5+S*17  V5+ CONTENTS OF CENT
18 VARIABLE
19 VARIABLE X12*10 TIME TO PRINT WARD REPORT
* V19= NO. OF SAMP * AVG NO. OF TEST PER SAMP * TIME TO PRINT ONE LINE
21 VARIABLE Q*17+S2 QUE WAIT SPIN + STORAGE FOR TIME PUNCH
*
1 FUNCTION RN1,C3 FIRST DELIVERY 745
0 93. .25 93. 1. 96.  SECOND DELIVERY 815
2 FUNCTION RN1,C3
0 99. .75 101. 1. 101.  THIRD DELIVERY 830
3 FUNCTION RN1,C4
0 114. .25 115.3 1. 115.3  FOURTH DELIVERY 930
0 126. 1. 126. 1030 DELIVERY
0 138. 1. 138. 1130 DELIVERY
6 FUNCTION RN1,C2
0 28. 1. 28. 30.  RNUMBER OF TEST ARRIVE AR 730
0 11 FUNCTION RN1,C2
0 100. 1. 100.  NUMBER OF TEST ARRIVE AT 815
12 FUNCTION RN1,C2
0 33. 1. 33.  NUMBER OF TEST ARRIVE AT 830
0 33. 1. 33. 37.  NUMBER OF TEST ARRIVE AT 930
0 10 FUNCTION RN1,C2
0 2. .8 2. 1. 2.  NUMBER OF TEST TUBES WITH A SAMPLE
0 51 FUNCTION RN1,D3
0 1. .9 1. 1.  NUMBER OF TEST TUBES WITH A SAMPLE
60 FUNCTION RN1,D4
.3 1. .6 2. .8 3. 1. 4.  NUMBER OF TEST TUBES WITH A SAMPLE
70 FUNCTION RN1,D2
.9 0. 1. 1.  ONE ONE HUNDRETH OF A MINUTE
1 FUNCTION VN10,C3 TIME TO GET PLACED IN CLORIDE MACHINE
4800. 1. 6600. 2. 10200. 2.  
72 FUNCTION C1,D3 TIME TO MAKE SEPARATE TUBES8 LIG IN FOR SPINNER
5700. 8. 7200. 11. 10200. 12.  
73 FUNCTION C1,D2 TIME SPENT FOR WB AT SPIN BENCH TO GO TO WB
### STORAGE DEFINITION CARDS

<table>
<thead>
<tr>
<th>Storage</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 STORAGE 12</td>
<td>CAPACITY OF SPINNER'S CENTRIFUGE</td>
</tr>
<tr>
<td>4 STORAGE 12</td>
<td>CAPACITY OF SPINNER'S CENTRIFUGE</td>
</tr>
<tr>
<td>5 STORAGE 13</td>
<td>CAPACITY OF READY RACK</td>
</tr>
<tr>
<td>6 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>7 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>8 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>9 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>10 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>11 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>12 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>13 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>14 STORAGE 13</td>
<td>CAPACITY OF READY RACK - ELECT SPINNER</td>
</tr>
<tr>
<td>15 STORAGE 8</td>
<td>CAPACITY OF MAGNETIC STIRRER FOR GLU</td>
</tr>
<tr>
<td>16 STORAGE 8</td>
<td>CAPACITY FOR CENTRIFUGE FOR GLU</td>
</tr>
<tr>
<td>18 STORAGE 114</td>
<td>CAP OF GLU WAIT RACK TO INCUBATE</td>
</tr>
<tr>
<td>19 STORAGE 114</td>
<td>CAP OF GLU WAIT RACK TO INCUBATE</td>
</tr>
<tr>
<td>20 STORAGE 114</td>
<td>CAP OF GLU WAIT RACK TO INCUBATE</td>
</tr>
<tr>
<td>31 STORAGE 30</td>
<td>CAPACITY OF UREA WHEEL NO. 1</td>
</tr>
<tr>
<td>32 STORAGE 30</td>
<td>CAPACITY OF UREA WHEEL NO. 2</td>
</tr>
<tr>
<td>35 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>36 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>37 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>38 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>39 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>40 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>41 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>42 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>43 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>44 STORAGE 13</td>
<td>CAPACITY OF READY RACK-NONELECT SPINNER</td>
</tr>
<tr>
<td>50 STORAGE 1</td>
<td>CAPACITY OF CHLORIDE MACHINE</td>
</tr>
</tbody>
</table>
57 STORAGE 6 CAPACITY OF BICARBONATE AIRATOR

* *
METHOD OF STARTING PROGRAM
* *

GENERATE 0,0,4800,1
TERMINATE 1
START 1
RESET

** TIMER FOR SNAPS

GENERATE 300,0,0
GENERATE RESET TRANS AT 8.00
A MEANS OF RESETING STATISTICS BEFORE WORK
START 18,,1
SNAP EVERY 1TERM (1/2 HOUR) TILL 5 OCLOCK
END

NOTE RELATIVE TIME IS 0 WHERE ABSOLUTE TIME IS 4800 OR 8.00 O'CLOCK
TERMINATE 1
## LIST OF PRINCIPAL SYMBOLS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>symbol</th>
<th>page</th>
<th>interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>:=</td>
<td>14</td>
<td>replaced by</td>
</tr>
<tr>
<td>η</td>
<td>22</td>
<td>denotes n spaces, where n is some number</td>
</tr>
<tr>
<td>Λ</td>
<td>22</td>
<td>null string, void</td>
</tr>
<tr>
<td>#</td>
<td>22</td>
<td>end of card</td>
</tr>
<tr>
<td>†</td>
<td>23</td>
<td>assertion sign</td>
</tr>
<tr>
<td>|</td>
<td>26</td>
<td>punctuation used in the contraction of remarks</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>assertion sign of canonic system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of canonic system of limited FORTRAN</td>
</tr>
<tr>
<td>!</td>
<td>45</td>
<td>denotes the letter 1 as in label, as distinguished from the number '1' one.</td>
</tr>
<tr>
<td>%</td>
<td>67</td>
<td>end of card</td>
</tr>
<tr>
<td>So</td>
<td>76</td>
<td>set of n-tuples corresponding to elements of an object system</td>
</tr>
<tr>
<td>C_{qr}</td>
<td>77</td>
<td>denotes an n-tuple, where q denotes a particular subset and r the member of that subset</td>
</tr>
<tr>
<td>Tr[C_{ij}, k, substitution rule]</td>
<td>78</td>
<td>denotes a translation. C_{ij} denotes the n-tuple to be operated on, k the scalar of that n-tuple, j the n-tuple, and i the subset</td>
</tr>
<tr>
<td>S_m</td>
<td>79</td>
<td>set of n-tuples of the model</td>
</tr>
<tr>
<td>Tr_m</td>
<td>80</td>
<td>translation among the n-tuples of the model</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>Abbreviations</th>
<th>=</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>=</td>
<td>argument</td>
</tr>
<tr>
<td>def</td>
<td>=</td>
<td>definition</td>
</tr>
<tr>
<td>info</td>
<td>=</td>
<td>information</td>
</tr>
<tr>
<td>loc</td>
<td>=</td>
<td>location</td>
</tr>
<tr>
<td>num</td>
<td>=</td>
<td>number</td>
</tr>
<tr>
<td>op</td>
<td>=</td>
<td>operation</td>
</tr>
<tr>
<td>pref stand num attrib</td>
<td>=</td>
<td>prefix standard numerical attribute</td>
</tr>
<tr>
<td>ref</td>
<td>=</td>
<td>reference</td>
</tr>
<tr>
<td>sna</td>
<td>=</td>
<td>standard numerical attribute</td>
</tr>
<tr>
<td>sna*</td>
<td>=</td>
<td>standard numerical attribute with indirect addressing</td>
</tr>
<tr>
<td>stand num attrib</td>
<td>=</td>
<td>standard numerical attribute</td>
</tr>
<tr>
<td>stm</td>
<td>=</td>
<td>statement</td>
</tr>
<tr>
<td>sym</td>
<td>=</td>
<td>symbol</td>
</tr>
<tr>
<td>var</td>
<td>=</td>
<td>variable</td>
</tr>
</tbody>
</table>
BIBLIOGRAPHY


References [1-7] are concerned with the simulation of complex systems. Some of the needs and deficiencies of existing simulation methodology are discussed in [1,2]. The titles of [3,4,5] are
self-explanatory. References [6, 7] are textbooks dealing with simulation.

The terminology of this dissertation is for the most part consistent with that used in these references. This dissertation is an attempt to fulfill some of the needs and deficiencies of existing simulation methodology discussed in [1-7]. In this dissertation is presented a different viewpoint of systems than is put forth in these references.


The subject of reference [8] is formal logic. Precise definitions of many of the terms used in the presentation of the complete syntactical specification of computer languages found in this dissertation (Chapters II and III) are given by Church. (e.g. syntax, semantics, premise).


References [9, 10] are the manuals of GPSS III. GPSS III is a simulation language. In this dissertation is presented a complete syntactical definition of GPSS III and a discussion of its use. The syntactical constructions that are ambiguously or incorrectly defined in these manuals

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
have been correctly defined in this dissertation. GPSS III is also used in this dissertation to demonstrate some techniques used in digital simulation.


In reference [11] is presented one of the first attempts at syntax directed compilation. The specification presented in this dissertation may similarly be used to make the syntax phase of compilers independent of their source language. It is impossible or difficult to express many of the constructions of existing computer languages in the specification used in [11]. Using the specification presented in this dissertation it may be possible to achieve that aim of making the syntax phase independent for any computer language since the specification may be applied to any computer language.


15. Iverson, K. E., "A method of syntax specification," Comm. ACM,


In references [12-24] are attempts to formally specify portions of
the following languages by grammars which are equivalent to a context free phrase structure grammar: ALGOL, JOVIAL, FORTRAN, NELIAC, APT, COBOL, BALGOL. These representations are useful; however, many constructions of existing computer languages are difficult or impossible to express in them, and in some cases simply have not been defined in them. E.g., column format, the restrictions that all statement labels must be different, that all reference labels must correspond to statement labels in the same program, all subscripted variables must be declared. The specification presented in this dissertation can be used to express all of these constructions.


References [25-33] are expositions of the languages which had portions of their syntax formalized in [12-24].


References [34-40] are studies in syntax directed compilation.
The aim of these works was to investigate ways to design compilers whose syntax phase was independent of its source language. That is, several source languages could be used with the same compiler. This aim was not reached for existing computer languages. Using the specification presented in this dissertation it may be possible to achieve that aim for any computer language since the specification may be applied to any computer language. If a program were written to accept any specification of the form presented in this dissertation and then to proceed to recognize strings of the language defined by the specification, the syntax phase of any compiler could be independent of its source language.


44. Bar-Hillel, Y., Perles, M., and Shamir, E., "On formal properties


References [41-54] are studies in the general theory of phrase
structure grammars, [55-61] are studies in the theory of languages and automata. These writers have presented proofs of insolubility and ambiguities associated with phrase structure grammars. They have studied the construction of parsing systems. The most valuable theoretical insight gained into these diverse systems is the realization that their diversity is, in a sense, only superficial. Many of these systems have been shown to be equivalent to context free phrase structure grammars. This realization of equivalence among these grammars is used in this dissertation to classify all the specifications of [12-24] as context free phrase structure grammars. Hence, these specifications cannot completely classify the syntax of all computer languages because many computer languages are context sensitive [65, 66].

If a link between Chomsky's work [46, 47] and the specification presented in this dissertation could be established many of the theorems stated and proven in the above works would become applicable to the specification. This dissertation uses Chomsky's [46, 47] definition of phrase structure grammars.


64. Wirth, N. and Weber, H., "EULER, a generalization of ALGOL

These works [62-64] are approaches to the problem of formalization of the semantics of computer languages. If the semantics of a computer language is formalized, then the syntax should also be formalized. The above works use a context free phrase structure grammar to specify the syntax of computer languages and thus suffer the limitations and difficulties of [12-24] - [65-66]. Namely, many syntactical constructions of existing computer languages are difficult or impossible to express in them. The specification presented in this dissertation may be used to express these constructions.


Context-free grammars have been used in attempts to describe languages such as ALGOL. Discrepancies arise in such use; ALGOL, for example, is not context free because it contains data-defining statements as well as definition-dependent data use. The above articles [65, 66] and that of Floyd [58] point out such discrepancies in the use of
context free grammars to describe computer languages. Using the specification presented in this dissertation it is possible to express the constructions that these references state are not expressible in context-free grammars and, hence, not expressible in any known specification for computer languages.


This dissertation has applied known methods of recursive definition to the development of a complete syntactical specification of a computer language. The specification is a modification of the work of [67]. The modifications have made this work applicable to programming languages. More's work is an applied variant of Smullyan's elementary formal systems [68] and Post's canonical systems [69]. Post's work is explained in [70]. These works are the basis of the specification presented in this dissertation.
References [71-72] are expositions of two simulation languages, GPSS and SIMSCRIPT. These languages are the most widely used languages for digital simulation of complex systems. This dissertation presents a discussion of the use of these languages and an analysis of the syntax of GPSS. An example of a digital simulation using these languages is given.

In reference [73] is presented the systems program that must be placed before every GPSS III program that is to be executed on an IBM DCS 7094-7040.

In reference [74] is presented an analysis of the organization, procedures, and future configuration of a clinical chemistry laboratory. This analysis was performed using simulation. Insight into the many problems of introducing data processing equipment is presented in this
reference. In this dissertation portions of the simulation of a clinical laboratory are used to demonstrate the digital simulation techniques presented.
