Title
Frame selection systems and languages for medical applications

Author
LeBeux, Pierre Joseph

Publication Date
1974

Peer reviewed|Thesis/dissertation

FRAME SELECTION SYSTEMS AND LANGUAGES
FOR MEDICAL APPLICATIONS

by

Pierre J. LeBeux
Ingenieur, Ecole Centrale des Arts et Manufactures, 1968
Doctorate, 3rd Cycle, University of Paris, 1971

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

MEDICAL INFORMATION SCIENCE

in the

GRADUATE DIVISION

(San Francisco)

of the

UNIVERSITY OF CALIFORNIA

Marsden S. Blois Jr.

[redacted]

Date Librarian

Degree Conferred: 22 JUN 1974

Acknowledgments

This research was done in the Office of Medical Information Systems at the University of California Medical Center at San Francisco, under the patient guidance and supervision of Dr. Marsden S. Blois and Dr. Ronald R. Henley. I was honored to have Dr. George Brecher on my Thesis Committee. Dr. Richard de Leon and Dr. Don Carlos Hines contributed substantially and advised me during the design of the pharmacy application.

The teaching of Professor John Rhodes in Berkeley gave me valuable insights into the theories of abstract machines.

Mr. Larry E. Selmer contributed to the programming of the pharmacy application and Mr. Sinclair Lai performed the entering and checking of the drug formulary.

I would like to acknowledge the valuable discussions with Mr. Martin Epstein, Mr. Giovanni Wiederhold, Mr. David Steingart, and Mr. Lance Carnes.

A special acknowledgment is due to Marina Mancia and Tina Walters for their patience, their spirit, and dedication during the typing of the many drafts of the dissertation.

Finally, I would like to thank Professor Laugier and Professor Gremy from the University of Paris for their advice and help in my coming to the United States.

This research was partially supported by a NATO Fellowship during the first year, and by an IRIA (Institut de Recherche Informatique et Automatique) Fellowship during the last year.

Abstract

A frame selection system and a frame programming language have been implemented to support medical information system applications.

A frame selection system is an interactive computer system which enables the user to enter data via a CRT terminal, by using a selection device to point at items or phrases displayed on the screen. A frame is a set of choices representing the alternatives available to the user at each stage of a selection process. A model has been developed to study the capabilities of frame selection systems. The simplest type of frame selection system can be modelled by a finite state machine called the selecton. A selecton is characterized by selection inputs and a set of frames organized in lattice or digraph structures. A selecton can be used to generate output strings or sentences of a right linear grammar encoded in the frame structure. This type of system enables the user to generate syntactically correct sentences of the language whose grammar is represented by the frame structure. Pursuing this property of frame selection systems, the addition of a pushdown store enables the model to become a generator of syntactically correct sentences of any context free language. A universal frame selection system is defined as a system which enables the execution of procedures as a result of a selection. This feature allows the generation of sentences of context sensitive languages. It also becomes possible to generate semantically correct sentences by defining semantic relations between choices of different frames.

The concept of frame programming languages arises from the need to have a tool to build frame selection systems. A frame programming language is a language which enables the definition of the actions to be

taken as a result of a selection. A frame programming language (FPL) is presented and has been implemented as a frame selection system. It is supported by a real-time operating system built on a minicomputer especially designed for the support of CRT's and selection devices.

The concepts of frame programming language and frame selection systems are useful in a medical environment because they can be used to develop interactive medical information systems which are not rigidly constructed but can be extended and developed incrementally.

To illustrate the capability of such systems and languages, a pharmacy ordering system is presented. It is used to capture all the medication orders generated within the hospital and keep a patient's profile accessible and modifiable in real-time. Various reports are generated, a data base of all medications orders is retained, the inventory control is achieved and the billing information is captured automatically. This application shows that dedicated stand-alone frame selection applications can be developed with minimum hardware and adequate response time.

TABLE OF CONTENTS

APPENDIX

Chapter 1

Introduction

" Ce n'est point de l'espace que je dois chercher ma dignite', mais c'est du reglement de ma penseé. Je n'aurais pas davantage en possedant des terres. Par l'espace l'univers me comprend et m'engloutit comme un point, par la penseé je le comprends."

Blaise PASCAL Penseès

1. INTRODUCTION

The advancement and usefulness of technology can frequently be measured by the success of its medical applications: chemistry gave drugs and more insight into the biological processes; physics gave x-ray and nuclear medicine; mechanics gave motors and engines for dialysis, transfusion, reanimation, etc.; electronics gave pacemakers and the monitoring of acutely ill patients.

It would be incorrect to say that computer science and data processing did not contribute to the medical field, but it is also true that their impact is not as yet as important as other branches of technology. The reason for this can be explained partially by the immaturity of the field, but also by a fact which is often overlooked: data processing did not start with the existence of computers and therefore there is no immediate need for their use in the medical environment as long as the function of data processing can be achieved by other means in a more satisfactory fashion.

Following the examples of other technologies: an x-ray machine cannot be replaced by human vision or a camera, penicillin cannot be replaced by a good meal, a cardiac pacemaker cannot be replaced by autosuggestion, etc. It seems therefore that a technology becomes useful for the medical field when it provides a service that cannot be obtained by other means and which is necessary or useful for the diagnosis and treatment of patients or the administration of the health care system.

It follows from these remarks that computers will be recognized as necessary and useful in the medical field if and only if they can provide information that could not be as effectively produced by other means and which is useful in the treatment and diagnosis of diseases.

If this statement is true then there is no doubt that in the long run this goal will be achieved and that computers will provide "a service" that nothing else could supply namely: speed, accuracy and consistency. The issue is now to show that these services are "worth their price" and can be achieved (at a worthwile level) with this technology. However, the analysis of cost effectiveness is even more difficult than for other technologies:

- a dialysis machine can be immediately justified by counting the number of persons which cannot survive without it
- a computer system will not directly save lives but it will indirectly provide a service that may prevent some deaths or illnesses.

The previous statement should therefore be reformulated: computer technology is becoming cheap enough so that it should be possible to demonstrate to medical professionals and administrators that a computer system could provide services that cannot be obtained otherwise. The challenge is therefore to build systems which are sophisticated enough to help the physician without being too costly to develop and to support.

The purpose of the following dissertation is to study frame selection systems and languages, and their applicability to the medical field. From a user's standpoint, a frame selection system is an interactive system using cathode-ray tubes (CRT), equipped with selection devices such as, light pen, joystick, etc., used as a primary input mechanism. In such a system the CRT images presented to the user are called frames and different frame structures can be built for specific applications.

Chapter 2 presents a short review of medical information systems (MIS) and looks at the different approaches already taken to build

medical applications. The most comprehensive and most recent systems have used a method similar to the frame selection approach, but very little is known about the capabilities of such systems. Hence our interest in the subject is motivated by the need for an analysis and a more formal study of such systems and their linguistic capabilities.

Since the area of medical information science is not yet a well delimited field with well defined boundaries, Chapter 3 is devoted to the concepts of information from different points of view: information theory, semantic theory, and linguistic theory. This is done by emphasizing the fact that all connotations of information are related to the selection power of an object from a class of possible objects. A brief study of the particulars of medical information follows.

The first section of Chapter 4 is a study of the static properties of the frame selection systems; some new concepts are introduced: selection process, selection formulas, frame decomposition, frame structures (lattice or digraph). The next section of Chapter 4 defines a model applicable to the concept of frame selection system: the model is a finite state machine called selecton which directly mimics the behavior of frame system and generates regular strings. However, the selecton model is not used as a recognizer of strings, but like a grammar, it is used to generate strings resulting from selections inputs. The frame structure is used to encode the grammar of a regular language. Similarly, the addition of a pushdown stack enables the model to generate sentences of a context free grammar encoded in the frame structure. Finally, the introduction of procedures, which can be executed as a result of a selection gives to the model the capability of generating some transformation on strings and allows the generation of sentences of context sensitive languages.

In Chapter 5 this type of frame selection system which has the capability of generating syntactically correct sentences of any language is used to define the concept of a frame programming language. A frame programming language is a language used to build frame selection applications, and can be, itself, a frame selection application. This method enables the immediate parsing of the sentences generated and the compilation process can start directly at the code generation.

An implementation of a frame programming language is presented and a real-time operating system designed for the support of frame selection system applications is described in Chapter 6.

The last chapter is a description of a pharmacy ordering system, developed as a frame selection system at the Medical Center of the University of California at San Francisco.

Throughout the text, the pharmacy application will be used as a concrete example to show the usefulness of the concepts. Conversely, the more abstract concepts will show that the same methods can be used successfully for other applications in the medical information field.

Chapter 2

Review of Medical Information Systems

"The modified offspring from the later and more highly improved branches will, it is probable, often take the place of, and so destroy, the earlier and less improved branches."

Charles Darwin

The Origin of Species

2. REVIEW OF MEDICAL INFORMATION SYSTEMS

The concept of a medical information system refers to the data handling and communications taking place in a medical facility (hospital or clinics.) Several studies have shown [COL70] that approximately one fourth of total hospital costs are related to information processing.

The primary source of information is still the patient, but with the growth of medical knowledge and advance of technology, the production of clinical data relating to the patient has been split into several specialized production centers (nursing wards, admissions, clinical lab, pharmacy, pathology lab, microbiology, radiology, etc.) which are geographically separated. Therefore, medical information is processed and aggregated by a complex system which includes several types of professionals: physicians, pharmacists, nurses and technicians. Such a system can be viewed as a communication network with several processing centers [BLO71].

In the last decade several attempts have been made to automate such systems, but very few seem to be recognized as successful. In the USA the principal reasons for adoption of such systems are cost savings, and improved information processing: the first argument is purely economic and managerial, the second is frequently limited by the problem of man/machine interaction.

A great variety of approaches have been taken, but so far the most successful systems seems to be those which have a limited scope such as clinical laboratory systems [BRE71,JOH71], pharmacy systems [COH72, GOU71], and medical history systems [SLA66]. Among the more comprehensive systems are those research projects or commercial systems which were designed to be used by more than one department.

The following review will not be exhaustive but we will look at the better known and representative types of system. For a more complete survey see [BAL74] which reviews the existing systems.

2.1 CARD-ORIENTED SYSTEMS

It is not our intention to consider the early billing and accounting systems as medical information systems. However, among the commercial systems which use punched cards as data input medium one of them deserves some attention: the MEDELCO total hospital information system. Although it is not a highly sophisticated system it is one of the few systems that became operational and has been duplicated at more than a few sites.

The analysis of this commercial success seems to be due to two factors:

- the first is the recognition of the communication aspects of medical information systems
- the second is that it does not try to solve the particularly difficult medical record problem

It is a communication system of medical orders originating on the wards and it transmits orders and requests to and from the nursing stations. The data entry is done by pre-punched cards for each order, service, or product available in the hospital, and each order card is accompanied by a patient's card which has been typed and punched at the time of admission of the patient. When the cards are read, the data is also printed at the originating station and will be inserted in the patient's chart.

The system does not have to face the problem of man/machine interaction because it replaces paper requisitions by pre-punched cards and teletype printers. Since the system appears to be cost-effective and accepted by the nursing personnel, it can be considered a success and

it may be considered as a first step toward the development of more sophisticated medical information system. The system has been partially emulated by several small vendors (Metric) and IBM using programmable message switchers and newer card readers.

2.2 COMPUTER TIME SHARING SYSTEMS

More sophisticated systems are offered by computer time-sharing systems. The advent of computer time-sharing has greatly affected the data-processing field and can be characterized by two main advantages over the batch-systems:

- easy accessibility and possibility of man/machine interactions
- development of conversational system and languages

A good example of such type of systems is MUMPS (Massachussetts Utility Multiprogramming Systems) developed at the Massachussetts General Hospital [BAR67].

MUMPS is both a time-sharing system built on medium-size computers and a high-level programming language [BAR70] for the development of medical applications. The MUMPS system made a major impact in the field of MIS because it demonstrated the validity of the following hypotheses:

- necessity of a new software tool: language designed for medical applications
- feasibility of implementation on medium and small computers
- importance of the files and data-base structures

However, at the same time, MUMPS design suffers from its historical past:

- the interpretative nature of the language certainly offers more flexibility for the development of programs, but it becomes a serious hindrance in a production environment. The ease of debugging and development of programs should not be paid by the

constant overhead of interpretation when the programs are operational and used on a routine basis.

- The language itself suffers from a lack of structuring features which is worsened by the necessity of sparing disc and core memory space by using abbreviated source code (which has the consequence of reducing the internal documentation to a minimum.)
- The data base file system although quite flexible suffers from the high overhead necessary to access the data elements which are stored in the "leaves" of tree-like structure. (There is a high input-output demand from the secondary storage to the main memory when a data element is needed.) Furthermore the modern techniques of data-base management have shown the necessity of separating the data from its structure and description in order to avoid a unique binding of the data.
- Another historical feature of the MUMPS system is the fact that it was designed for teletypewriter interactions. This is acceptable for motivated users in a research environment but becomes unacceptable for a production environment where physicians and nurses are not accustomed to and resent using typewriters. The replacement of such terminals by CRT terminals does not solve this particular problem because the structure of the dialogue remains the same: a question is followed by an answer which has to be typed in.
- Finally, although MUMPS systems can be built in a modular fashion [BAR74], it is difficult to link different systems in an efficient manner to insure the fast communication of data from one module to another.

Nevertheless, MUMPS applications are among the best known systems which are truly medically oriented. This is certainly due to the inherent concern of the designers to build a tool which was adapted to the need of the medical field and to develop applications with the participation of the future users.

In this sense, MUMPS must be considered a success because it meets the original requirements of specificity and adaptation to the medical field. However, if we consider the field as an applied research area in which some progress is to be made, the question to be asked is: given the existence of such systems and the present technology, is it possible to keep the positive attributes of earlier systems and to remedy as much as possible their recognized shortcomings?

Following the analysis of MUMPS, another time-sharing system should be mentioned for its relevance to the field; it is the ACME system (Advanced Computer for Medical Research) developed at Stanford [CRO69, WIE69]. Here also the project is based on the recognized need for a high level programming language which allows an interaction between the user and the machine.

The language developed on a PL/I subset called PL/ACME [BREI68] is also an interpreter, but here the overhead problem associated with the interpretative execution was solved by using a medium size computer with an extensive main memory capacity. The ACME language has, in many respects, more modern features than MUMPS (possibility of structuring programs, extensive arithmetic capabilities as well as string manipulation, extensive file structuring capabilities) [WIE70], and is also adapted to the development of interactive and real-time medical applications.

However, ACME was more intended for research purposes than for

production purposes and a good example is the pharmacy drug-drug interaction project [COH72] which was initially developed on ACME and then transferred to a MUMPS system when it was decided to use it in a production environment. Although many interesting research applications have been developed on ACME it has never been used as a basis for a production of a total medical information system.

The most interesting results from ACME may not be the language itself, but the data-base concepts that have emerged from the actual use of the language and system. The concept of time oriented data bank (TOD) [FRI72] and the formal approach to data base design which separates the data from the structure in the form of data-base schemas [WIE72] is the most interesting and promising fall-out of the ACME project.

Unlike MUMPS, ACME has not been exported because of its major cost but it is most likely that the data-base approach will have to be considered in the design of new medical information systems. The new generation of powerful mini-computers makes it reasonable and possible to implement such data base concepts on rather small systems therefore reducing the cost of the hardware for supporting such a system.

Another interesting operational medical application developed with computer time-sharing techniques is the system developed at the Texas Institute for Rehabilitation and Research [VAL68].

The system was designed on a medium size computer with CRT terminals through which data is entered and retrieved via keyboard entered codes and was programmed in a high-level programming language [PL/I].

Although the system was developed in a specialized environment, several applications have been implemented; admission and bed census, patient treatment scheduling laboratory reports and hospital management.

At the same time, the system allows for clinical data processing for research purpose and assists the physician in the decision-making for patient management. The system also provides the functions necessary for general acute care hospitals (physiological monitoring.) However, despite the fact that the system was successfully implemented and is still operational, it has not been transported to other institutions because most of the application programs were specifically related to the specialties of that particular institution.

2.3 THE MENU-TREE SYSTEMS

At the same time, that MUMPS and ACME were pursuing the route of conversational languages and time-sharing, several commercial attempts were made to build real-time and interactive medical information systems. The best known of which are REACH, LOCKHEED (later TECHNICON), SANDERS (KAISER), CONTROL DATA, SPECTRA MEDICAL. [BAL74,SCHW72]. These systems were designed for the input of medical orders generated on the wards and their communication to the different departments (laboratory, x-ray, pharmacy, etc.) In return these orders generate responses or results, and the data is stored in a computerized record created at the admission of a patient. These record files will serve as a data base for the generation of a variety of CRT displays and printed reports. At the same time the data can be used for the patient's billing and for accounting purposes.

The most striking commonality of these systems is the way they approach the problem of man/machine interaction. Every one of them differs in the implementation but the common basic concept is a "menu-tree" type of system. They are called menu-tree systems because the input is accomplished by reproducing the technique used to order a menu

in a restaurant: one may first select an "hors-d'oeuvre" from the list provided, then a main dish from the list of meat or fish, a beverage from the list of beverages, etc. The method allows the composition of a multitude of meals from a finite list of items.

Since a major portion of the medical information being processed is in the form of messages which have to be transmitted from one point to another (communication) this approach is in fact adapted for the building and transmission of messages (medical orders, results, requests, etc.) It can be implemented with a CRT associated with a selecting device such as light pen, sidescreen buttons, joystick, touch screen, etc.

The main advantage of the method is that typing is not required for most of the messages and by selecting phrases to build the messages the process of data entry is accelerated. The user has the impression of having a "phrase typewriter."

Besides the fact that a menu-tree method avoids the necessity of typing, it is interesting because of the exponential ramification of trees: if each CRT image contains n choices, at the second level there are n^2 possible different choices and n^3 at the third level and so on.

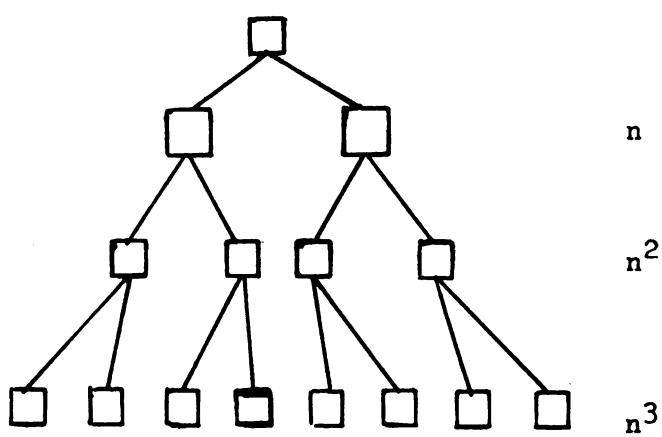


Figure 2.1 - Generalized "Menu-tree"

This type of tree is called a generalized menu-tree (it can have as many levels as desired.) Example: for n=30, the third level gives 30^3= 27,000 choices. With four levels one could access 810,000 items which is certainly enough to access all the words contained in Websters dictionary plus all the medical vocabulary.

Another advantage of the method is that it does not require any computation (a selection is the expression of one's freedom of choice on a given set) as opposed to a typed or punched word which must be looked up in a dictionary to be validated.

However, most of these systems have been restricted to simple trees and do not allow loops and more general graph structure. It is therefore difficult to encode the complexity of some medical orders or sentence constructs. Despite the fact that this type of man/machine interaction was more readily acceptable than a typewriter keyboard, these commercial systems failed to be widely accepted because they were too costly or did not meet the expectations of the medical professionals.

These failures can usually be traced to three factors:

i) the designers deliberately or unconciously ignored the modular approach and had to use complex and expensive equipment.

ii) the designers did not have a clear appreciation of the complexity of the medical record.

iii) the reliability and acceptability to the users (especially the convenience of the terminals and an adequate response time) were difficult to meet.

The first error can clearly be corrected by using a modular approach but the second one is more difficult to prevent because of the variety of schools concerning the medical record and the engineering tendency

to believe that everything can be quantified, planned, and standardized in a rigid fashion.

The third factor is more related to the engineering design. The usual assumption is that a MIS is just an application system that can be built on any standard operating system. On one hand, there exists the standard real-time operating systems which allows a fast switching between tasks and users but do not support sophisticated data base systems, and on the other hand, there are the standard multiprogramming or time-sharing systems which may have suitable data base features but do not have the efficiency to give quick response time for urgent requests.

The result is that if one chooses the first type of system ("standard real-time") a great deal of system work has to be done on the file and data base structuring; if one chooses the second type of system, it becomes a "fight for adequate response time" which is either solved by programming tricks or by switching to a larger machine. This in turn accounts for the fact that many of these projects have been delayed and had unexpected development costs.

The conclusions from these attempts to build comprehensive medical information systems are that most seem to agree on the suitability of the CRT menu-tree interface, but difficulties arise because the hospital-wide approach is intimately related to the medical record problem. In addition, this total approach represents a gigantic effort which requires not only technical skills but also an explicit and detailed knowledge of the hospital processes. The main part of this knowledge is not necessarily represented by the routine functions but also by all the special cases, the exceptions and emergencies and the parallel data paths developed to shunt the regular paths. This knowledge can only be

obtained by working in full cooperation with the hospital personnel (not only physicians, but also pharmacists, nurses, technicians, clerks, etc.) To be successful, the personnel must be prepared technically and psychologically to use new techniques and machines.

2.4 FRAME SELECTION SYSTEMS

Another type of system which is closely related to menu-tree systems is the frame selection system. A frame selection system differs from a pure menu-tree system by allowing a more general structure between the CRT displays (frames.) Instead of being a tree the structure can be any graph.

The mechanism of man-machine interaction can be identical to the ones used for the menu-tree systems and therefore all the advantages of such devices can be applied to a frame selection system.

Although the term frame selection system was not used the first system of this type was developed at the Western Reserve University in cooperation with Control Data Corporation and then pursued at the University of Vermont [WEE69]. Such a system has been used for the implementation of a prototype computerized Problem-oriented Medical Record [SCH71]. It uses a touch-screen device for the selection of items on a display. This work is particularly interesting because unlike the previous commercial systems, the subject of medical information systems was approached through a careful analysis of the medical record and a clear philosophy of organization of the medical record.

For the reader not familiar with the Problem-oriented Medical Record [WEE68] it is a record which is organized by defining, enumerating and following the problems of the patient. These problems may or may not be associated with a diagnosis depending on the completeness of

the data and the ability of the physician, but each problem must be followed, and each action taken to get more data or to treat the patient must be related to a given problem. Each result and progress note is also related to the problem list. The advantage of the method is that it requires a systematic approach to the treatment of patients which is useful for teaching and also for record-structuring purposes. In a conventional medical record, where the data is source-oriented, the organization tends to be diffuse and the interpretation of the data by a reading physician is more difficult.

Among all the systems operational or in a prototype stage Weed's system is certainly the most comprehensive, the most medically-oriented and the most useful for its users and patients. However, in regard to the design concepts already mentioned, at least one concept is clearly missing: modularity and low cost.

This may account for the fact that this system which has a great potential in educational institutions has not been exported because of the major cost associated with the support of a large time-shared system. A computerized problem-oriented medical record system could be used as a teaching tool if it was available on a mini-computer which could work alternatively on a medical ward, surgical ward, pediatric ward, etc., by simply moving the terminals from one place to another. Not only could this method serve a teaching purpose, but it would help the progressive improvements of the system and the smooth introduction of computers in the daily routine of patient care.

Weed's computer system is also associated with a language called SETRAN (selectable element translator) which is used to build the branching logic of the display frame structure.

Although this language allows the building of a more general structure than a regular menu-tree (it allows multiple selection and loops on frames,) it is still rudimentary in many respects because it uses fixed format fields and octal codes to specify the different fields. When the number of frames becomes as high as 40,000 as it is in Weed's prototype implementation of the P.O.M.R., it is certain that the use of octal codes to link these frames becomes cumbersome if not unrealistic.

The problem of frame languages will be studied later and a more thorough analysis of SETRAN will be done, but it is fair to say that at the time it was conceived, the nature of frame selection systems was not well understood.

Chapter 3

Theories of Information and their Applicability to Medical Data

"The importance of originality is self evident. Selective emphasis on one particular aspect of reality, with its concommitant exaggerations and simplifications, is the essence of model - making and plays almost as great a part in the changing fashions and schools in science as in art."

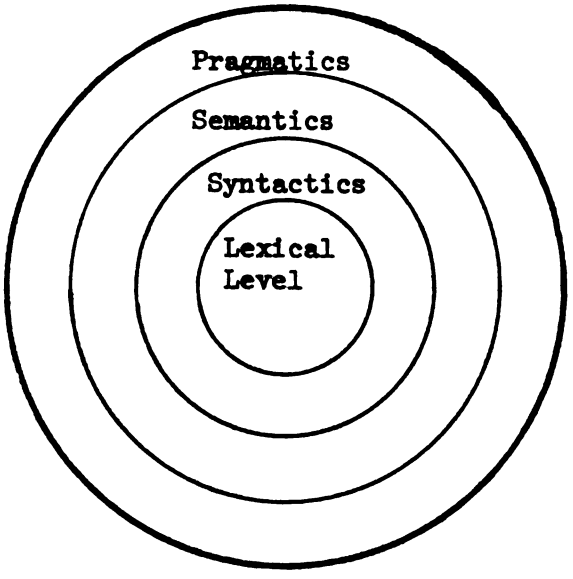
Arthur Koestler

The Act of Creation

3. THEORIES OF INFORMATION AND THEIR APPLICABILITY TO MEDICAL DATA

The study of languages and signs, also referred to as semiotics, can be studied [CHE66] at three differents levels of abstractions: syntactics (signs and relation between signs,) semantics (relation between the signs and their designator,) and pragmatics (aspects which involve the sign users.) It could be extented to include the notion of lexical level.

These four levels can be represented by the following schema:



Cherry [CHE66] remarks that "information in most, if not all of its connotations, seems to rest upon the notion of selection power."

Information theory regards the information source as exerting a selective power upon a collection of lexemes. A lexeme is the orderly selection of the signs, but not the physical signs that are word-tokens or utterances themselves.

At the syntactic level, the generation of sentences is done by the selection of production rules constituting the grammar.

In the semantic theory of Carnap, the information content of statements relates to the selective power they exert upon a set of states.

At present, no formal theory has been published for the pragmatic level, but to a certain extent the information associated with the pragmatic level depends also on a selective power exerted on the environment, the culture, the knowledge and the experience of the individual.

In the following, we present briefly the existing theories corresponding to these different levels: information theory, semantic theory, and syntactic theory.

3.1 INFORMATION, COMMUNICATION AND DATA

Before starting to use the terms information, communication and data, it is necessary to define their meanings precisely. Brillouin [BRI62] states that "science begins when the meaning of the words is strictly defined." The Third Webster's Unabridged Dictionary defines the word information as follows:

> "the communication or reception of knowledge or intelligence...knowledge communicated by others or obtained from investigation, study or instruction...knowledge of a particular event or situation: intelligence, news, advice...facts or figures ready for communication or use as distinguished from those incorporated in a formally organized branch of knowledge: data...a signal purposely impressed upon the input of a communication system or calculating machine..."

Although this definition might be sufficient for the common use of the term information, it is imprecise and does not tell how the words, information, communication and data are related. In fact, this definition suggests that these terms are analogous.

It is therefore not surprising that these terms are easily misunderstood and confused unless we restrict their use to the same precise meanings that they have in areas such as information theory, computer

sciences and linguistics, where their formal definitions may differ from one area to the other.

3.1.1 INFORMATION

In L. Brillouin's book Science and Information Theory, [BRI62], he distinguishes two types of information:

i) live information - which is transmitted with the energy required for its detection (speaking, radio broadcasting.)

ii) dead information - this type of information is involved in the process of writing and reading. The information is stored and unconnected with either energy or negentropy but an additional source of energy is necessary for the reading (light.)

information and entropy - the theory of information shows that information and physical entropy are of the same nature, and entropy can be viewed as a measure of the lack of detailed information about a physical system. Conversely, information represents a negative term in the entropy of a system (negentropy.)

This is due to the fact that from a purely thermodynamic point of view, no observation can be made on a system without increasing the entropy in the physical system itself or in the equipment used for the experiment.

If an observation yields a certain amount of information ΔI and the entropy increase is ΔS during the experiment, the negentropy principle of information states that: $\Delta S \geq \Delta I$ or $\Delta I + \Delta N \leq 0$ where $\Delta S = \Delta E/T$ (or $\Delta Q/T$) and $\Delta N = -\Delta S$ (N is the negentropy) E is the amount of energy that must be degraded.

The purpose of an observation is to obtain some information about the system, and information can be viewed as the ratio of number of

possible system-states before and after the measurement.

If p_o is the number of equally probable states before the measurement, and p_1 the number of such states after the observation, the information gain is defined by:

$$\Delta I = k \log (p_o/p_1) = k \log p_o - k \log p_1$$

This definition of information is related to the <u>freedom of choice</u> that one has to select a given representation of a system.

Shannon's theory of communication [SHA49] defines the amount of information as the logarithm of the number of choices available. If one considers the process of producing a message by successively selecting discrete symbols (letters, words, musical notes, etc.,) the information associated with the message constructed corresponds to the freedom of choice one has in constructing the message. Shannon extends this to the case of n independent symbols whose probabilities of choices are p_o, p_i, ...p_n (they represent the constraints on the freedom of choice). The corresponding information is defined as:

$$I = - \Sigma p_i \log pi$$

If all the p_i are equal that is, $p_i = 1/n$. Then $I = - \Sigma\ 1/n \log 1/n = -\log 1/n = \log n$ which is consistent with the previous definition in the case of equiprobable choices. It results from this that every type of constraint, every additional condition imposed on the possible freedom of choice immediately results in a decrease of information and I is maximum when all the choices are equiprobable.

Shannon defines a <u>relative entropy</u> as the ratio of the actual entropy to the maximum entropy and the <u>redundancy</u> as equal to 1 - relative entropy. For example, the maximum information which corresponds to the maximum entropy of a character selected from a set of 27 symbols (26 characters +

blank) would be $\log_2 27 = 4.76$ bits per letter. If one computes the frequencies of words and letters in English, the information (actual entropy) carried by a letter would be 2.14 bits [BRI62] so that the redundancy and relative entropy of English is approximately 50%. Therefore, when one writes English (or any natural language), half of the text is chosen freely and the other half is determined by the structure of the language. The consequence of this statistical definition of information is that:

- information is related only to the freedom of choice
- information is distinct from meaning and from knowledge
- it ignores the human value of information (i.e. a text of 1,000 letters selected at random carries more information than a poem consisting of the same number of letters)
- information cannot be used to predict the next choice
- information must be paid for by an amount of negentropy greater than or equal to the information obtained.

3.1.2 COMMUNICATION

Information as defined by Shannon is concerned with the technical problem of communication of live information created by a source. In the process of communication, an emitter transmits energy (power) and negentropy which are propagated together with information on a communication channel and a receiver absorbs the energy and negentropy in the same operation by which information is received. In the process of transmission some energy is degraded, some negentropy is lost and some information is lost.

The capacity of a channel C is defined as the number of bits, it can transmit per unit of time (second). In order to transmit information,

the emitter (source) generates a message which is then encoded by a transmitter which sends a signal on the channel and the receiver decodes the signal into a message which cannot carry more information than the message emmitted. If the source emits symbols carrying an information of I bits per symbol, the fundamental theorem established by Shannon states: if the channel has a capacity of C in bits/second, it is not possible to transmit at an average rate greater than C/I symbols/second. If the channel is noisy, the uncertainty is increased and the received signal contains paradoxically more information than the emitted signal.

However, one has to separate this information into two parts: the useful information and the noise information. In this case, there are two statistical processes at work: the source and the noise. Let I (s) be the information of the source and I (r) the information of the received signal. I_r (s) is then defined as the information contained in the input when the output is known and $I_s(r)$ (noise information) is the information contained in the output signal when the input is known.

$I_r(s)$ is called the <u>equivocation</u> and measures the ambiguity of the received signal or the uncertainty in the message source when the signal is known. The total information carried by the system of communication is:

$$I(s;r) = I(s) + I_s(r) = I(r) + I_r(s)$$

and the useful information transmitted in spite of the noise is:

$$I(r) - I_s(r) = I(s) - I_r(s).$$

Therefore, the capacity of a noisy channel will be defined as $C = \max [I(s) - I_r(s)]$, the maximum being taken on all the possible sources of input to the channel.

The fundamental theorem [SHA49] states that given I(s), I(r) and

C in bits per second, such that $C \geq I(s)$, then by an appropriate coding the message emitted by the source can be transmitted over the channel with an arbitrarily small error rate. Stated in other words, this important result shows that it is possible to transmit a message on a noisy channel by adding some redundancy in the coding process so that the receiver will be able to reconstruct the original message despite the noise created on the channel.

Since human communication is imperfect and subject to errors and noise, any message must be transmitted with some redundancy to be reliably interpreted by the receiver. The degree of redundancy depends on the desired reliability and the technical medium used. In human communication, it is probable that the 50% redundancy of natural languages is necessary to enable the recognition of messages transmitted verbally.

3.1.3 DATA

From an information theoretical approach, data can be viewed as dead information which has been written on a memory medium (stone, metal, paper, magnetic tape, film, electronic memory, molecular structure.) This data has usually been collected from observations of the real world.

It must be emphasized that the concept of information exists independently of any memory concept, but the concept of datum is tied with the existence of a memory medium. Conversely, data can become information only if it is associated with the energy (light, electricity, etc.) necessary to read the memory medium. Data is always associated with a given state of the physical world, and as such is subject to the general law of decay according to the second law of thermodynamics. Therefore, if data is to be conserved as a source of potential information, it must be stored with a certain amount of redundancy.

Computing and Data: It is often believed that computing generates new information from data. This is not really true. Although it may seem that a computer program creates new results that did not exist before the execution of the program, no new information is created, and the computation only gives a set of data which is redundant with the original data. At best, this data set contains as much information as the original set.

Usually the effect of computation is to reduce and transform the information contained in the input data into a more useful representation of information contained in the output data. The usefulness can refer to either the meaning, or the most salient features contained in the data. The role of computing is either to translate or to aggregate. Data can be reformatted and displayed in a more convenient representation for viewing and analyzing the most salient aspects of these data. For example, given a set of data measurements in the metric system, a computation can give the value of the measurements in the British system (no new information is created, it is merely a translation.) Another case is the case where the computation will, for instance, give the mean and the variance of a set of values (the information is reduced and aggregated into new results which are possibly more meaningful for the reader.)

Therefore, the task performed by a computer is to translate or to summarize the information contained in the data. Computing only adds a utility value to the information and this utility value may vary among the users. The recent development of data bases [COD70] shows that common data can be treated by different types of users to provide different types of analyses and results.

3.2 SEMANTICS AND INFORMATION

The preceding definition of information is based exclusively on the mathematical theory of communication. It was already mentioned that it did not take into account the value of information. The problem of understanding the meaning of the messages is not considered and the receiver is supposed to know the structure of the language used in the communication process.

From a communication point of view, information is a scalar measure of the information carried by the message when the redundancy due to the internal structure of the language has been removed. This structure can be physical, logical, syntactical, and/or semantic, but in order to be able to give a meaning to a message the receptor must have the physical, logical, syntactical and/or semantic structure encoded in its own memory. (The sender and the receiver must share a universe of discourse - if human.)

It is therefore important to consider the information contained in these structures. A physical structure such as the geometrical structure of an object is always more difficult to code than the word that represent the object. It requires more information to describe the object in the three dimensional space than to define the word in a given alphabet.

This is how any language is able to perform its functions. By associating words with complex objects it can convey much more information than is in reality given by the free choice of the word in a given language or the letters of a given alphabet. This kind of information is referred to as <u>semantic information</u> and it must be learned by experience. Semantic information seems to be related to experience as opposed to the previous which was more related to observation.

In a first approximation semantic information is related to the complexity of objects (shape, structure, etc.) and to the description of time relations (verbs.) The introduction of logic is the next step which enables the building of new concepts by defining sentences using words already defined and the basic logical connectives:

$\neg$	not	negation
$\vee$	or	disjunction
$\wedge$	and	conjunction
$\supset$	if...then	implication
$\equiv$	if and only if (also called iff)	equivalence

According to logical rules every sentence can be evaluated as logically TRUE or FALSE. From a pure information theoretic approach, it would seem that such sentences always carry an information equal to unity. However, a sentence such that: "IF John is sick THEN John is sick." while logically TRUE, actually carries no semantic information because it is a tautology.

It is not our intention to pursue further the consideration of this part of logic known as the propositional calculus but the point is that even in natural languages the introduction of logic is essential to study the semantics of sentences.

The introduction of two other operators known as the universal operator ($\forall$: for all) and the existential operator ($\exists$: for some, there exists) enables the definition of sentences which can be decomposed in a structure of the type "subject/predicate." The subject is an individual taken from a set of possible "values" and the predicate is a proposition where the subject is replaced by a variable: "x is a sick man" is a predicate and "John" is a possible subject among the set of men.

The predicate carries the semantic information and the subject represents the "freedom of choice" of an element among the finite set and can be considered as giving a certain amount of information in the sense of information theory. Thus semantic and statistical information are interrelated: a certain freedom of choice is allowed in some "empty space" of a given predicate which specifies the semantic of the sentence.

The semantic theory [CAR56] defines truth and L-truth (logical truth) of sentences as follows:

In a system S an atomic sentence consisting of a predicate and a subject is true if and only if the individual to which the subject refers possesses the property to which the predicate refers.

A class of sentences in S, which contains for every atomic sentence either this sentence or its negation, but not both, and no other sentences, is called a <u>state description</u> in S because it gives a complete description of a possible state of the universe of individuals with respect to all properties and relations expressed by predicates of the system.

A sentence is L-true if it holds in all state descriptions. Two sentences are equivalent if both have the same truth value, that is both are true or both are false. Two sentences are L-equivalent if they hold in the same state description.

In this semantic theory the meaning of any expression is decomposed into two meaning components: the <u>intension</u> and the <u>extension</u>. Two sentences have the same extension if they are equivalent in S and have the same intension, if they are L-equivalent in S. The extension of a sentence is its truth value and the intension of a sentence is the preposition expressed by it.

For example, the extension of the sentence "John is ill" may be true or false and the intension expresses the fact that John is ill.

The extension is clearly related to the freedom of choice in the description (John may be ill or well.) The determination of the extension can be done by a scientific investigation and as such may be subject to uncertainty and error.

The intension of a predicate Q for a speaker X is the general condition which an object y must fulfill in order for X to be willing to ascribe the predicate Q to y.

Once the extension has been determined, the assigment of an intension to a sentence can also be considered as a matter of choice: in our previous example, assuming that somebody ignorant of the English language is told that the sentence "John is ill" is true, then the intension of the sentence might be chosen from all the propositions that are true for John.

The theory was later extended by Bar Hillel and Carnap into a semantic theory which conceptually parallels the statistical theory of communication [BARH64].

The important point is the fact that semantic information is also related to a certain freedom of choice within the state description of a semantic system.

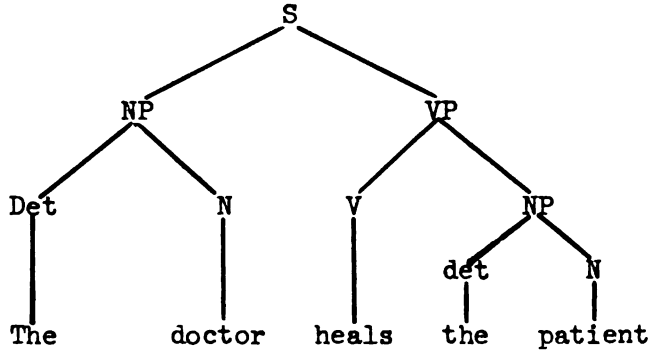
3.3 SYNTAX AND INFORMATION

Another type of structure is necessary to support the "content" of a sentence, namely the syntactic structure. The syntax is defined as a set of rules (grammar) that characterize the structure of the language. These rules are used to generate sentences in a determined fashion which involves the order of the elements in the sentences and the use of modifiers to indicate semantic nuances (plural, gender, tense, etc.)

A generative grammar is a system of rules that assigns structural descriptions to sentences. This system of rules is iterative and can generate an infinite number of structures.

According to Chomsky's syntactic theory [CHO65], the syntactic component of a grammar must specify a deep structure that determines its semantic interpretation and a surface structure that determines its phonetic interpretation. The surface structure is determined by repeated application of grammatical transformations applied to elementary objects contained in the base. The base is a system of rules that generates a highly restricted set of basic strings.

The deep structure can be represented by a tree diagram such as:



where the nodes of the tree contains either formatives symbols (the, patient...) and category symbols (S, NP, VP, V and N.)

The following set of rewriting rules can be used to generate the tree:

S → NP VP

VP → V NP

NP → Det N

Det → the

V → heals

N → doctor

N → patient

Such an unordered set of rewriting rules is called a constituent structure grammar or phrase structure grammar. Despite the simplicity of the example, it can be seen that the set corresponding to the first three rules enables the construction of a large number of sentences, hence the generative properties of such grammar. Therefore, here also the generation of syntactically correct sentences is a matter of selection of the rules to be applied.

However, some of the syntactically correct sentences may be inaccurate or meaningless; i.e. the sentence, "The patient heals the doctor" is not true in this situation and the sentence "The table heals the patient" is a nonsense statement.

Although a lot more is known about syntax than about semantics, it is very rare to see the word information associated with syntax and yet it would be impossible to translate (or understand) a language by using a dictionary without knowning anything about the syntax of the language. It is therefore obvious that a syntactic structure carries information: in a statistical sense by representing a particular configuration among several others allowed by the rules and in a semantic sense by giving

the relations existing between the elements of the sentence and the particular modifiers to be applied to the semantic concepts of the sentence.

It must be noted that syntactic structure also carries redundancies. For example, the semantic content of the sentence "The men are mortal" could be rendered as well by "The men (is) mortal" or "The (man) are mortal" where only one of either the subject or the verb indicates the plural instead of both in the correct sentence.

The fact that the syntax rules of programming languages are usually not redundant may explain the fact that a program may easily be misinterpreted by the compiler and is difficult to understand for someone who reads it for the first time. The so-called structured programming approach [DAH72] can be viewed as an attempt to add some "super-structure" to the existing syntactic structures in order to make programs understandable by adding some redundant structural information.

Finally, another type of structure relating to information is known as "data structures" and is used to define structural entities in which data will be stored. This type of structure is to data what syntax is to the semantics of a language - it supports the informational content of data. Examples of such structures are linear lists, tree, arrays, files, [KNU68], etc.

There also the structure may be redundant (two way linked lists) in order to provide for a possible reconstruction of the structure in case of its destruction. Now it should be clear that information has several facets which are statistical (communication theory) semantic and structural (syntax.)

To conclude we may draw an analogy with the basic process of molecular biology. The DNA can be considered as the data base for the

genetic information: a DNA molecule carries "free choice" information in the sequence of nucleotides but it also contains structural information necessary for its replication (geometric as well as "syntactic:" pairs G-C, A-T.) The communication of this data involves the messenger RNA which assumes the role of a "channel" which transmits coded information between the nucleus and the cytoplasm. Then the information will be processed by the transfer RNA which can be viewed as a machine that transforms the input codes of the RNA into outputs of polypeptide chains of aminoacids. This phase involves the processing of the "semantic" information contained in the RNA codon (a sequence of three nucleotides that code for an aminoacid) to generate the corresponding protein.

3.4 MEDICAL INFORMATION

In view of the previous chapter, this section will try to delineate the particular characteristics that information may have in the field of medicine. Hopefully, the recognition of these characteristics might help in designing systems adapted to the processing of this information. Although some purists will argue that medical information is not different from any other type of information, an attempt will be made to define and describe the peculiarities of "medical information."

3.4.1 SOURCES OF MEDICAL INFORMATION

It was previously noticed that information could not be obtained without observation (freedom of choice) or experience (semantics and pragmatics of language.) It follows that the principal sources of medical information will be the "patient" and the medical professionals (physicians, nurses, technicians, etc.) There is a qualitative difference between observation made on a human being and observation made on a physical system: observations on both physical (mechanical) and bio-

logical systems may be impeded by inaccessibility, the striking difference is the complexity of the latter which means that the analytic description of a biological system involves an enormous number of degrees of freedom, not all of which are recognized or known.

3.4.2 THE NATURE OF MEDICAL INFORMATION

The fact that quantitive data is difficult to obtain is compensated by a descriptive vocabulary which helps either to locate the observations (anatomy), to describe the abnormalities (pathology), and to define normal and abnormal states of a patient (diseases) or a population (epidemiology).

The complexity of the subject has a multiplying effect which gives to the field of medicine the widest specialized vocabulary. In addition, all the common vocabulary is available for the report of verbal interviews with the patient (history, complaints, etc.) as well as for progress notes, visual observation, etc.

Despite the increasing number of laboratory tests which give results in the form of quantitative data (numbers), most of the medical data is in the form of language data. This explains the important role played by the medical record in processing medical information. To our knowledge there is no study which gives the percentage of language data versus numerical data in a typical medical record but it is probably about 90%.

Therefore, most of medical information belongs to the pragmatic level of information; it is non-numeric, complex and encyclopedic. The medical record is a document used to record observations, problems, facts and historical data about the patient, progress notes containing the most salient aspect of the treatment given, test results and reports. The archival nature of the medical record requires an unlimited memory medium.

3.4.3 THE QUALITY OF MEDICAL INFORMATION

By quality we mean "accuracy" in the transcription of live information into data on a memory medium. With the current manual record, not only the quality may vary with the professional's experience and knowledge, but also with his ability to and willingness to write legibly. Even with all the redundancy of natural language, there is normally a non-negligible loss of information due to incompleteness and illegibility. On the other hand, the 50% redundancy of a natural language is quite often a luxury when the object is to record useful data.

Some effort has been made in the direction of systemization and coding of nomenclature in pathology (SNOP) and it shows that the information (in the semantic sense) present in pathology diagnosis can be represented using the systematized nomenclature of pathology (SNOP) and exercising the necessary selection within four structured lists:

- Topography (T)
- Morphology (M)
- Etiology (E)
- Function (F)

The advantages of such an approach are immediately evident: structuring of the semantic data into a much simpler structure than the normal English grammar, reducing of the redundancy, possibility of direct coding for computer manipulation, etc.

It is certainly conceivable to create such systematic nomenclature for other specialities of medicine. By looking at a medical record, it is striking to see the number of abbreviations (often improvised and non-standard) used to reduce the writing overhead and the number of stereotyped sentences which shows a natural tendency to use the same phraseology.

Some efforts in this direction have already been made by using the International Classification of Diseases Adapted code (ICDA) for discharge summaries.

However, despite the fact that such a trend is visible and even if it were proved that the same semantic information could be carried by such languages, it will be impossible to enforce their use in the daily activity of an hospital because, with a manual data entry system, it would be too constraining to look up terms in lexicons and to follow rigid syntactic rules. Given the human constraints of the medical environment, the key factors are the quality of the information and the ease of transcription of information.

The problem is how to obtain quality medical information with a minimum of constraints, and a minimum of effort. Stated in the language of operations research, the question is: how to maximize quality given the existing set of economic and psychological constraints.

3.4.4 COMMUNICATION OF MEDICAL INFORMATION

The quality of medical information is clearly related to the means by which it is transmitted from one point to another (from the medical observer at the bedside to the different active centers necessary for the collection or transcription of data and vice-versa.) This is a pure communication problem and it accounts for an important part of the clinical information handled within a hospital [BL071]. (We avoid the term processing because communication is not a processing of information but rather a transmission of information between the different processing centers.) The processing (i.e., translation and aggregation) of medical information is for the most part done by medical professionals (however, for numerical results, an ever growing part is done by machines.)

If one considers the communication problem, each processing center can be viewed as a black box M_i with some input values $a_1^i, a_2^i, \ldots a_n^i$ (numeric or linguistic) and some output values $b_1^i, b_2^i, \ldots b_m^i$.

$$-a_1^i \ldots a_n^i \rightarrow \boxed{M_i} - b_1^i \ldots b_m^i \rightarrow$$

If the reliability of each Mi is Gi (Gi≤1) the overall reliability of the system is defined as $G = \frac{\sum_{i=1}^{k} Gi}{k}$ for k components. It is clear that as the number of processing centers increases the total reliability can only decrease if the reliability of each component does not increase. (This is usually done by adding more redundancy in the system.)

Following Hsieh [HSI66], we can define a measure of quality of a communication process by:

Q = conditional probability of right output, given the right input

1-Q = conditional probability of wrong output, given the right input

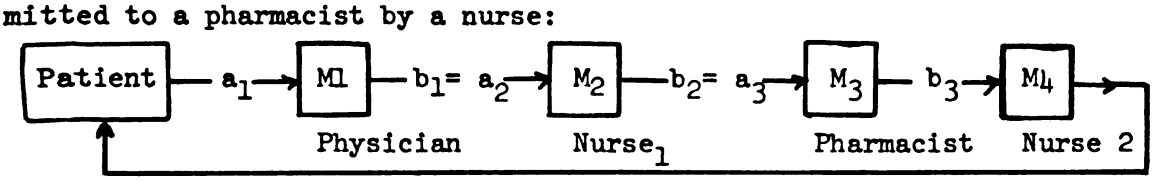
and a measure of the correction capability of an element by:

R = conditional probability of right output given the wrong input

1-R = conditional probability of wrong output given the wrong input

R is related to the redundancy because a given wrong input can be rectified into a right output.

Now if we consider the following example of a clinical medication system where the medication order is generated by a physician and transmitted to a pharmacist by a nurse:



The reliability of M_1 (physician) is: $G = Q_1$.

The reliability of M_2 (nurse) is: $G_2 = Q_1Q_2 + (1-Q_1)R_2$. $(1-Q_1)R_2$ is the correction capability of M_2 (wrong output from M_1 is rectified.)

The reliability of M_3 (pharmacist) is:

$$G_3 = Q_1Q_2Q_3 + Q_1(1-Q_2)R_3 + (1-Q_1)R_2R_3 + (1-Q_1)\ (1-R_2)R_3$$

$Q_1(1-Q_2)R_3$ is an error correcting term for a right output from M_1, an error by M_2, rectified by M_3. $(1-Q_1)R_2Q_3$ corresponds to an error from M_1 transmitted by M_2 and corrected by M_3. $(1-Q_1)(1-R_2)R_3$ is an error from M_1 transmitted by M_2 and corrected by M_3...

The last element of the chain (the nurse giving the medication to the patient) is therefore the one that has to do the most checking if quality of information is to be preserved. She is dependent on the errors made by the first three elements. This shows that in a communication system, the information can only be degraded by going from the source to the destination. However, if some redundancy is carried by the data and if the processors of information are intelligent, then it is possible to restore the original information.

With a computer system, it is possible to eliminate some of the communication errors by direct transmission of orders from the physician to the pharmacist.

3.4.5 MEDICAL INFORMATION AND ALGORITHMIC PROCESS

"I ran up against an absolute blank wall...you cannot do that because computers can only do arithmetic, they cannot write programs." Well, I got mad at this and arbitrarily stated that, "I could make a computer do anything which I could define." Grace Murray Hopper in Computer, October 1973.

Although it was pointed out that medical data is mainly linguistic, it should not be concluded that medical information cannot be processed by computing machines. It may be true that it will be more difficult to process medical information than numerical information because the machines and the programming languages are usually more adapted to numeric computation. However, computers can also perform non-numeric operations and memorize facts with perfect accuracy.

To a certain extent, the term computer is an unfortunate name for machines that are essentially algorithmic machines. An algorithm is defined by Knuth [KNU68] as a set of rules which give a sequence of operations for solving a problem and which has the following features: it is finite (it must terminate after a finite number of steps), it is definite (each step must be precisely defined), it has zero or more inputs, and one or more outputs, it must be effective (all the operations can be done in a finite length of time). According to this definition, a medical diagnostic process is fundamentally algorithmic (it may be subject to uncertainty, but the uncertainty reflects only the incompleteness of the data or the insufficient development of the medical sciences).

It is a fact, however, that computers and programming languages have been tailored to implement numerical algorithms (every computer has an instruction to perform the addition algorithm, but very few have instructions to compare or move strings). This may explain part of the difficulty to build medical information systems: one has either to use the inadequate tools available (current instruction sets, languages, operating systems, etc.) or to invest time and effort in the building of more appropriate tools.

In summary, it is not because medical information is mainly linbuistics that it is difficult to implement the basic algorithms which are used in medical information processing, but because most of the current computer technology and software are not adapted to the efficient realization of such algorithms.

It is true that some modern machines have partially resolved the gap in the hardware instruction set, but the inadequacy of the operating systems and present day languages is still a reality. It would be false, however, to state that medical information cannot be processed efficiently by present day computers, but it is important to recognize their inadequacies and the necessity of building new software tools to realize the abstract machine that is more adapted to the user's problems.

Finally, it is desirable to get away from the concept that the medical care processes are an "art" that cannot be aided or handled in part by machines. It is certainly true that a machine will not yield a better diagnosis than a good physician if it is programmed by a less than average physician. However, if a good physician is willing to explain the rational process that he follows when doing his work in such a way that it can be algorithmically described, then a machine could give a better diagnosis than the average physician or at least help the average diagnostician to become a better one.

This may sound too simplistic because a machine can only reproduce known algorithms and the flexibility and creativity of a human being is certainly irreplacable. On the other hand, human memory and ability (accuracy and speed) to compute and remember accurately is certainly more limited than that of a computer. The point is that it is possible to build algorithmic models [GOR67] for most of the standard diagnoses

and treatments. It will be seen later how the concept of frame selection system, where the algorithm is a pragmatic set of rules and can be fuzzy, (in the sense of Zadeh [ZAD73]) could be used to allow flexibility in the decision process involved in medical data processing.

In conclusion, since medical information is strongly related to language, the main problems in medical information processing will be related to that peculiarity. The medical language currently used is a collection of specialized subsets of semantic terms (anatomy, physiology, pathology, pharmacology, etc.) which are related to each other by a syntactic structure which is usually the syntax of a natural language.

However, it is possible that although medical information is mainly "linguistic", the same semantic information could be carried by a much simpler syntax. Such a syntax could not be enforced or even desirable with a traditional type of medical record, but the development of computerized medical records might be the occasion to define more systematized languages to be used for computer manipulation of medical information.

Chapter 4

Formal Description of Frame Selection Systems

"A new scientific truth does not triumph by convincing its opponents and making them see the light, but rather because its opponents eventually die, and a new generation grows up that is familiar with it."

Max Planck

4. FORMAL DESCRIPTION OF FRAME SELECTION SYSTEMS

In the previous chapter it was noted that the information I, as defined by Shannon, was intimately related to the freedom of choice one has when presented with a given situation: if there is no choice (certainty) the information I is zero; if the choice is binary (YES or NO) the information I is equal to unity. For a language described by an alphabet, the information carried by the messages or words built from this alphabet can be measured by the freedom of choosing a sequence of letters or symbols contained in the alphabet. In other words the accumulation of information is done by selection of an individual object (a letter, a word, etc.) from a given set of possible objects or the selection of an alternative among a set of possible choices.

4.1 SELECTIONS PROCESSES AND FORMULAS

A selection process generates information each time it operates in a situation where the number of choices or alternatives is at least two. We exclude the case where the number of alternatives is infinite for practical purposes, but there is no definite upper bound on the number of alternatives.

4.1.1 THE SELECTION OPERATOR AND THE FREE SEQUENTIAL SELECTION PROCESS

A selection is a monadic operation on a finite set or a collection of sets. A free sequential selection process is defined as the process of choosing an element from a set or a collection of sets where there is no semantic restriction on the choices made at each successive time interval.

A set A is defined as: $A = \{x \epsilon A : x \text{ is ill}\}$ which denotes the set of sick persons. The order of a set is the number of elements in the set and noted $|A|$. The selection operation will be denoted by the symbol $\square$.

It must be emphasized that selection is a function operating on a set or a collection of sets. Therefore, it is not an operator on elements of a set, nor is it an operation of the type union, intersection or complement defined on sets.

If A is a set, $\square A$ will denote the selection of an individual element of the set. If C_A is a collection of sets, $\square C_A$ will denote the selection of one set from C_A. A free sequential selection process can therefore be described by:

$$\square A_1 \ \square A_2 \ldots \square C_i \ \ldots \square A_K \ \square C_m \ldots \square A_Z$$

where the Ai are sets and the C_i represents collection of sets.

If $Ai = \{a_i^1 \ldots a_i^j \ldots a_i^m \ldots a_i^n\}$ then the selection of the element a_i^j from the set Ai will be denoted by $\square Ai = a_i^j$ and if C_K is a collection of sets $C_K^1, C_K^2, \ldots C_K^P$ then $\square C_K = C_K^j$ will indicate that the set C_K^j has been selected.

4.1.2 PROPERTIES OF THE SELECTION OPERATOR

The selection operation is associative; that is,

$$(\square A \square B) \square C = \square A \ (\square B \square C)$$

If we consider the union of sets $A \cup B$, the selection of an element of $A \cup B$ is equivalent to the selection of an element of A or an element of B. Therefore, $\square (A \cup B) = \square A$ or $\square B$.

More generally, $\square (A \cup B \cup C \ldots \cup Z) = \square A$ or $\square B \ldots$ or $\square Z$.

The cartesian product of the two sets is defined by:

$$A \times B = \{(a,b): a \text{ in } A \text{ and } b \text{ in } B\}.$$

This is equivalent to a double selection: one element from set A, and one element from set B. Therefore, $\square (A \times B) = \square A \square B$.

If a selection process operates only on sets, a sequential selection process is equivalent to a generalized cartesian product. If a

selection process operates successively on the same set or collection of sets, then $\square^n A = \underbrace{\square A \ldots \square A}_{n}$. If the number of selections that can be made on the same set is unknown, then $\square^* A = \square A \ldots \square A$. If f is a bijective function operating on a set A, then $f(\square A) = \square f(A)$.

When a sequence of selections is repeated n times during a selection process, the notation $(\square A_1 \square A_2 \ldots \square A_m)^n$ will be used. If the number of iteractions is unknown before starting the selection process, $(\square A_1 \square A_2 \ldots \square A_m)^*$ will be used. It can be shown that $\square^n A = (\square A)^n$ and $\square^* A = (\square A)^*$. (If $A = \{a_1, a_2 \ldots a_m\}$ then for $i_1, i_2 \ldots i_n \varepsilon \{1, \ldots m\}$ $\square^n A = a_{i1}\, a_{i2} \ldots a_{in} = (a_{i1})(a_{i2}) \ldots (a_{in}) = (\square A)(\square A) \ldots (\square A) = (\square A)^n$

4.1.3 <u>SELECTION FORMULAS AND SELECTION SENTENCES</u>

- A selection formula is defined as a repeated application of the previous rules using the properties of the selection operator.
- A selection sentence is represented by the sequence of objects or alternatives chosen during a sequential selection process.

An example of a selection formula is:

$$S = \square A_1 \ldots \square A_m (\square A_p \ldots \square A_r)^n \square A_s{}^* \ldots \square A_2$$

Selection formulas are a concise and precise way of describing sequential selection processes just as arithmetic or algebraic formulas are concise representations of sequential computational processes. A selection sentence is a particular result of a selection formula when the sequential selection process is carried out.

If $A_i = \{a_{1i}, a_{2i}, \ldots, a_{mi}\}$ a selection sentence corresponding to $\square A_1 \ldots \square A_n$ can be $a_{j1} a_{k2} \ldots a_{zn}$

<u>Examples</u>: Besides the simple example of menu selection, more sophisticated sequential selection processes are given by:

- decision making processes (including the diagnostic process.)
- multiple answers questionnaire

- strategies and games
- evolution

If D= {0,1,2,3,... } then:

i) $\square^n D$ will denote the selection of a succession of n positive integers and the corresponding selection sentences represent all the positive integers of n digits.

ii) $\square^* D$ will generate any positive integer.

iii) Let A,N,V denote the sets of articles, nouns, and verbs respectively:

A= {x:x is an article}

N= {x:x is a noun}

V= {x:x is a verb}

Then $\square A \square N \square V \square A \square N$ can generate simple sentences of a natural language (we are not concerned here with their meaning.)

iiii) If T= {x:x is a topographic term}

M= {x:x is a morphologic term}

E= {x:x is an etiologic term}

F= {x:x is a function term}

Then $\square T \square M \square E \square F$ is the selection formula which generates all the sentences of the systematized nomenclature of pathology [SNOP] .

4.2 SELECTION FRAMES

So far we have not considered the problem of the feasibility of selecting an item from a given set. In practice, we will define subsets with a maximum of selectable elements (corresponding to a given implementation.)

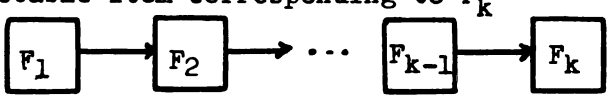
A frame will be defined as a set of semantic or pragmatic terms s_i such that FR= $\{s_1 \ldots s_k\}$ with $|FR| < MAX$ where MAX is the maximum possible number of selection points on a CRT.

A finite set A can always be decomposed into a set of frames by breaking down A into subsets which have less than MAX elements. This decomposition can be achieved using the following two methods.

4.2.1 HORIZONTAL OR SERIAL DECOMPOSITION

This method divides a set A into K frames such that K= $|A|/MAX-1$ and each of these frames introduces an artificial choice that can be selected to indicate that the desired item is in the complementary set of the set represented on the frame.

If F_k' denotes the complement of the frame Fk in the set A, then A can be represented by a linear list of frames linked together by a default selectable item corresponding to F_k'



This is called the horizontal extension of A and is denoted A^h. It can also be called serial decomposition.

4.2.2 VERTICAL OR PARALLEL DECOMPOSITION

Another method of decomposition can be defined by breaking the set into K frames such that K= $|A|/MAX$ and then create an "index" frame which will be used to select the proper frame or subset. The structure used is a tree:

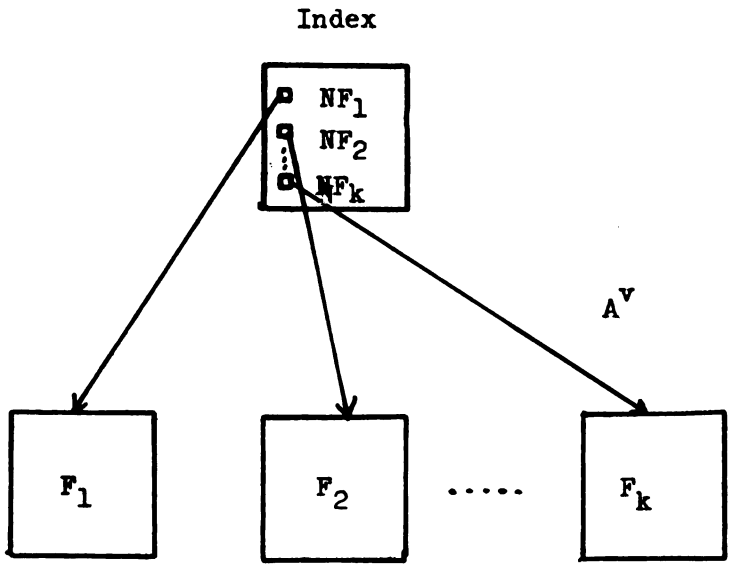


Figure 4.1 Vertical Decomposition

The index is a frame which indicates the names of the subsequent frames: NF_1 is the name of F_1, NF_2 is the name of F_2 and NF_K is the name of F_K.

To select an element of the frame F_i, one selects the corresponding NF_i in the index and then the desired element within the frame F_i.

The decomposition can have more than one index level if necessary. This is called the vertical extension of A and noted A^v.

4.3 RELATIONS BETWEEN SELECTION FRAMES

4.3.1 BINARY RELATION

Selection processes where some constraints are imposed on the selections are now considered. This can be done by introducing the concept of binary relation. Formally a relation is a set R of ordered pairs: $(a,b) \varepsilon R$ also written aRb where $a \varepsilon A$, $b \varepsilon B$.

Thus, the selection of elements in A and B are constrained by the relation R. This defines a constrained selection process under the relation R. The image of an element $a \varepsilon R$ is the subset of B defined by $R(a) = \{y: \text{every } y \varepsilon B \text{ such that } a\ R\ y\}$. The inverse image of $b \varepsilon B$ is the subset of A defined by $R^{-1}(b) = \{x: \text{every } x \varepsilon A \text{ such that } x\ R\ b\}$, if $\square A = a$, then $\square B$ must be restricted to $\square R(a)$; or conversely, if $\square B = b$, then $\square A$ must have been restricted to $\square R^{-1}(b)$. The domain of a relation R is defined as the set of all first elements x of the ordered pair (x,y). The range of the relation R is defined as the set of all second elements y of the ordered pair (x,y). These concepts are important for the semantic relation between sets of semantic terms.

For example, two frames, F_1 and F_2, may be related by using a matrix of semantic compatibility defined as follows: if the element a_i^j of the matrix is zero, the term c_j of F_2 cannot be selected after the selection of c_i in F1; if a_i^j is not zero, then the two elements c_i and cj can be selected during a selection process.

4.3.2 EQUIVALENCE CLASSES AND PARTITIONS

A binary relation R is called equivalence relation on a set A iff R is reflexive, symmetric and transitive that is:

xRx (reflexive)

xRy implies yRx (symmetric)

xRy and yRz implies xRz (transitive)

The subsets E_x of A such that $E_x = \{y: yRx\}$ are called R- equivalence classes.

The collection of equivalence classes under an equivalence relation R_E is called the quotient of A with respect to R_E: A/R_E. Equivalence classes of x under R_E are sometimes noted $[x]/R_E$. Equivalence relations partition a set into disjoint subsets which are the equivalence classes.

Conversely, by defining a partition P of a set as a collection of non-empty disjoint subsets of A whose union is A, then the <u>relation induced</u> by P written A/P is defined by: xA/Py if x and y belong to the same partition. It can be easily verified that it is an equivalence relation. Therefore, partitioning a set is equivalent to defining an equivalence relation on the set and conversely.

This makes it possible to select an element of a set by first selecting the equivalence class containing the element x and then the element inside the equivalence class by a two step selection process.

$$\square A = \square\{C_a[\square E_a],\ C_b[\square E_b],\ldots,C_x[\square E_x]\}$$

This is called an <u>imbedded selection formula</u>. The brackets [] are introduced to indicate that the selection of a class in A/R_E (C_x is a class defined by a partition of A) is followed by a selection inside the class E_x if the class of x was selected in the first step.

This introduces the notion of <u>depth</u> in a selection process. The

depth is the maximum of imbedded bracket pairs encountered in a developed selection formula.

Example: Suppose A= {x:x is a living organism}

A partition of A is given by the fauna (F_1) and flora (F_2) and the corresponding classes are C_{F1}, C_{F2}. F_1 can then be partitioned into two new classes, C_V and C_I for vertebrates (V) and invertebrates (I).

V can be decomposed into fish (F), amphibians (A), reptiles (R), birds (B), mammals (M) and the corresponding classes C_F, C_A, C_R, C_B, C_M. The selection of any living organism can be done by using the following developed formula.

$$\square A = \square\{C_{F1}[\square\{C_V[\square\{C_F[\square F], C_A[\square A], C_R[\square R], C_B[\square B], C_M[\square M]\}], C_I[\square I]\}], C_{F2}[\square F_2]\}$$

Although C_I and C_{F2} could also be partitioned the selection process as defined by this formula has depth 3.

Therefore, a selection process can be decomposed into a sequence of selections steps operating on classes or partitions of the original sets. The construction of these classes is usually done on a semantic or pragmatic basis so that the partitioning reflects the semantic relations or differences between the elements and subsets of the original set. So by exerting the freedom of choice on such subsets or collection of sets, one can introduce a certain amount of semantic information which is contained in the relation defined by the partitioning.

Although this type of information has never been quantified, it can be measured by the depth of the selection process necessary to identify the object selected. Starting with a semantic information content of unity for the universe, one can define classes of objects (abstract or real) and by using the partitioning method one ends up with a measure of semantic information for the elements contained in those classes. For example, the terms used in the systematized nomenclature of pathology

[SNOP] could be associated with such a measure.

4.3.3 ORDER RELATIONS AND SELECTION FRAMES

Another type of relation defined on sets is the order relation.

A reflexive partial order on a set A is defined by a relation R such that: for every x,y,z in A

- x R x (reflexive)
- x R y and yR x implies x = y (antisymmetric)
- x R y and yRz implies xRz (transitive)

A reflexive total order is such that for every x and y in A then either xRy or yRx. An ordered set can be decomposed in a series or parallel collection of frame $F_1 \ldots F_k$. containing elements such that $x_{iy} R y_{kl}$ for all x_{iy} in F_i and all y_{kl} in F_k.

- a serial decomposition verifying this order relation will be called a frame chain and noted $\overline{F}_I$ where I= {0,1...k}.
- a parallel decomposition verifying this property will be called a frame fork and will be noted $\hat{F}_I$ where I= {0,...k}. If k>MAX then I can itself be decomposed in a frame fork or frame chain.

In order to select an element of a frame chain or a frame fork, one has to select the frame which contains the elements and then the element. In a frame chain if c_0 represents the choice that the element is not in the current frame then: $\square \overline{F}_I = c_0{}^k \square F_k$ if the element selected is in the kth frame of the chain. In a frame fork $\square \hat{F}_I = \square C_F[\square F_k]$ where C_F represents the collection of ordered frames obtained by the fork decomposition. The frame fork is more appropriate when the number of elements in the set is great. The frame chain is appropriate when the number of frames in the chain is low or when the chain itself is ordered so that the most likely selections are contained in the first frame and the

following frames contain the elements with a low expected frequency of selection. Example: Suppose D= {x: x is a drug name} and D is ordered lexicographically, then it is possible to decompose D into a collection of frames which can be accessed by selecting the alphabetical range corresponding to the elements contained in a given frame or by "paging" through the frame chain.

In practice, if the list to be searched is small (2 or 3 frames) a chain will be used, but if one wants to access a formulary of 2000 items, a fork decomposition is more suitable.

4.3.4 LINGUISTICS RELATIONS AND SELECTION FRAMES

The previous approach may seem too rigid because it dealt with the mathematical concepts of sets, equivalence classes and relations.

Since it was previously emphasized that medical information was mainly linguistic it is useful to see how this characteristic can be related to the concept of linguistic variables. The concept of a linguistic variable was introduced by Zadeh [ZAD73] and relates to variables whose values are not numbers but words or sentences of a language. For example, the temperature of a patient may be qualified by the terms, high, low, normal, very low, very high, etc. which may be called the linguistic values of the linguistic variable temperature (although, in this case, the temperature could also be measured by numbers.) The value of a linguistic variable is called its term-set. For example: temperature= {low, high, normal, almost normal, above normal, very low, very high,...} It can be seen that the selection operator can be applied to the term-set of a linguistic variable. For example: □ temperature - will select a particular value from the term-set temperature as defined above. Therefore, all the concepts already defined: selection process, selection formulas, frames, can be applied to linguistic variables and term-sets.

A term-set can be considered as a fuzzy partition of the linguistic concept represented by the linguistic variable. Closely related to this notion is the concept of fuzzy set also introduced by Zadeh [ZAD65] and defined as a set of objects A with a membership function f_A which associates with each object a real number in the interval [0,1]. An example of fuzzy set can be given by the "set of dangerous drugs." Such a definition leads to fuzzy boundaries where a given drug might be considered not dangerous at a low dose and clearly dangerous at a higher

dose, not dangerous for a well person but dangerous for a given patient. A function f_D might give the membership of each drug in this fuzzy set. The idea introduced by fuzzy set is that despite the fuzziness of linguistic concepts, it is possible to associate a number which quantifies the membership of an element to a class.

For instance, a poisonous substance may have a membership function $f_d = 1$ in the set of dangerous drugs and aspirin may have a membership of 0.01. This membership function can be a discrete function associating a number which quantifies the toxicity of a drug which can be considered as a linguistic variable.

It is also possible to introduce fuzzy relations and ordering of elements of fuzzy sets in the same way as for simple sets. Although much of the concern of the theory developed by Zadeh is to introduce a calculus in the context of fuzzy sets [ZAD73], fuzzy relations, etc., our concern is to point out that a practical way to implement linguistic variables and fuzzy sets is via the use of the frame concept. Each of the concepts defined previously in the context of set theory is applicable to fuzzy sets. The membership function of a fuzzy set can be used as an ordering relation of elements within the fuzzy set therefore, making it possible to decompose fuzzy sets into a collection of ordered frames containing the elements of the fuzzy sets.

In a medical information system, whenever a linguistic variable or a fuzzy set is selected from a collection of other linguistic variables or fuzzy sets the user could be asked to select the particular instance of the term-set or the particular element of the fuzzy set. If L is a set of linguistics variables $L_1, L_2 \ldots L_k$ and T_i is the term-set of L_i, $\square L[\square T_i]$ will denote the selection of a linguistic variable and a par-

ticular value of its term-set T.

A fuzzy set can be decomposed in frame chains or forks by using the membership function to order the elements. For instance, the set of dangerous drugs can be decomposed into frames whose membership function will be between 0 and 0.1, 0.1 and 0.2,...0.9 and 1. In that case, the access to these frames could be done by using an index frame representing fuzzy partitions of toxicity such as: poisonous, very very dangerous, very dangerous, dangerous, not dangerous, etc. Furthermore, the introduction of selection processes to select linguistic variables enables the introduction of a semantic context between the variables; the toxicity of a drug can be modified by the patient's allergies or the other drugs he might have taken recently so that the selection of an item might modify the membership function of a given drug to the fuzzy set of dangerous drugs for a given situation.

4.4 SEMANTIC COMPLEXES AND FRAME STRUCTURES

Now that the intuitive ideas have been introduced, the concept of frame will be defined more axiomatically. It is assumed that the universe of discourse U is composed of a finite set of semantic and pragmatic terms or phrases of a given language. For instance, this universe of discourse can be the medical language or a subset of it such as the language of pathology, anatomy, physiology, etc.

Certain finite subsets of U are called frames if they satisfy the following conditions:

- each frames contains at least one element and at most MAX elements
- every subset of a frame is a frame
- every frame defines a new object called the semantic simplex spanned by that frame (it can be associated either with a set, a fuzzy set or a linguistic variable.) The semantic simplex corresponds to the intension of the frame.

The semantic simplexes associated with sub-frames are called semantic facets of the simplex. The dimension of a simplex is equal to the number of its elements minus one. A simplex containing one item has a dimension of zero because there is no freedom of choice. A simplex with two elements has a dimension of one and so on.

The introduction of the selection operator enables the selections of elements or facets of a semantic simplex. Information-wise this operation separates clearly the semantic information within the semantic simplex and the information related to the freedom of choice in the semantic simplex which is only related to the dimension of the simplex. A finite system of semantic simplexes is called a semantic complex of the universe of discourse. Such complexes can be represented as lattices

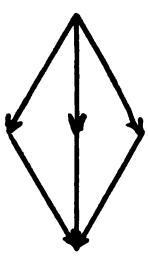
or directed graphs.

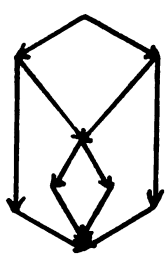
Considering a set of frames F, a partial order can be introduced on the set of frames such that $f \leq g$ if f precedes g in a sequential selection process involving the semantic simplexes of f and g.

An element t of F is said to be an <u>upper bound</u> for a subset A of F if $t \geq a$ for every a in A. t is a least upper bound if t is an upper bound and $t \leq u$ for any upper bound u of A. A similar definition will define the greatest lower bound.

Definition: A <u>lattice structure</u> is a partially ordered set in which any two elements have a least upper bound and a greatest lower bound. The <u>join</u> of f and g is the least upper bound of f and g noted $f \sqcup g$, and the <u>meet</u> of f and g is the greatest lower bound noted $f \sqcap g$.

A lattice can be represented by diagrams such as:





<u>Example</u>: Considering a typical drug order represented by the following selection formula:

$\Box \hat{D}[\Box DF]\Box R\Box F$ where $\hat{D}$ is the frame fork corresponding to the set of drug names. DF is the set of frames for the dose forms. F is the frame for frequency of administration for the dmug. R is the frame for route of administration for the drug.

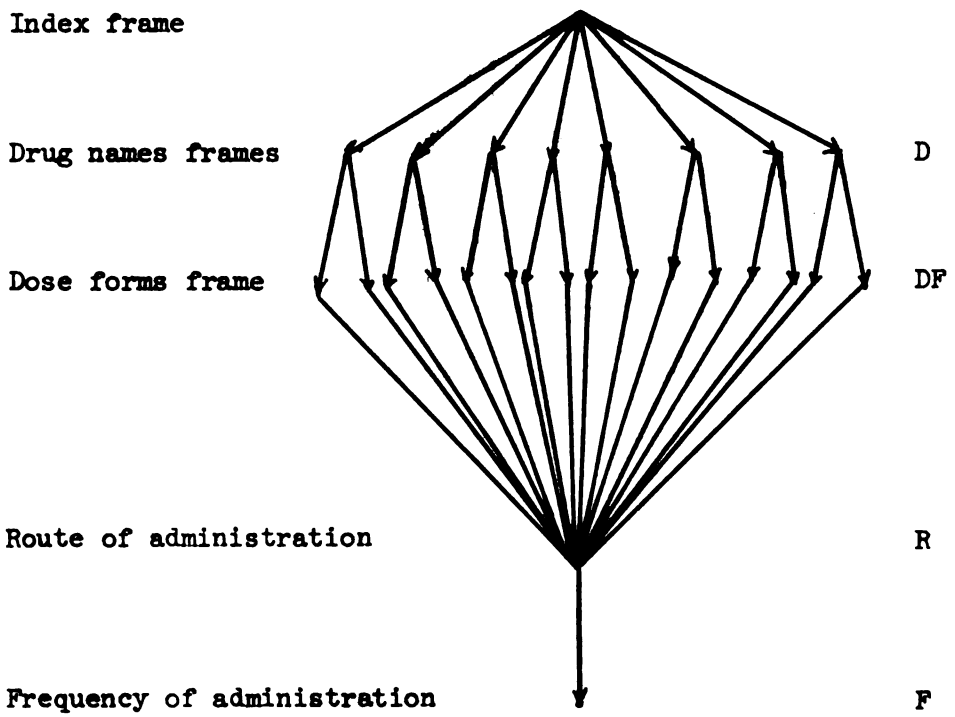


Figure 4.2 Frame Structure for a Drug Order

This frame structure is a lattice with a least upper bound represented by F and whose greatest lower bound is represented by the index frame to access the drug names. It can be seen that this structure is more general than a tree structure. Such a lattice represents the semantic complex referred to generally as drug order in the medical language. A lattice structure can represent any selection process where there is no need for loops on some sequence of frames. Therefore, the frame lattice structure can be considered as the basic building block of semantic frame complexes.

The use of loops can be introduced by a feedback mechanism in the lattice structure. For example:

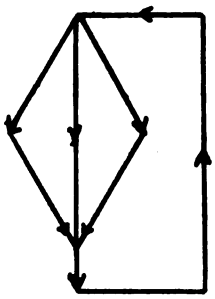


Figure 4.3 Frame Digraph

The feedback loop is important for its repetitive function but it does not add any depth in the selection process, its only purpose is to use a structure that already existed to obtain some new semantic information.

More generally, the introduction of loops in frame structure can be studied by using the concept of directed graph or digraph [HAR65] that is a collection of ordered pairs (a,b) of elements of a set of nodes N .

Up to this point, the frame concept was studied from a static point of view. It was seen that semantic complexes can be broken down into semantic simplexes called frames. Conversely, the structure to build the complexes from the simplexes can be lattice structures which can be extended to digraph structure by the use of loops.

The following is a study of the dynamics of frame selection systems, their relation to finite state machines and how they can be used to generate well formed sentences of a language.

4.5 FRAME SELECTION SYSTEMS AND FINITE STATE MACHINES

4.5.1 FRAME INPUT CODES AND SELECTION STRINGS

A frame has been defined as a finite set of items (choices or alternatives in U) with an upper bound on the number of selectable elements.

The frame items can therefore be described by a finite alphabet of symbols

$$C = \{c_0, c_1, c_2 \ldots c_{max}\}$$

where max is the maximum number of selection points on a CRT. This alphabet is used to code each possible selection on a frame; these selections can be considered as input codes which are associated with the selectable elements on the frame. A sequential selection process can therefore be represented by a sequence of letters from the alphabet C.

SELECTION STRING (OR WORD)

If C^* represents the set of all the strings over the alphabet C, then the execution of a selection process can be represented by a string of symbols of C, each symbol representing the choice which has been selected in each semantic simplex (frame) during the selection process:

$$c_{i1}, c_{i2}, \ldots, c_{ik} \quad ik \in \{0, \ldots max\}$$

this string can be considered as having a meaning in the semantic complex associated with the selection process. Another way to look at such a string is to consider it as a particular representation of the selection formula representing the selection process (just as $2^2 + 3 \times 2$ is a particular representation of the formula $X^2 + 3x$).

CONCATENATION OF SELECTION STRINGS

Given two strings s and t in C*, we define the operation of concatenation noted * by associating the string st to s*t.

If $s = c_{i1}\, c_{i2} \cdots c_{ip}$

and $t = c_{j1}\, c_{j2} \cdots c_{jq}$

then $s*t = c_{i1}\, c_{i2} \cdots c_{ip}\, c_{j1}\, c_{j2} \cdots c_{jq}$

such an operation satisfies the property of associativity

$$(s*t)*u = s*(t*u)$$

A semigroup is defined as an ordered pair (S,.) where S is a set and . is a mapping of SxS into S such that $(s_1.s_2).s_3 = s_1.(s_2 s_3)$ for all s_1, s_2, s_3 ϵS. A monoid is a semigroup containing an identity element.

Then, by definition, C* is a semigroup under the operation of concatenation. If one includes in C* the empty string e, then e*s = s*e = s, e is an identity and C* is a monoid.

4.5.2 FRAME SELECTION SYSTEMS AND ABSTRACT MACHINES

Algebraic machine theory [ARB68] has usually been used to model the action of logical hardware rather than programs or computer systems. However, it does not model the action of such devices directly, but only through the movements of abstract "states".

This chapter will show that the theory is directly suited to the description of frame structured systems in terms of their software capabilities as well as for a model to describe the behavior of such systems.

The following is a definition of an abstract machine called the selecton which directly mimics the behavior of frame selection systems. It must be emphasized that this theoretical approach is not just an attempt to "re-discover" automata theory, but rather to show how well

the models that have been conceived abstractly are really a direct representation of the behavior of frame selection systems which have been conceived from a pragmatic point of view. It is even more striking to see that the "selecton" model is closer to the real implementation of frame selection system than the automaton model is to logical hardware design.

THE INTUITIVE APPROACH

A machine is considered as a black-box with inputs and outputs.

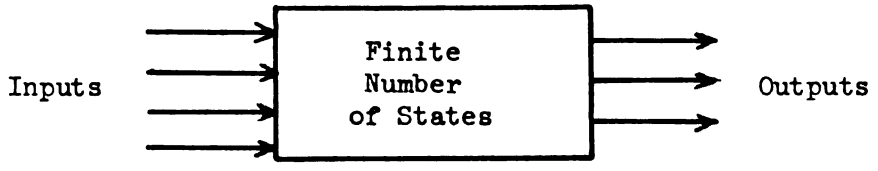


Figure 4.4 Finite State Machine

The "inside" of the black-box is characterized by a finite number of states and the behavior of the machine is described by two functions:

- the action function gives the next state knowing the current input and the current state
- the output function gives the output knowing the current state and the current input. (If the output depends only on the state, it is called "state-output").

These functions can be deterministic or non-deterministic, but in the following it is assumed that they are deterministic.

ABSTRACT MACHINES

Following the notation used by Rhodes [RHO73] adapted to this problem a frame complex is considered as a black-box with inputs (selections) taken in C and outputs taken in the finite set consisting of objects in X.

Then by definition, the function:

$$m: C^* \rightarrow X$$

is called a machine

According to this definition, a selection process can be viewed as a machine which generates output in X from an input sequence in C^*.

The natural extension of m is defined by the function:

$$m^t: C^* \rightarrow X^*$$

such that:

$$m^t (c_1,\ldots,c_n)= (m(c_1), m(c_1,c_2),\ldots m(c_1,\ldots,c_n))$$

this will apply to a selection process which generates strings in X^* from input sequences in C^*.

The length of a string

$$c_1,\ldots,c_n \text{ is } L (c_1,\ldots,c_n)= n$$

if the string contains n symbols. A mapping $m:A^* \rightarrow B^*$ is length preserving if and only if:

$$L(m(a_1,\ldots,a_n))= L(a_1,\ldots,a_n)= n$$

the length of a string is the same after the mapping.

A causal mapping is a mapping where the output at time t depends only on the inputs up to time t. In the following all the mappings are supposed to be causal.

4.5.3 SEQUENTIAL SELECTION CIRCUIT OR SELECTON

The previous definition of a machine is quite general and it is now necessary to define an abstract object which relates more directly to the frame selection concept.

Besides the set C and X, we consider a set of frames, F.

We consider the frames as the states of an abstract machine called a selecton defined as follows:

Definition:

A selecton (also sequential selection circuit) is a 5-tuple

$S = (C, X, F, r, p)$

C is a non-empty set of codes or letters representing the choices selected (inputs to the selecton). X is a non-empty set of basic outputs (which may be also described by an alphabet.)

r is a mapping $r: F \times C \rightarrow F$ which is the action function associating with a given input and frame, another frame (this function is called the route of the selection process.)

p is a mapping $p: F \times C \rightarrow X$ called the present output function which associates with an input code and a frame f an output from X.

This definition is related to the previous definition of a machine by the following correspondence:

let $fo \in F$, then

$S_{fo}: C^* \rightarrow X$

is the machine corresponding to the selecton S starting with the frame f_o.

S_{fo} is defined by:

$S_{fo}(c_1) = p(f, c_1)$ for all $c_1 \in C$ and $f \in F$

$S_{fo}(c_1, \ldots, c_n) = S_{f1}(c_2, \ldots, c_n)$

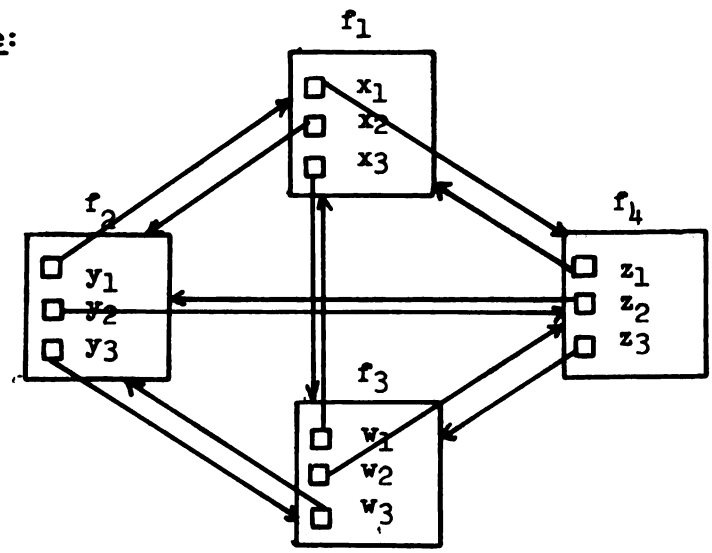
with $f^1 = r(f, c_1)$ for all $n \geqslant 2$ and $c_1, \ldots, c_n \in C^*$

A machine m of the form $S_{fo} = m$ is said to be realized by the selecton S starting with the frame fo.

A selecton can be represented by a diagram where the frames are represented with their selection points (squares), the route function is represented by the arrows going from one selection point to another frame, and the output can be represented by a string next to the selection point.

<u>Example</u>:

Figure 4.5



where wi, xi, yi, zi, ε. X.

This diagram can be called the frame transition diagram of the selecton.

The route and output functions are defined by:

$r(f_1,c_1)= f_4$	$p(f_1,c_1)= x_1$
$r(f_1,c_2)= f_2$	$p(f_1,c_2)= x_2$
$r(f_1,c_3)= f_3$	$p(f_1,c_3)= x_3$
$r(f_2,c_1)= f_1$	$p(f_2,c_1)= y_1$
$r(f_2,c_2)= f_4$	$p(f_2,c_2)= y_2$
$r(f_2,c_3)= f_3$	$p(f_2,c_3)= y_3$
$r(f_3,c_1)= f_1$	$p(f_3,c_1)= w_1$
$r(f_3,c_2)= f_4$	$p(f_3,c_2)= w_2$
$r(f_3,c_3)= f_2$	$p(f_3,c_3)= w_3$
$r(f_4,c_1)= f_1$	$p(f_4,c_1)= x_1$
$r(f_4,c_2)= f_2$	$p(f_4,c_2)= x_2$
$r(f_4,c_3)= f_3$	$p(f_4,c_3)= x_3$

A selecton will be called <u>reduced</u> if and only if starting S with different frames

$$f_1 \neq f_2, \text{ there exists } s = c_1, \ldots, c_n$$

such that $Sf_1(s) \neq Sf_2(s)$. The selecton is reduced if the same input applied to Sf_1 and Sf_2 gives different outputs.

An important result of automata theory is that there exists a <u>unique reduced circuit</u> [RHO73] for any finite state machine. Therefore, it is possible for a given selection process (machine) to be realized by a selecton (circuit) which uses a minimal number of frames and is unique.

For a proof of this result, see [RHO73] or [GIN68]. This shows that given a selection process or a selecton operating on a given semantic complex, there exists a unique frame structure which is minimal and which will produce the same outputs.

This is important because there is no such equivalent result for computer programs. (There is not a minimal program for a given algorithm for instance.)

Therefore, given a semantic complex, all the implementations of the complex by a selecton are equivalent to a reduced selecton and there exists algorithms to find the reduced representation [RHO73,GIN68].

Before proceeding, some new definitions are needed:

<u>Definition</u>:

If S is a selection string (or word) (f, s) in F x C* is called a <u>configuration</u> of the selecton S.

A <u>move</u> by S is a binary relation denoted as $\vdash$, and defined by the following:

if the present frame of S is f, and the choice selected on the frame is c, and if the next frame is f^1, then S makes a move from f to f^1 noted:

$$(f,cs) \vdash (f^1,s)$$

If there is a sequence of moves corresponding to a selection process, such that:

$$(f_o,c_1c_2\ldots c_ks) \vdash (f_1,c_2\ldots c_ks) \vdash (f_2,c_3\ldots c_ks) \ldots \vdash (f_k,s)$$

then:

$$(f_o,c_1c_2\ldots c_ks) \vdash^k (f_k,s)$$

If the number of moves is indeterminate, $(f,s) \vdash^* (f^1,s^1)$ means that there exists a finite number of moves which brings the selection from frame f to frame f^1.

A frame f^1 will be accessible from f_o if there is a selection string s such that $(f_o,s) \vdash^* (f^1,e)$.

If there is $f_o \varepsilon F_{ss}$ called the <u>initial frame</u>, and a set $\{f_e\} \varepsilon F_{ss}$ called the <u>final set of frames</u> such a selecton will be called a <u>start-stop selecton</u>.

A selection string is said to be generated by a start-stop selecton if:

$$(f_o,s) \vdash^* (f_e,e) \text{ for some } f_e \text{ in } \{f_e\}$$

The start-stop selecton is deterministic if r(f,c) gives no more than one frame for any f in F and c in C. It is said to be <u>completely specified</u> if r(f,c) gives exactly one frame.

Transposing the corresponding result obtained in automata theory [AHO72], it can be shown that the class of completely specified deterministic selecton has the same capability as the class of non-deterministic start-stop selecton. Therefore, in the following we can assume that all start-stop selectons are completely specified.

4.5.4 OPERATION ON SELECTONS

4.5.4.1 HOMOMORPHISM AND TRIVIAL CODES

Given two semigroup $(S_1,.)$ and $(S_2,*)$, $h:S_1 \rightarrow S_2$ is a homomorphism if and only if:

$$h(s_1.s_2) = h(s_1)*h(s_2) \quad s_1,s_2 \in S_1$$

If C_1 and C_2 are two sets, then $h:C_1^* \rightarrow C_2^*$ is a trivial code if h is a homomorphism; h is a trivial length preserving code iff h is a trivial code and h is length preserving.

Therefore, if two selectons are defined on different sets C_i,X_i, it is possible to compare their intrinsic structure or capability by using trivial code or trivial length preserving code by defining:

$$h_1:C_1^* \rightarrow C^* \qquad O_1:X_1^* \rightarrow X^*$$

$$h_2:C_2^* \rightarrow C^* \qquad O_2:X_2^* \rightarrow X^*$$

The homomorphism operation means that one selecton can be simulated by (or simulates) another homomorphic selecton.

An application of this is important if one considers the operation of translation of output strings from one natural language to another (English or Spanish, for example, will require two different alphabets). In this process of translation, the basic structure of the selectons will not have to be modified, thus the name "frame" is well justified because it really represents a frame of alternatives that can be filled by words or phrases of any language.

In the following, we suppose that C and X are identical for all selectons.

4.4.4.2 UNION OF SELECTONS

Given two start-stop selectons

$$S_A = (C, X, F_{ssA}, r_A, p_A)$$

$$S_B = (C, X, F_{ssB}, r_B, p_B)$$

S_{AUB} is defined as the union selecton which will have the same outputs as S_A and S_B given the same input string:

$$S_{AUB} = (C, X, F_{ssA} \times F_{ssB}, p_{AB})$$

- the new set of frames is the cartesian product of F_{ssA} and F_{ssB}.
- the route funtion is defined as follows:

$$r_{AB}[(f_A, f_B), c] = (f'_A, f'_B)$$

$$\text{iff} \; r_A(f_A, c) = f'_A$$

$$r_B(f_B, c) = f'_B$$

- the output function is:

$$p_{AB}[(f_A, f_B), c] = (x_A, x_B)$$

$$\text{iff} \; p_A[f_A, c] = x_A$$

$$p_B[f_B, c] = x_B$$

This new selecton is equivalent to S_A and S_B since, for each input c it outputs the pair of outputs that would have been produced by S_A and S_B separately. Furthermore, the next frame is the cartesian product of the two frames that would have been brought by the corresponding move in S_A and S_B.

More concretely, this can be applied to model frame selection systems which operate simultaneously on a different set of frames or applications.

We can also view this operation as the parallel composition of the two selectons S_A and S_B.

4.5.4.3 CONCATENATION OF SELECTONS

Given the two start-stop selectons S_A and S_B, a concatenation of S_A and S_B is a selecton which gives output strings which are the concatenation of the outputs given by S_A and S_B.

It is defined as follows:

$$S_{AB}=[C,X,F_{ssA}UF_{ssB},r,p]$$

- the frame set is the union of frame F_{ssA} and F_{ssB}
- the route and output functions are defined by:
- $r(f,c)= r_A(f,c)$ for all f in $F_{ssA}-\{f_e\}A$

  $p(f,c)= p_A(f,c)$ for all f in $F_{ssA}-\{f_e\}A$
- $r(f,c)= f_{oB}$ for all f in $\{f_e\}A$

  $p(f,c)=$ e for all f in $\{f_e\}A$
- $r(f,c)= r_B(f,c)$ for all f in F_{ssB}

  $p(f,c)= p_B(f,c)$ for all f in F_{ssB}

where $\{f_e\}A$ is the set of final frames of S_A and f_{oB} is the starting frame of B, e is the empty string.

S_{AB} can be viewed as the serialisation of the two selectons S_A and S_B where the mapping frame the final frames $\{f_e\}A$ is to the initial frame of S_B. Therefore, the "tail" of S_A is directly mapped into the "head" of S_B. It is easy to verify that such a selecton will produce outputs strings that are concatenation of outputs of S_A and S_B. This operation can also be viewed as the serial composition of the two selectons S_A and S_B. Such composition can be generalized to n selectons.

4.5.4.4 ITERATION SELECTON

Given a selecton S_A, an iteration selecton (also called star selecton) is defined by the following:

$$S_A^* = (C, X, F_{ssA} \cup \{F_e\}, r, p)$$

where Fe is a new frame such that:

$$r(Fe,c) = \{\emptyset\} \cup \{f_{oA}\}$$

$$p(Fe,c) = e$$

the route and output functions are given by:

$$r(f,c) = r_A(f,c) \text{ for all } f \text{ in } F_{ssA} - \{f_e\}A$$

$$p(f,c) = p_A(f,c) \text{ for all } f \text{ in } F_{ssA} - \{fe\}A$$

$$r(f,c) = Fe \text{ for } f \text{ in } \{fe\}A$$

$$p(f,c) = p_A \text{ for } f \text{ in } \{fe\}A$$

The frame Fe is created to enable the selecton to loop back to the initial frame f_{oA}.

The same result could be obtained by concatenation of S_A to itself as many times as required. The loop is introduced as was noticed previously to reuse the capability of the already existing selecton.

This operation is important from a practical point of view because it enables the indefinite use of loops in frame complexes by the introduction of procedural frames whose purpose is to give to the user the alternative to loop back to a previous node in the frame complex.

4.5.4.5 REGULAR SET AND EXPRESSIONS

The previous operations on selectons are in fact equivalent to the definition of regular sets and expressions.

If $A, B, \subseteq C^*$ are subsets of strings in C^*, then by defining:

$$A.B = \{a.b : a \varepsilon A \text{ and } b \varepsilon B\}$$

$$A^* = \{a_1 \ldots a_k : k \geq 1 \text{ and } a_j \varepsilon A\} \text{ or}$$

$$A^* = \bigcup_{n=1}^{\infty} A^n \text{ where } A^1 = A \text{ and } A^{n+1} = A^n.A$$

A^* is also called the transitive closure of A.

Then <u>regular sets</u> and expressions are recursively defined [AHO72] by: the regular sets over C and the regular expressions associated with those sets are:

i) $\emptyset$ (empty set) is a regular set and $\emptyset$ is a regular expression

ii) $\{e\}$ (identity) is a regular set and e is a regular expression

iii) $\{x\}$ is a regular set over C for all c in C and x is a regular expression

iv) if A and B are regular sets over C, then so are AUB, A.B and A^* and the corresponding regular expressions are:

(a+b), (ab), (a)*

v) nothing else is a regular set or expression

According to this definition, it follows that the operations defined previously on selectons (S_{AUB}, $S_{A.B}$, S_{A^*}) are precisely identical to the definition of regular sets or equivalently the set of selection strings accepted by a start-stop selecton is a regular set.

Therefore, a selection process will be called <u>regular</u> if it can be built by composition of selectons according to the rules previously defined.

The selection formula associated with a regular selection process can be represented by a regular expression.

4.5.4.6 CASCADE COMPOSITION OF SELECTONS

Given 2 selectons S_1 and S_2

$$Si = (C_i, X_i, F_i, r_i, p_i) \quad i = 1,2$$

the cascade composition of S_1 and S_2 is defined by:

$$\tilde{S} = (C, X1 \times X2, F_1 \times F_2, r, p)$$

- the route function is given by:

$$r[(f_2,f_1),c] = (r_2[f_2, M_2(c,x_1)], r_1[f_1, M_1(c)])$$

with M_1 and M_2 being mappings:

$$M_1: C \rightarrow C_1$$

$$M_2: CxX_1 \rightarrow C_2$$

x_1 can also be written as $p_1\ [f_1, M_1(c)]$

- the output function is defined by:

$$p[(f_2,f_1),c] = (p_2[f_2, M_2(c,x_1)], p_1[f_1, M_1(c)])$$

schematically $\hat{S}$ can be represented by the following diagram:

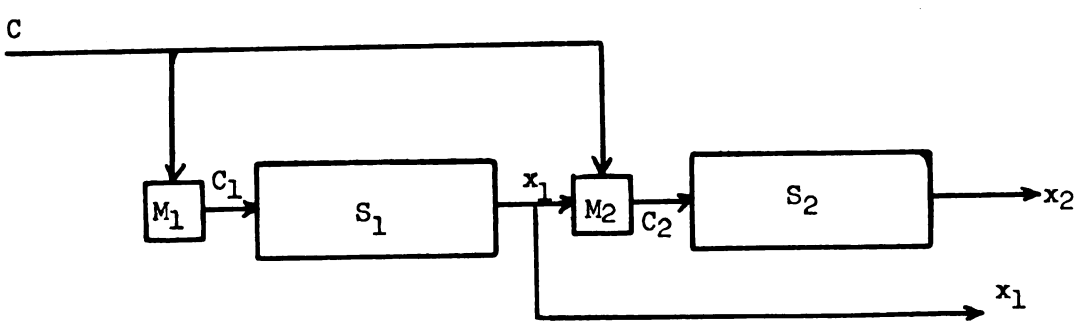


Figure 4.6 Cascade of Selectons

by generalization, one can define the cascade of selectons: S_i i= 1,n as $\tilde{S} = (C, X_1 x X_2 x \ldots x X_n, F_1 x F_2 x \ldots x F_n, r, p)$

with the mappings

$$M_1: C \rightarrow C_1$$
$$M_2: CxX_1 \rightarrow C_2$$
$$M_3: CxX_1xX_2 \rightarrow C_3$$
$$M_n: CxX_1xX_2 \ldots xX_{n-1} \rightarrow C_n$$

and

$r[(f_n,\ldots,f_1),c]=(r_n[f_n,M_n(c,x_1\ldots x_n)],r_{n-1}[f_{n-1},M_{n-1}]\ldots,r_1[f_1,M_1(c)])$

$p[(f_n,\ldots,f_1),c]=[p_n[f_n,M_n(c,x_1\ldots x_n)],\ldots p_1[f_1,M_1(c)]]$

It can be shown [ARB68] that a cascade composition is equivalent to any series-parallel composition.

The main advantage of such a composition is the "triangular" action introduced by the M_i:

- a selection c input can be modified by a mapping depending on the outputs (X_i) of the previous components of the cascade.

In particular, this can be used to specify semantic relations between component selectons in the cascade:

given the outputs sequence

$x_1,x_2,\ldots x_{p-1}$, the mapping M_p: $CxX_1x\ldots xX_{p-1} \rightarrow C_p$

can express the semantic relation existing between the X_i up to X_p and the next set of choices or alternatives that can be selected in C_p.

<u>Example</u>:

- Given a dose form and strength of a drug, one can restrict the choices of routes and frequency of administration to the only ones that are appropriate for the previous items selected.
- The same mechanism could be applied for potential drug allergies or drug-drug interactions.

The cascade composition of selectons is therefore more than a series-composition or a concatenation of selectons, it is a way to construct a selecton in which the parts are interdependent of each other and this can be used to introduce semantic constraints between the parts.

The cascade composition is also interesting from a mathematical point of view because it can be viewed as a product of components and

c

m

o

c

[

be

ca

be

ea

co

bi

Ex

conversely, one can find a decomposition of a given machine into component machines which "divides" the original one.

The theory of Krohn and Rhodes [RHO73] is a study of these properties of machines from an algebraic point of view and it shows that the cascade composition is equivalent to the "wreath product" defined on semigroups [RHO73].

The most important result of the theory is that a given machine can be decomposed into a product of "prime" machines (just as a given integer can be decomposed into a product of prime numbers).

The prime decomposition theorem [RHO73] states that any machine can be decomposed into a cascade of combinatorial machines and group machines.

In this application, a <u>group selecton</u> will be a selecton for which each selection permutes the frames of the selecton.

A <u>combinatorial selecton</u> will be one for which each selection is a constant map on the frames or the identity. It can be shown that combinatorial selecton can be built from flip-flop selectons.

<u>Examples</u>:

- a group selecton (or cyclic selecton)

$$S = (\{c_1,c_2,c_3\}, \begin{Bmatrix} x_1,x_2,x_3 \\ y_1,y_2,y_3 \\ z_1,z_2,z_3 \end{Bmatrix}, \{f_1,f_2,f_3\}, r, p)$$

with the following mappings:

$$\begin{array}{lll} r(f_i,c_1) = f_i & i\varepsilon\{1,2,3\} & p(f_1,c_i) = x_i \\ r(f_1,c_2) = f_2 & & p(f_2,c_i) = y_i \\ r(f_2,c_2) = f_3 & & p(f_3,c_i) = z_i \\ r(f_3,c_2) = f_1 & & \\ r(f_1,c_3) = f_3 & & \\ r(f_2,c_3) = f_1 & & \\ r(f_3,c_3) = f_2 & & \end{array}$$

it can be represented by the following diagram:

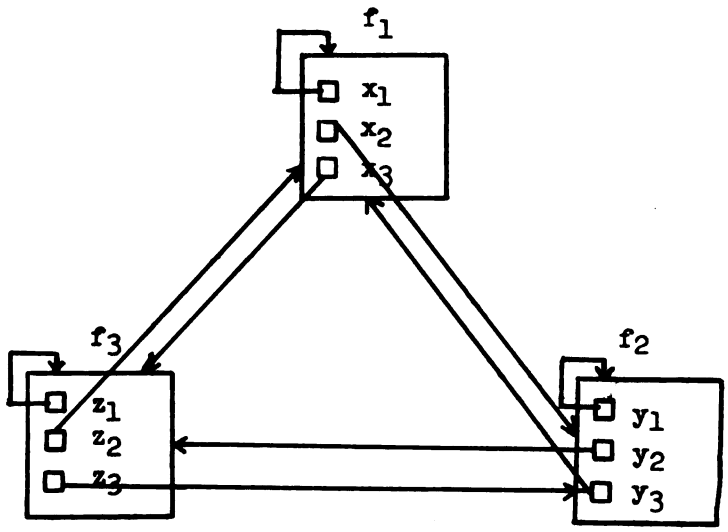


Figure 4.7 Group Selecton

A group selecton can be used to build repetitive messages or portions of phrases. An example of a combinational selecton is:

$$S = (\{c_1,c_2,c_3\},\begin{matrix}\{x_1,x_2,x_3\}\\ \{y_1,y_2,y_3\}\\ \{z_1,z_2,z_3\}\end{matrix}\{f_1,f_2,f_3,f\},r,p)$$

with $f(f_i,c_i) = f$, $p(f_1,c_i) = x_i$, $p(f_2,c_i) = y_i$, $p(f_3,c_i) = z_i$
$r(f,c_i) = f$, $p(f\ ,c_i) = e$ $\quad i\varepsilon\{1,2,3\}$

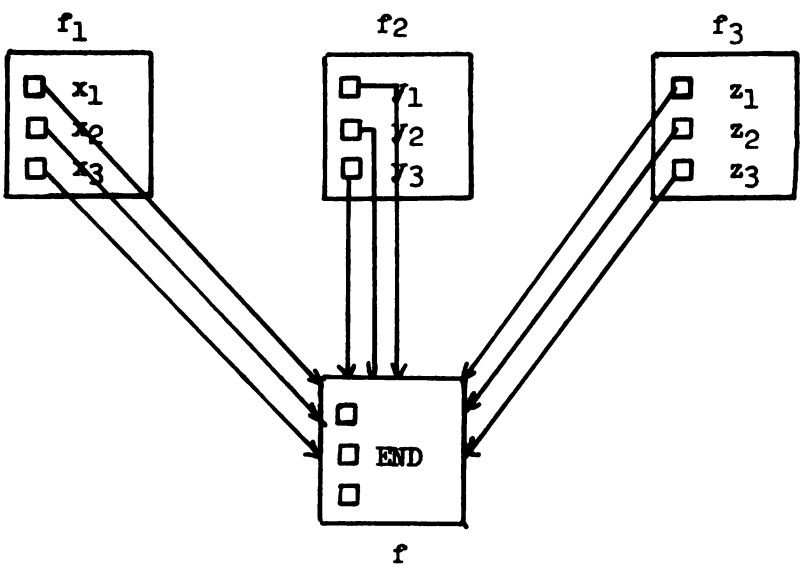


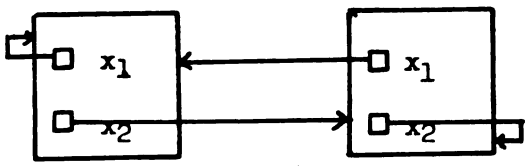
Figure 4.8 Combinatorial Selecton

The flip-flop selecton is:

$$S = (\{c_1,c_2\},\{x_1,x_2\},\{f_1,f_2\},r,p)$$

$$r(f_1,c_1) = f_1,\ p(f_1,c_1) = e \qquad r(f_2,c_1) = f_1,\ p(f_2,c_1) = x_1$$

$$r(f_1,c_2) = f_2,\ p(f_1,c_2) = x_2 \qquad r(f_2,c_2) = f_2,\ p(f_2,c_2) = e$$



f_1 Figure 4.9 f_2

Such a selecton can be used to confirm a binary selection by allowing it to modify the previous answer or to do a cross-examination (two operations that should lead to the same answer).

Remarks:

Although the previous results are not new to a mathematician, it is the author's belief that this is the first time that it has been recognized that a frame selection system, which is a software concept, can be implemented by a composition of abstract machines which directly model the actual system.

Furthermore, the use of regular expressions for the selection formulas is a very satisfying and condensed way to represent the selection process.

Finally, it is important to point out that the approach taken by defining the selecton model is somewhat the converse approach of the automaton model because the purpose of a selecton is not to recognize correct strings, but to restrict the choices or alternatives to only those which are syntactically and semantically correct and makes sense (i.e., regular strings).

The selecton model can be viewed as a model for software machines; it runs according to "soft wires" (route of the selection process) and "soft states" (semantic simplex represented by the frame.) Furthermore, selectons are real-time machines which can interact with a human operator. Such a machine is not driven by clock cycles in a synchronous manner, but asynchronously by selections of alternatives which can be done in real-time.

It is a characteristic of medical information systems, that they should be able to respond in real-time, in most instances. Frame selection systems are software machines which are precisely designed to meet this requirement in the most natural way.

The next section is a study of the power of the languages generated by selectons.

4.6 FORMAL LANGUAGES GENERATED BY SELECTONS

Definition:

Let A be a set of letters called the alphabet, then L is a language with alphabet A if and only if $L \subseteq A^*$.

According to this definition, any set of strings over A^* is a language. Another way of defining a language is through a set of generative rules (also called productions) which tells how to operate on symbols to obtain a string which belongs to the language.

Such a system is called a grammar, and the symbols on which the productions or rules operate are composed of two sets: the terminal symbols which are the alphabet A of the language, and the non-terminals symbols which are syntactic categories. For example, a simple sentence of English could be generated by the production $S \leftarrow NP.VP$ where NP is a noun phrase and VP a verb phrase. NP and VP are examples of nonterminal symbols.

More Formally:

Definition:

A grammar is a 4-tuple

$G = (N, A, P, No)$

where:

- N is a finite set of non-terminal symbols
- A is the alphabet of the language or the set of terminal symbols
- P is a set of productions (p,q) also denoted by $p \rightarrow q$

  where p has the form $\alpha n \beta$ with $\alpha, \beta \varepsilon (N \cup A)^*$, $n \varepsilon N$ and $q \varepsilon (N \cup A)^*$

  p is the left member of the production and must contain at least one non-terminal symbol.

  q is the right member of the production and may or may not contain non-terminal symbols.

- No is a distinguished non-terminal symbol called the start symbol.
- <u>No is a sentential form</u> and if $\alpha p \beta$ is a sentential form, and $p \rightarrow q$ a production then $\alpha q \beta$ is a sentencial form.
- A <u>sentence</u> is a sentencial form without non-terminal symbol.
- <u>The language L (G)</u> is the set of sentences generated by G.

<u>Theorem</u>:

The grammar G= (F,X,P,fo) generates the same language as the selecton S_{fo}= (C,X,F,r,p) fo εF

where $r(f_i,c_k)= f_j$ and $p(f_i,c_k)= x_{ik}$

$f_i, f_j \varepsilon F, \quad c_k \varepsilon C, \quad x_{ik} \varepsilon X$

corresponds to the productions of G defined by:

$f_i \rightarrow x_{ik} f_j$

$f_e \rightarrow e \qquad$ if $f_e \varepsilon \{f_e\}$

By definition, the grammar will reproduce the same strings that the selecton does, each production corresponding to the move of the selecton. Starting with

$f_o \rightarrow x_{ok}\ f_1 \qquad k= 1,\ldots,Max$

$f_1 \rightarrow x_{1m}\ f_2 \qquad m= 1,\ldots,Max$

The strings generated are $X_{ok} X_{1m} f_2$ with $\{^{k}_{m}$= 1...Max

- <u>if</u> $f_2\ \varepsilon\{f_e\}$ then all the strings generated by G are the same, than those generated by S_{fo}
- <u>if</u> $f_2\ \varepsilon\{f_e\}$ the prefix $x_{ok}\ x_{1m}$ is identical to the selection sentence already generated by the selecton composed of frames {fo,fl}

Since the grammar is built according to the moves and output of the selecton, the sentential form of G will be always of the form:

$x_{ok}\ x_{1m} \ldots x_{nr}\ f_{n+1}$

and the prefixes generated $x_{ok}...x_n r$ will be identical to the output selection sentence generated by the selecton composed of the frame $\{f_o, f_1, ..., f_n\}$

Therefore, if the prefix string has length $\ell(s) = n$ either $f_n \varepsilon \{f_e\}$ and all the strings generated by G are identical to those generated by S_{fo} or $f_n \rightarrow x_{nt} f_{n+1}$ and then all the prefix string of length n + 1 generated by G are identical to the output selection sentences generated by S_{fo}.

This shows that a selecton has the same power as a grammar which has productions characterized by:

- one non-terminal symbol in the left-hand side of the production
- at least one terminal symbol in the right-hand side of the production

Such a grammar is known as belonging to the class 3 in the Chomsky hierarchy [HOP69], and is called a right linear grammar. This result is important because it shows that a regular frame selection system implemented as a combination of regular selectons will only be capable of dealing with problems which can be stated in terms of right linear grammars.

It must also be pointed out that menu tree systems have less capabilities than the regular frame selection system because they do not have the loop capability. Therefore, menu tree systems generate languages that are even less powerful than those generated by a right linear grammar.

4.7 PUSHDOWN SELECTON AND CONTEXT-FREE LANGUAGE

The next question is how the capability of selecton languages may be enhanced without losing the basic concepts of frame selection systems. The most natural improvement is to try to "jump" to the next class of language in the Chomsky hierarchy; i.e., class 2, or context free languages. In the following, it is shown that by introducing a stack associated with a regular selecton, such a jump from regular language to context-free language can be made.

Definition:

A pushdown selecton is a six-tuple

$S_z = (C,X,F,Z,r,p)$

where C, X, F are the respectively the input, output, and frames sets

Z in the finite set of pushdown symbols

r is the action function that maps $C \times F \times Z^*$ into $F \times Z^*$

p is the output function $C \times F \rightarrow X$

The essential difference between a regular selecton and a pushdown selecton is the introduction of a pushdown stack and a pushdown alphabet.

The action function or next state function operates on the stack either by pushing objects on the stack or by removing objects from the stack as a result of a selection.

Applications:

The introduction of a stack is very important in a frame structure because it enables the use of conditional branching in the structure and in fact allows conditional statement such that

If selection i was selected, then the route will be route j.

This can be done by using the stack for temporary storage of strings or conditional variables when selection i is made and by checking later if the condition is met.

More precisely, the pushdown stack can be used to do conditional branching on the frames: supposing that a common regular selecton S_c is needed for several different selection processes such that the paths before and after the use of the common part of the selecton are different as represented by the following diagram:

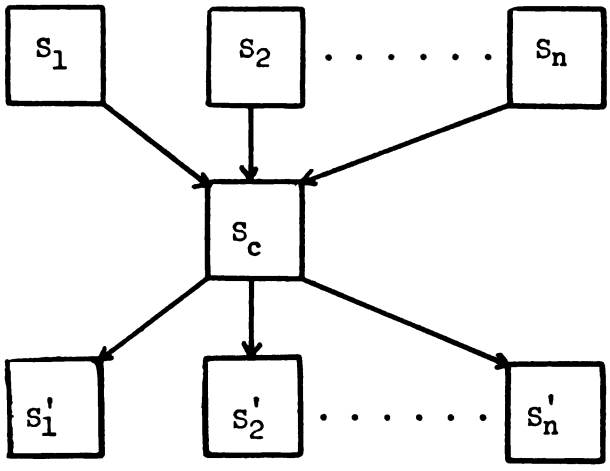


Figure 4.10

Such a case can be implemented by pushdown selectons $S_1,\ldots,S_n$, which will push respectively $S_1',\ldots,S_n'$ on the stack and at the end of S_c the debranching will be done by "popping" the path (or selecton) to be taken after the execution of S_c. Thus more complicated structures can be implemented with the aid of the pushdown stack.

It can be shown that the pushdown selecton generates the class of languages known as context-free languages. Before proceeding, the definitions of context-free languages and normal form for a context-free language are necessary.

Definiton:

A grammar $G= (N,A,P,N_o)$ is said to be context-free if the production rules have the following form:

$$n \rightarrow \alpha \quad \text{where } n \in N \text{ and } \alpha \in (NUA)^*$$

In other words, the left side contains one non-terminal symbol and the right side of the production may or may not contain a single terminal symbol. It can be proved that every context-free language can be reduced to a normal form called Chomsky normal from (CNF) where all the productions are in one of the two forms [AHO73]:

$$P \rightarrow QR \qquad P,Q,R \in N$$

$$P \rightarrow a \qquad P \in N, a \in A$$

Theorem:

The pushdown selecton

$$PS_{fo} = (C,X,F,Z,r,p)$$

with the following r and p functions:

push action $\quad r\ (c_1,f_i,z) = (f_j,f_k z)$

$\qquad\qquad\qquad p\ (c_1,f_i) = e$

or

pop action $\quad r\ (c_2,f_i,f_j z) = (f_j,z)$

$\qquad\qquad\qquad p\ (c_2,f_i) = x_i$

generates the same language as the context free grammar $G= (F,X,P,fo)$ with the following production types:

$$f_i \rightarrow f_j\ f_k$$

$$f_i \rightarrow x_i$$

Since every context-free language can be represented in Chomsky normal form:

$$f_i \rightarrow f_j\ f_k$$

$$f_i \rightarrow x_i$$

are normal form productions and therefore generates a context-free language.

Conversely, if $f_i \rightarrow f_j\ f_k$ is a production of G, it can be implemented by a pushdown selecton which "pushes" f_k and goes to f_j. If the production is of the type $f_i \rightarrow x_i$, then we use the second action function which "pops" the stack and outputs x_i.

The following is an informal proof of the theorem stated before:

i) assuming that the grammar has one production

$f_o \rightarrow x$

$L\ (G) = x$

it is equivalent to the selecton $PS_{fo} = (c,x,f_o,z,r,p)$

with:

$r\ (c,fo,e) = (e,e)$

$p\ (c,fo) = x$

$L\ (G) = L\ (PS_{fo}) = x$

ii) If the grammar is in normal form, the next CNF grammar must have three productions because the addition of a new production of the type $f_i \rightarrow x$, cannot be useful without a reference in another production, therefore, it must be of the form:

$f_o \rightarrow f_1\ f_2$

$f_1 \rightarrow x_1$ $\qquad L\ (G) = x_1\ x_2$

$f_2 \rightarrow x_2$

In the following e represents the empty frame or string:

$r\ (c_1,f_o,e) = (f_1,f_2)$

$p\ (c_1,f_o) = e$

$r\ (c_2,f_1,f_2) = (f_2,e)$

$p\ (c_2,f_1) = x_1$

$r\ (c_2,f_2,e) = (e,e)$

$p\ (c_2,f_2) = x_2$

it follows that:

$L(PS_{fo}) = x_1 \; x_2$

and

$L(PS_{fo}) = L(G)$

iii) Reasoning by induction:

supposing $L_n(G) = L_n(PS_{fo})$ when the grammar has n useful productions. Then to obtain a grammar with more useful productions a production of the type $f_i \rightarrow x_i$ must be replaced by three productions of the type:

$f_i \rightarrow f_j \; f_k$

$f_j \rightarrow x_j$

$f_k \rightarrow x_k$

This is equivalent to say that $L_{n+2}(G)$ is obtained from $L_n(G)$ by replacing every x_i in the previous language by $x_j \; x_k$ in the new language. Therefore, since $L_n(PS_{fo}) = L_n(G)$, it is true that $L_{n+2}(PS_{fo}) = L_{n+2}(G)$. This proves the induction step and therefore every pushdown selecton with any number of frames is equivalent to a context-free grammar.

Discussion of results:

This theorem is also very important because it shows that frame selection systems with the addition of a stack operation have the same capabilities as the class of context-free language.

In regard to the fact that usual computer programming languages are basically context-free (ALGOL-like languages in particular) this shows that frame selection systems are theoretically as powerful as any conventional programming language.

In practice, frame selection systems may not be convenient for genera-

ting computational formulas (although it is possible), but they are more adapted to deal with generation of sentences using English-like phrases using a context-free grammar.

This is what was meant by approximation of a natural grammar by a formal grammar. Such a grammar might be more restrictive but it is easier to handle by computer.

Furthermore, the frame selection approach does not require that the user learn the underlying grammar as is the case for the usual programming languages. The grammar is encoded in the structure of the frame complex, therefore, preventing the user from committing syntactical or lexical mistakes. The only thing that the user is required to do is to select the items of the semantic simplexes represented on the frames.

The argument of inflexibility is not really valid because with this software approach it is possible to improve and adapt each application to the particular needs of the user.

In many respects, it is in fact more flexible than regular time-sharing systems which will refuse an answer if it is not in the proper form (misspellings, punctuation, codes, etc.) Instead, with this approach the possible answers are listed on a frame and need only to be selected.

Frame selection systems are also more extensible because the introduction of new frames and selections can be done without re-programming if one has available a frame programming language (see next chapter.)

Finally, the question of possible ambiguity of the sentences generated is irrelevant because the messages can always be accompanied by the selection strings corresponding to the message generated.

4.8 COMPUTATION AND FRAME SELECTION SYSTEMS

It has been shown that a frame selection can be used to generate regular or context-free languages designated as classes three and two in the Chomsky hierarchy.

The classes one and zero are known as context-sensitive and recursively enumerable sets and are associated with Türing machines.

A Türing machine is characterized by a tape and a finite control

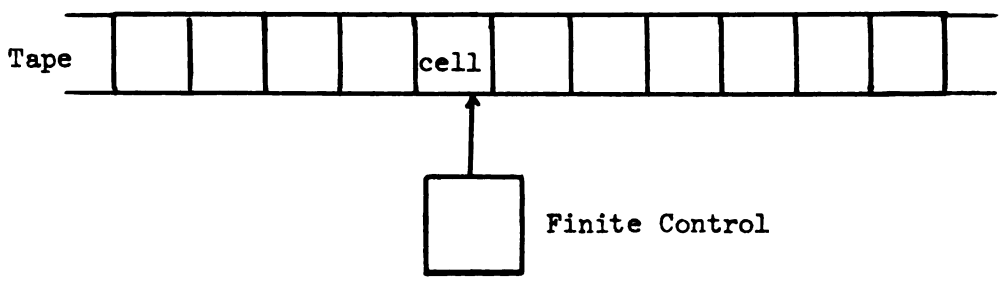


Figure 4.11 Türing Machine

The tape is of indefinite length, and is divided into cells which are either blank (0) or not blank (1), and the tape can move either to the right or the left. The tape can run through the reading head of the finite control so that it can examine one cell at a time.

The operations that can be performed by the machine are:

0 - it can erase the cell (makes it blank)

1 - it can write on the cell (non-blank)

2 - it can move to the cell on the right

3 - it can move to the cell on the left

If I= {0,1} represents the condition of the tape, and X= {0,1,2,3} represents the operations as defined above.

If Q is a set of states of the finite control, then the machine is a mapping:

$$Q \times I \rightarrow Q \times X$$

and the behavior of the machine is determined by the quadruple

$$q_i \; I_k \; X_e \; q_j$$

- q_i: the initial state
- I_k is the input taken in I (i.e., the condition of the tape cell: blank or not)
- X_e is the output taken in X (i.e., the operations defined above)
- q_j is the state after the operation is performed.

Despite its apparent simplicity, such a device is theoretically interesting because it can carry out any computation. A sequence of quadruples $q_i \; I_k \; X_e \; q_j$ is called a program, and a program computable by a Türing machine is also called a <u>procedure</u>.

The new element in the Türing machine model is the fact that the tape can be moved left or right with respect to the reading head. Applied to the selecton model, this would mean that instead of a pushdown store, we would have to consider two pushdown stores. This is true because there is a theorem which states [HOP69] that "an arbitrary single tape Türing machine can be simulated by a deterministic two-pushdown tape machine." A deterministic two pushdown tape machine is a deterministic Türing machine with a read-only input and two storage tapes. If either tape moves right with respect to the head a blank is printed on that tape. This shows that if a regular selecton is associated with two pushdown stores, it becomes a two pushdown selecton which has the same capability as a Türing machine.

The corresponding theory will not be developed here, but some remarks will be made:

- First, it shows that selecton models are capable of generating any language from class 3 to class 0, in the Chomsky hierarchy.
- Second, it shows that with the adjunction of a second pushdown store, the model can generate context-sensitive language. Indeed this can be practically done by allowing after each generation of a phrase, to look back at the previous words or phrases (the context) to modify the present output in the correct context.
- Third, the second pushdown store can be used in order to "reverse" the selection process and enables the erasure of the phrases already emitted up to a certain point (the phrase already emitted can be considered as the second tape.)
- The introduction of procedures can help in generating more powerful languages with context-sensitive features as shown below.

APPLICATION TO TRANSFORMATIONAL GRAMMARS:

The linguistic theory developed by Chomsky [CHO65] for the syntax of natural language is based on a model which allows the basic phrase structure of a sentence to be modified by transformation rules applied to the terminal strings obtained by a phrase structure grammar(usually a context-free grammar.)

An example of such a rule is given by the passive versus the active form:

NP1-V-NP2 → NP2 - be + en -V- by NP1

For example, "The physician examines the patient" can be transformed to "the patient is examined by the physician." It must be emphasized that such a transformation can be done automatically by a procedure.

There is no information created by going from the active to the passive form, it is just a computation on the elements already existing.

Returning to the selecton model such transformation rules can be implemented as frames representing the alternatives transformations that may be applied to a given phrase structure. The example presented before could be implemented by the selection formula:

$$\square\ NP_1\ \square\ V\ \square\ NP_2\ \square\ AP$$

where AP will be a frame to select the alternative active or passive. This suggests that a given selection might be associated with a procedure in order to carry out a computation on the string already generated.

It would be too constraining and unrealistic to carry out these computations by a Türing machine (here the computation is stable and the only freedom of choice is given by the data representing the string already produced and the program of the Türing machine is predetermined by the transformation to accomplish.) Therefore, instead of defining a selecton which will have the same capability as a Türing machine, the same result can be obtained by a set of procedures T (transformation or computations) which can be executed as a result of a selection.

A frame selection system is defined as follows:

<u>Definition</u>:

A frame selection system is the eight-tuple:

$$FSS = (C,X,F,Z,T,r,p,m)$$

where C,X,F,Z are the input, outputs, frames and pushdown sets.

- T is a set of computational procedures or transformations operating on the string already produced by the underlying selecton.
- m is a mapping from $C \times F$ to T such that $m(c,f) = t, t \in T$, t can be the empty or null procedure also denoted "NOP."

Such a device will work like a pushdown selecton for producing strings but occasionally a selection will trigger a procedure which will be executed as a result of the selection.

Thus, the device called a frame selection system is basically composed of two parts: a pushdown selecton or a regular selecton and a set of procedures which can be executed as a result of a selection. Such a device has the same capability as any computing device because one can consider the sequence of procedures $t= t_1\ t_2 \ldots t_n$ corresponding to the selection string $s_1 \ldots s_n$. In this case, t is a concatenation of procedures; i.e., a collection of procedures which has the effect of executing $t_1, t_2, \ldots t_n$ successively. The procedures $t_1 \ldots t_n$ can be considered as a program which is a sequence of subroutines call such as:

```
CALL   SUB1
CALL   SUB2
 .
 .
 .
CALL   SUBn
```

Each of these subroutines can be written in any standard computer programming languages.

This corresponds to standard programming techniques and it should pointed out that this approach is in fact similar to the structured programming approach: [DAH72]

Each frame can be associated with a set of procedures which can be viewed as the alternative instructions of a CASE statement [WIR66] Similarly, a two choices alternative will correspond to an IF statement. For instance, if each choice is associated with a procedure (possibly NOP), the frame would be equivalent to the following statement:

```
CASE OF CHOICE
        CHOICE 1:  CALL SUB 1
        CHOICE 2:  CALL SUB 2
           .
           .
        CHOICE n:  CALL SUB n
```

A frame selection system can be viewed as a sequence of CASE statements and each particular selection string corresponds to a string of procedures in the sequence of CASE statements. It is clear that if a problem is decomposed and analyzed in order to be implemented as a frame selection system, each large task will be decomposed into smaller tasks in a top-down fashion and each alternative would have to be clearly specified on frames. The end result is a sequence of frames associated with procedures that can be easily implemented, modified and maintained. The frame structure becomes a super structure that "holds the parts together" and furthermore, this superstructure is understandable by non-programmers.

Therefore, the frame structure becomes the common language between the user and the designer-programmer and this is very important in an environment where the applications must be user-oriented.

In conclusion, the next figure summarizes the different aspects of frame selection systems in relation to the theory of automata and the theory of formal languages. It can be seen that the frame selection model encompasses some capabilities that are particular to each theoretical model with the addition of a pragmatic capability which is absent in the other models.

	Theory of Automata	Theory of formal grammars	Selecton and frame selection system
Application	Modeling of hardware and string recognizers	Study of language syntax	Real-time interactive systems
Structure	States + next state function	Production rules	Frames + route functions
Freedom of choice in the inputs	Inputs from alphabet	Selection of productions	Selection inputs
Lexical capability	Recognition of words or tokens	Generation of words or terminal symbols	Generation of choices item or phrases
Syntactic capability	Recognition of sentences or string	Generation of sentences and transformations	Generation of sentences + execution of procedures
Output	Accept or reject input	Syntactically correct sentences	Syntactically correct sentences
Semantic capability	Execution of procedures to recognize semantically correct sentences	No	Semantic restriction and execution of procedures to produce semantically correct sentences
Pragmatic	No	No	User interaction and selection of alternatives

Figure 4.12

Chapter 5

A Frame Programming Language (FPL)

"It seems to me that all my creation is an effort to weave a web of connection with the world; I am always weaving it because it was once broken. But as I want these webs to be always truthful, I do not know how to break the false ones..."

Anais Nin

Diary Vol. 3

5. FRAME PROGRAMMING LANGUAGES

In order to build application systems utilizing the frame selection process, a frame programming language (FPL) was found necessary in order to conveniently define the frame structure and the actions to be taken following each selection from a frame. In order to illustrate the distinctiveness of FPL, this Chapter contains a brief introduction to the frame selection language SETRAN-HIP, used in the development of the medical information system of the University of Vermont [WEE69] and the language FOPS employed in the frame selection system built by the Sanders Corporation for the Kaiser Hospitals [VAN70,SIN70].

5.1 PREVIOUS FRAME PROGRAMMING LANGUAGES

5.1.1 SETRAN-HIP

SETRAN (Selectable Element TRANslator) was designed by Control Data Corporation in order to develop a frame system using a touchscreen device as the selection mechanism. HIP (Human Interface Program) is the frame selection application as it appears to the user.

SETRAN is described as a "keyboard program by which pages are built into branching displays...; SETRAN pages differ from HIP displays by the presence of code letters and numbers within which lie the directions for the branching displays" [WEE69].

SETRAN gives the user the capability to read, write, erase pages (frames) on the mass storage device. The implementation distinguishes different type of pages:

- T pages which do not have a paragraph segment (message area.) They are used to initiate and terminate paragraphs.
- I pages (intermediate) have a paragraph segment and have branching capabilities.

- G pages which are used for descriptive information where the branching capabilities of I displays are not needed.

A typical command would be: RI1363 which means: read I page number 1363 where 1363 is an octal number.

It must be noted that in SETRAN every page (frame) is referenced by an octal number. This feature implies that for a medium or large system (above 100 frames), a highly organized documentation scheme must exist to keep track of the page ID number.

Similarly, the branching system is entirely based on an octal code system. Each displayed selectable element is associated with a fixed format field which contains:

- page ID numbers (absolute or relative to the current page.) They indicate the next page to be brought (lower order display) or the page starting a loop (high order display) which will be put on a stack.
- letter codes are used to indicate flags for message inclusion of the selectable element or multiple choice frames.

5.1.2 FOPS

FOPS (File Oriented Programming System) was developed by Sanders Co. [SIN70] for the Kaiser Foundation Hospitals.

FOPS is a statement oriented interpretative language including commands, operators, modifiers, and system variables. FOPS has direct statements executed immediately and indirect statements to be executed at a later time. A program is a sequence of indirect statements which execute a specified task.

Frames can be build by using an operator called FORM. The FORM operator enables the user to define page displays which contains elements

selectable by a light pen. A FORM can be divided into PARTS which are fields to be filled in and when a form is constructed the user may designate these parts to be selectable.

Rather than present a formal description of FOPS, an example is given below which illustrates its capabilities:

```
DATE:                                DR.
PATIENT NAME                         ROOM

DRUG:
DOSE:                ROUTE:

   PHENERGAN                         25 MG
   PHENOBARBITAL                     50 MG

                      ORAL
                       IV
```

Example of a form in FOPS

This form could be used to build drug orders. A drug order will be built by selecting a given drug name which will be moved in the part DRUG:, a dose to be moved in the part DOSE:, and similarly a route in the part ROUTE:.

This type of selection process is static because all the elements to build the drug order must be on the same display and no procedure can be executed after a selection.

This reduces the message building capabilities, and it does not allow for semantic restrictions that can be applied as a result of previous selection (in this example, the two drugs must have the same possible dose forms as well as the same possible route of administration.)

5.2 A FRAME PROGRAMMING LANGUAGE (FPL)

In Chapter 4, it was shown that a frame selection system can generate regular or context-free languages. The approach taken in the following is that this capability should be used to design the frame programming language itself.

Instead of defining a grammar abstractly and then build a compiler which will recognize the correct sentences, a frame selection system can be constructed as an interactive system which will generate only syntactically correct sentences of the abstract grammar. It is, in fact, the converse approach of classical programming languages: instead of having a syntax-directed compiling [DON72], it can be viewed as a syntax-directed programming approach.

A classical compiler can be represented schematically as follows:

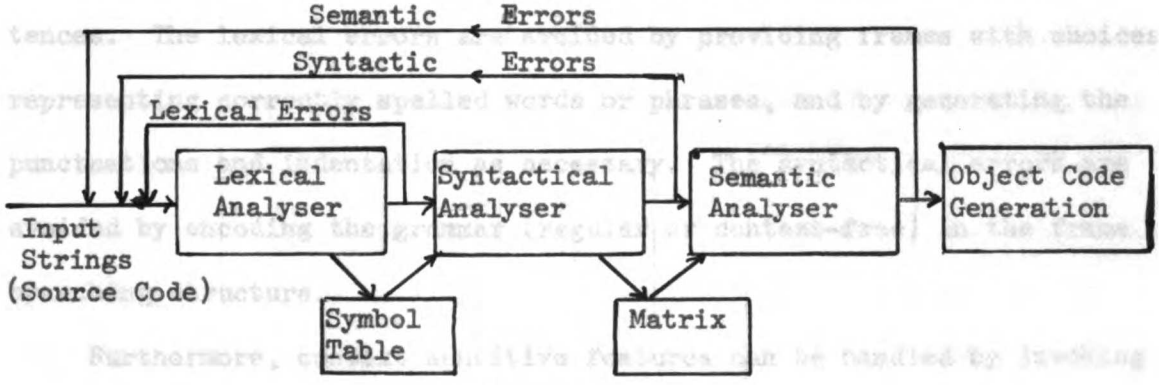


Figure 5.1

The input strings are submitted to several successive types of analyses:

- the lexical analyser detects lexical errors such that mis-spellings, punctuation errors and determine the tokens to build a symbol table.
- the syntactical analyser (parser) detects syntactic errors according to the grammar of the language and generate a matrix of syntactic units recognized.
- the semantic analyser detects the semantic errors and provides

interpretations to statements which are syntactically correct but may be interpreted differently depending on the semantics of the language. In this scheme, some other possible steps of a compiler system, such as the optimization and storage assignment, are ignored for the sake of simplicity. The point is to notice the existence of several feedback loops which take place in case of errors.

All these types of errors generate several corrections and compilation passes until the program is error-free (as far as the compiler is concerned.) The major difficulty in compiler design is that they must detect all these errors before running time.

The approach taken in FPL is to restrict the freedom of choice of the user by preventing him from generating syntactically-incorrect sentences. The lexical errors are avoided by providing frames with choices representing correctly spelled words or phrases, and by generating the punctuations and indentation as necessary. The syntactical errors are avoided by encoding the grammar (regular or context-free) in the frame branching structure.

Furthermore, context sensitive features can be handled by invoking computational procedures associated with a given selection, or by restricting the choices on future frames to be referenced. The following schema represents the FPL approach:

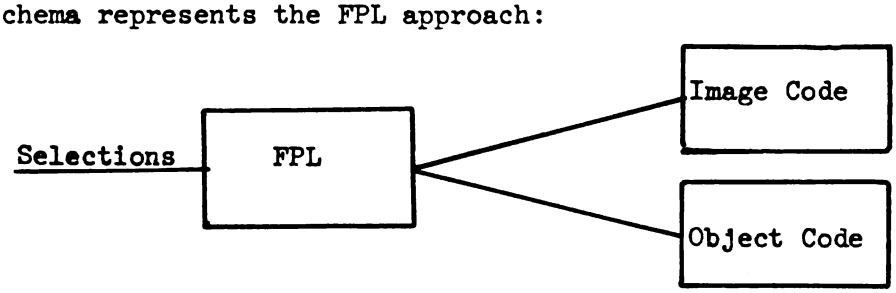


Figure 5.2

The image code represents the strings and sentences produced by FPL as a result of the selection and execution of semantic routines applied to the output strings. The selection strings (i.e., the successive selections made to generate an output string) represent the parsing of the image code from which computational routines can generate the object code.

It must be noted here that the problem of ambiguity and avoidance of key-words becomes irrelevant because the selection strings associated with the image code indicate which particular path of the parse tree was meant by the user.

The precedence relations can naturally be handled by the structure of the frame sequences. Although it is possible to encode any programming language grammar in such a way, the approach is more appropriate for developing interactive problem-oriented language where the users have little knowledge of algorithmic languages and want to develop their applications in terms familiar to them.

Due to the extensible nature of a frame programming language, it is possible to start the language with a minimal number of features and then extend the capability of the language by adding new frames. In effect, a frame programming language can be bootstrapped from a minimal syntax which corresponds to a regular grammar.

A minimal frame language is characterized by the possibility of defining frame displays, writing, reading and erasing frames from a mass-storage device. It must also allow the possibility of linking frames and output strings or execute procedures as a result of a selection.

5.2.1 BUILDING AND FILING OF FRAMES

This aspect of the language corresponds to the definition of the declarations of classical programming languages. It involves the definition of the frames' content and is followed by the filing of frames on a secondary storage.

IMPLEMENTATION OF THE FRAME CONCEPT: SELECTION FRAMES, EDITING FRAMES AND MIXED FRAMES

In this implementation, a CRT screen is composed of 24 lines of 48 characters (1152 characters).

Three types of frames are distinguished: editing frames, selection frames and mixed frames. An Editing Frame (Fig. 5.3) contains message prompts and empty fields to be entered from the keyboard by typing. A Selection Frame (Fig. 5.4), contains a maximum of thirty selectable items arranged in a grid of 10 lines by 3 columns. A third type of frame is called a Mixed Frame (Fig. 5.5) because it is a selection frame which allows the definition of a selectable item by entering it, on a keyboard.

Examples of such frames are given below:

```
Name←(Last.First.Init):
Hosp #:          Bed:        Birthdate:  -  -
Sex:  Race:  Wt:   Kg Adm:  -  -   PhCh:$
MD:                          Last Surg:  -
Adm Prob:
DX:


Diet:        Low Fluid:     I/O:
Spec Orders:
```

Figure 5.3 Editing Frame

ACETAMINOPHEN	DIFHENHYDN Na	PENTAZOCN Cl
ALLOPURINOL	DOSS	PENTAZOCN Lac
AMPICIL TRIHD	Fe SO4 USP	PENTOBARB Na
ASPIRIN	FLURAZEPAM Cl	PHENOBARBITAL
BENADRYL	LOMOTIL	POT Cl
BISACODYL	MAALOX	PREDNISONE
CHLORAL HYD	METHYLDOPA	PROCLOPER ED
DEXAMETHASONE	MOM	PROPOXPHEN Cl
DIAZEPAM	MULTIVITS UC	RIOPAN
DIGOXIN	PEN PHENOX K	SECOBARB NA

Figure 5.4 Selection Frame

FREQUENCY OF ADMINISTRATION

qd	q1h	ac
bid	q2h	pc
tid	q3h	with meals
qid	q4h	hs
STAT	q6h	prn
	q8h	
	q12h	:

Figure 5.5 Mixed Frame

Both selection and mixed frames do not use the four upper lines of the screen which are reserved for a message window for the sentences to be build during a selection process. The building of frames starts with the presentation to the user of an index (FPL index) which indicates the

alternatives available. The type of frame is specified by selecting one of the alternatives on the FPL index and this brings either a frame skeleton representing the grid which can be filled in by selectable items (selection and mixed frames) or a full screen available for prompts and fields (editing frame.)

When the display is filled with the text to be shown on a frame, a function key will trigger the storage of the frame in a temporary storage area. This operation will bring back the FPL index and then a name can be given to the frame by selecting the laternative "identification of frame;" a mixed frame will be displayed in order to type the name ot the frame built previously. Then by selecting the choice "WRITE," the frame will be stored permanently on the secondary storage medium and the frame directory will be updated. If the name given was already used for another frame a warning message will be given to the user and he will reenter a new name.

The frame used to enter the name can also be used to read, rewrite or delete a named frame by executing the corresponding procedures as a result of a selection.

The following figure represents the different frames and selections used in this phase of a frame construction:

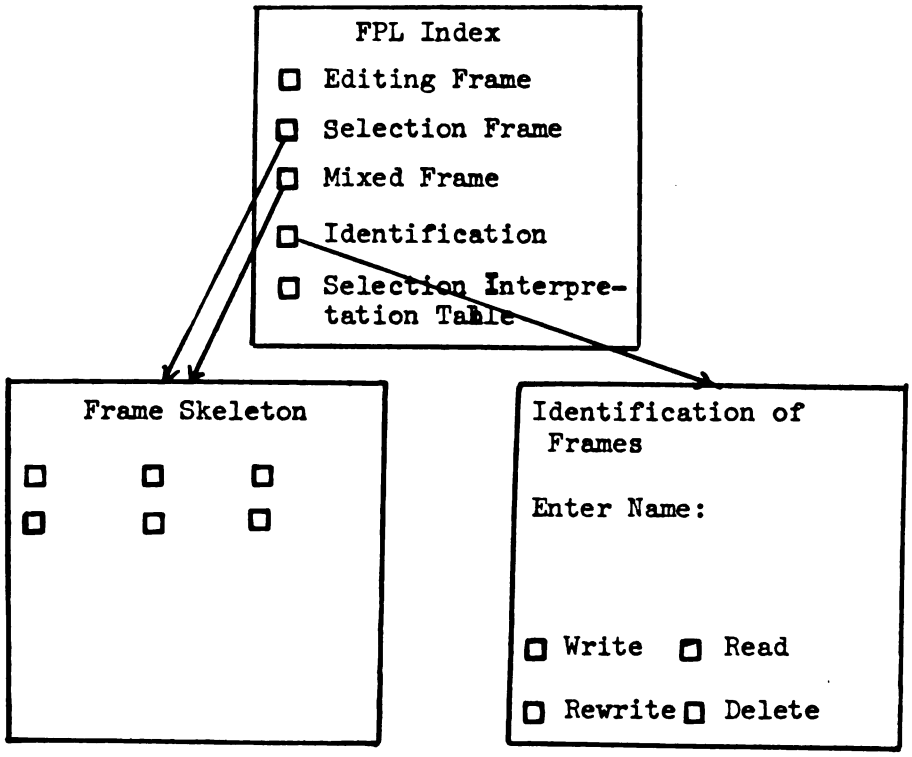


Figure 5.6 Creation of a Frame

Once a set of frames corresponding to an application or a subset of an application has been defined, one can proceed to the definition of the executable selection instructions to be followed as a result of a selection in the actual application.

5.2.2 GENERATION OF SELECTION INSTRUCTIONS

A selection frame is associated with a set of selection instructions contained in a selection interpretation table. A selection instruction corresponding to a given selection is composed of the following sub-instructions:

- a string instruction
- a procedure instruction
- a route instruction
- a mask instruction

The syntactic specification of the selection instruction in Backus Naur Form (BNF) is as follows:

<selection instruction>= <string instruction><procedure instruction>
<route instruction><mask instruction>

i) <u>String Instruction</u>

The string instruction specifies what will be the output of the selection. If the selection is not used for message building, no string will be moved. If the string on the frame represents a phrase for the message building, it will be moved as a result of the corresponding selection.

If the string on the frame is an abbreviation of a more explicit phrase or associate string representing the full text, it will be moved (also an abbreviation of the string on the frame can be moved instead of the full text.)

A string may be associated with a positioning format which will indicate where to move the string in the message area. The following choices are available:

- concatenation
- line feed/carriage return
- indentation
- tabulation

The corresponding instruction can be represented as follows:

S	A	F	Address of string

S is a boolean variable indicating the presence or absence of a string

A indicates if there is an associated string

F is a code for the format

The remaining part of the instruction is for the address of the string (either relative to the beginning of the frame or relative to the associate string area.)

This instruction can be built by selecting the corresponding atributes from the two following frames:

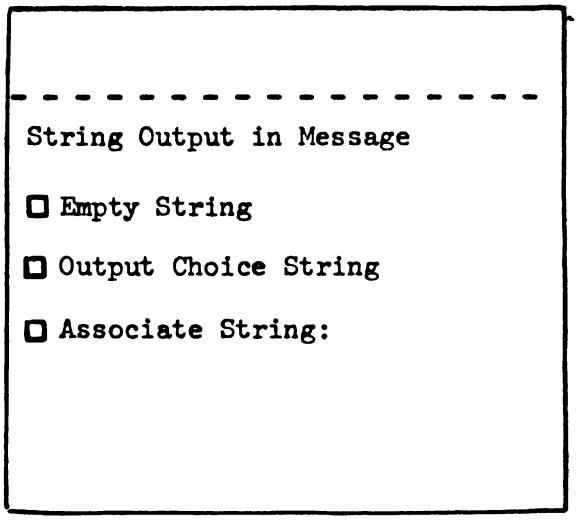


Figure 5.7 "String" Frame

the associate string can be entered on the frame just after the colon following the choice "associate string." If the moving of strings is requested, the next step will be to select the format from the following frame:

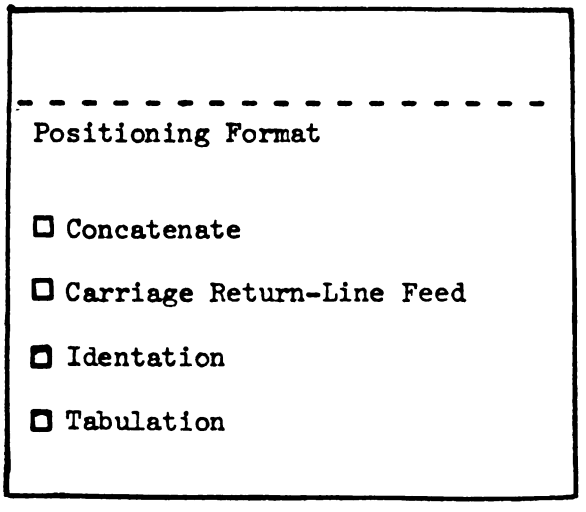


Figure 5.8 "Format" Frame

The address of the string is computed by a procedure executed after the selection of the string parameters and at that time the corresponding string instruction is compiled and stored in the selection interpretation table.

ii) Procedure Instruction

Following the string instruction, the next step will be the building of a procedure instruction to be executed as a result of a selection. The procedures which can be selected are listed by name in a procedure library. A procedure can be selected to be executed at a given level of priority: foreground, middleground or background.

Only foreground procedures will be executed immediately (between two frames), therefore, they must require a short execution time (no more than a few hundred machine instructions that is a few milliseconds.) A middleground or background procedure will be executed as parallel processes while the selections proceed in the foreground. The procedure code may be either resident in main memory or brought into an overlay area from the disc storage.

The following represents the procedure instruction:

P	PR	O	Address of procedure

P is a boolean variable to indicate the presence or absence of a procedure

PR is a variable giving the priority at which the procedure should be executed

O is a boolean variable indicating if it is an overlay procedure or not

The address of the procedure refers either to a table of resident procedures or to a disc address if it is an overlay procedure. The corresponding instruction can be built by selections from the following frames:

Procedure to be Executed

- ☐ No Procedure (NOP)
- ☐ Execute Procedure (Resident)
- ☐ Execute Procedure (Overlay)

Figure 5.9 "Procedure" Frame

Priority of Execution

- ☐ Foreground
- ☐ Middleground
- ☐ Background

Procedure Name:

Figure 5.10 "Priority" Frame

iii) Route Instruction

The route instruction defines the next frame to be invoked after a given selection is made. If it is a multiple choice frame, the route is invoked. For a selection frame, a selection will result in linking a new frame.

A given selection may also push a given frame on a pushdown store for future reference or pop a frame from the stack as shown in Chapter 4.

The following is a representation of a route instruction:

R	Pop	Frame Address	Push	Frame Address

R is a boolean variable indicating the presence or absence of route The variable 'Pop' indicates the next frame to be fetched. Two cases are distinguished:

i) If Pop is not zero and if the next field is not zero, it indicates the next frame address to be fetched; if the field "frame address" is null, the frame on the top of the stack is removed and fetched.

ii) If Pop is null no frame is fetched (multiple choice frames.)

The variable 'Push' indicates if a frame is to be pushed on the stack and the next field gives the address of the frame to be pushed.

The following frame is used for the route instruction:

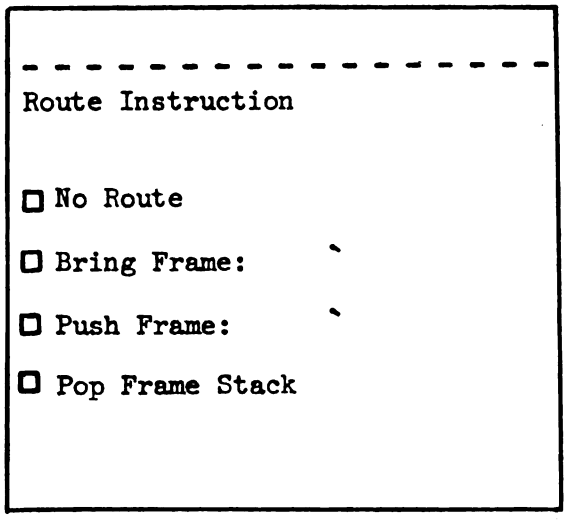


Figure 5.11 "Route" Frame

iv) <u>The Semantic Restriction Relation</u>

Associated with a given route there is a vector which specifies which choices may be selected on the next frame. This feature allows for semantic relations between frames, this is implemented by a vector mask which is applied against the subsequent frame when a given selection is made. The syntactic specification of this process is as follows:

<VECTOR MASK>* <SELECTION BOOLEAN>*

<SELECTION BOOLEAN>* TRUE|FALSE

This mask is built by bringing the next frame specified in the route, and selecting the choices which are to appear on the frame.

For example, this technique is used in the pharmacy application to restrict the route of administration to those which are semantically compatible with a given dose form.

The complete syntactic specifications of the frame programming language is given in the Appendix.

In this implementation, a selection statement corresponding to a selection instruction will look like:

```
FRAME 'FRANAM'
    CHOICE 'CHOICE PHRASE X'
      OUTPUT CHOICE STRING AND CONCATENATE
      EXECUTE PROCEDURE 'PROCDU' FOREGROUND
      BRING 'NEXTFR' AND PUSH 'FRANAM'
      NO MASK ON 'NEXTER'
```

This example tells that, for the choice 'CHOICE PHRASE X', the choice string will be moved and concatenated in the message area, a foreground procedure will be executed and the next frame will be 'NEXTFR' while the frame 'FRANAM' will be pushed on the frame stack. It must be noted that the keywords CHOICE, OUTPUT, AND, BRING...are automatically generated by the selection implementing the language FPL when the corresponding selection are made.

Although this minimal version of FPL provides all the features required to build the pharmacy application, new features can be added to the basic language by using the extensibility of the language. By designing new frames and new procedures, it is possible to give a greater flexibility to the language by introducing more stacks and more general semantic relations between frames elements.

5.3 SOME EXTENSIONS TO FRAME PROGRAMMING LANGUAGE

The existence of procedures to be executed as a result of a selection provides the capability to extend the language by simply replacing or adding new selections, new frames and new procedures.

In the following some improvements are suggested and can be implemented easily.

5.3.1 STRING PUSHDOWN STACK

A selection may invoke a string which will not be output immediately, but at some later point in the selection process. For instance when selecting a drug name one could invoke a string, which later will indicate the possible adverse reactions or side effect of the drug.

The string stack can also be used to output delineator or punctuation sign to be generated inside a statement or at the end of the statement (for instance a semicolon at the end of a statement can be pushed on the stack at the beggining of a new statement and removed from the stack when the statement is complete, the same is true for parentheses in nested expressions.)

This can be implemented by a new syntactic unit in the string instruction:

<stack string instruction>:= <PUSH><Stringaddress><String length><POP>

where PUSH and POP are booleans, which indicate if a string is to be pushed on the stack or popped from the stack. If PUSH and POP are true then it is equivalent to an immediate output of the string. If POP only is true then the string address and the length are null.

5.3.2 PROCEDURE STACK INSTRUCTION

Similarly a procedure stack can be implemented to invoke a procedure which may be executed at a later point in the selection process. For

instance, a procedure might be executed only if two particular selections are made during the selection process: the first selection will push the procedure on the stack, and the second will remove it from the stack and execute the procedure. If the second selection is not made, the procedure will be removed from the stack without executing it.

The corresponding additional syntactic unit is:

<procedure stack instruction>:= <PUSH><Procedure address><POP>

PUSH and POP have the same meaning as for the string stack instruction. This mechanism could be used for executing procedures which will check for drug-drug interactions or drug allergies.

5.3.3 THE SEMANTIC RELATIONS BETWEEN FRAMES

It was mentioned in Chapter 4, that it was possible to define relations between any pair of frames. Each frame f_i is considered as a linear vector of n elements c_{ik} k= 1,n representing the choices on that frame.

$$M_{ij} = \begin{array}{c} \begin{array}{ccccc} c_{i1} & c_{i2} & c_{i3} \cdots & c_{ik} \cdots & c_{in} \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & & 1 \\ 1 & 0 & 1 & & \\ & & & 1 & \\ & & & & \\ 0 & 0 & 1 & & 1 \end{bmatrix} \end{array} \begin{array}{c} c_{ji} \\ c_{j2} \\ \vdots \\ c_{jm} \\ \vdots \\ c_{jn} \end{array}$$

Each row indicates if there is a relation between all the elements of f_i (c_{ik}, k= 1,n) and the elements in f_j (c_{jm},m= 1,n). A zero indicates no relation and a one indicates a relation.

If one considers the columns, each choice of f_i (c_{ik}) is associated with a vector specifying the relation with each choice of f_j (c_{jm}). This is generalization of the vector mask defined before.

A semantic relation can be defined between any two frames by using a selection mask instruction:

<selection mask instruction>:= <selection mask vector><Frame ID>

Since a given frame may have several masks, the mask vector must be associated with a frame ID and when a given frame is called, all the mask vectors referring to that frame are logically "ored" to give all the semantic restrictions imposed by the previous selections.

5.3.4 DUPLICATION INSTRUCTION

For a given set of selections on a given frame or for a given set of frames of a given application, it is frequent to have to repeat the same selection instruction several times.

In the first implementation of the frame programming language, an *ad hoc* solution to this problem has been obtained by inserting a new frame which permits the duplication of the previous selection instruction for the next selection choice.

A more general solution can be obtained by defining a set of selection instructions as a named entity which can be referred to as a selection block such that:

<selection block>:= <block name><selection instruction>*

<block name>:= <character>*

then a new type of instruction can be defined to enable the duplication of previously defined selection blocks. The syntax specifications could be as follows:

<duplicate instruction>:= FROM<choice field>DUPLICATE<selection block>

<choice field>:= <choice index>TO<choice index>

<choice index>:= CHOICE<number>

Such an instruction could also be extended to the duplication of selection blocks from one frame to another.

This type of instruction will be particularly useful in frames structures where a large number of selections have the same function and therefore have a large number of identical selection instructions.

5.3.5 DATA BASE DESCRIPTION INSTRUCTIONS

Finally, a more elaborate frame programming language would include features to allow for the data base description of the content of each field and each message. For instance, each field of an editing frame would be associated with a variable name and a data type according to the following syntax:

<FIELD DESCRIPTOR>:= <VARIABLE NAME><DATA TYPE>

<DATA TYPE>:= INTEGER|ALPHANUMERIC|REAL

Similarly a message to be built as a result of a selection process might be described by a message descriptor according to the following specifications:

<MESSAGE DESCRIPTOR>:= <PHRASE COMPONENT>*

<PHRASE COMPONENT>:= <COMPONENT NAME><DATA TYPE>

Such an addition to the frame programming language would allow for a total description of the content of the data base resulting from the use of the corresponding frame selection system. Furthermore, this formal description would permit the building of a query language which could also be designed as a frame selection system thus enabling the generation of query statements to interrogate the data base.

To conclude this chapter, it is important to emphasize again the advantages of a frame programming language as opposed to a more classical approach (compilation or interpretation.) A frame programming language allows for a direct deterministic parsing of the statements generated, therefore, it eliminates the backtracking and indeterminacy encountered in a compiled approach. A frame programming language is not an interpreter, but a real-time generator of syntactically correct sentences. The statements generated by a frame programming language can be compiled for later execution, but the introduction of procedures in the selection instruction allow the extensibility of the language by modifying the procedures. Although, any programming language can be implemented as a frame selection system it is believed that such an approach is better suited to implement problem-oriented languages such as question-answering system, query languages, job control languages, computer assisted instruction, etc.

The next chapter will present a real-time operating system designed to support frame selection systems and the frame programming language as presented in the section 5.2.

Chapter 6

Implementation of a Frame Selection System

"An activity has to be understood in terms of the experience from which it emerges - these arabesques that mysteriously embody mathematical truths only glimpsed by a very few - how beautiful, how exquisite - no matter that they were the threshing and thrashing of a drowning man."

R.D. Laing

6. IMPLEMENTATION OF A FRAME SELECTION SYSTEM

This chapter will describe the implementation of a frame selection system on a minicomputer (Four Phase Model B).

Although the implementation has been done on a machine particularly well-suited for the support of multiple CRT terminals (see Section 6.3.4), the design is not specific to a given machine. Because of the particular requirements of frame systems (selection mechanism, response time) no useful software was available, and thus it was necessary to design a complete operating system to support the frame selection system application.

This chapter deals with the basic operating system design and with the implementation of a frame selection system as an external layer of the basic operating system.

The design of the frame selection system is related to the application systems and, in particular, to the implementation of a frame language which is implemented as a frame selection system application.

This chapter is more concerned with the programming techniques and tools used to implement the system.

6.1 BASIC OPERATING SYSTEM CONCEPTS

This section describes the overall organization of the operating system which is used for the frame selection system. The general features of the system are a real-time, multiprogramming system, with three hardware levels of processing. It is a system which allows the use of parallel processes. Before describing the implementation it is necessary to define some operating system concepts.

The object of the operating system is to handle all the tasks needed by the application processes. Since much emphasis is put on the use of CRT's, it is oriented towards a real-time interactive mode using a se-

lecting device which can be simulated by a keyboard. Rather than being developed for a specific application, the operating system has been developed to support a variety of applications.

The purpose of an operating system is twofold:

i) provide the potential user more flexibility than he would have by using the bare hardware machine

ii) provide for a management of the resources available on a given machine (memory, CPU, disc files, peripheral, etc.)

Although the term "flexibility" is hardly quantifiable, one can define a hierarchy of conceptual levels or layers at which the user becomes less and less dependent on the particular hardware structure of the machine. The ultimate goal is to transform the hardware machine which is fundamentally an undeterministic automaton (undeterminacy of the external interrupts) into a deterministic device where the system controls all the information coming in and out of the system in a predictable fashion.

The concept of resources is also very broad; a resource can be a memory word, a buffer, the central processing unit (CPU), a file, a processor, or a peripheral device. Pursuing the concept of resource management, the purpose of the operating system is to provide the user with the capability to work with more and more sophisticated entities which usually correspond to the abstractions defined at each successive level of the operating system.

From the users' standpoint, he would like to consider the machine as if it was working for him alone. The major problem of an operating system is not so much to build a deterministic machine for a particular user, as to be able to manage harmoniously the sharing of the resources by all the potential users of the machine. To precisely describe the problem, the concept of sequential process will be used.

6.1.1 SEQUENTIAL PROCESSES

This concept has been defined in a broad sense by Dijkstra [DIJ68]. It means that the rules of behavior of a process must be interpreted sequentially in time rather than simultaneously in space. These rules constitute an algorithm and the realization of this algorithm in a given language on a particular machine is a program. The events happening during the execution of a program constitute the sequential process.

An application programmer usually thinks of his problem as a unique, sequential process and thus does not want to know about the other processes that might interfere with his program if it runs in a multiprogramming environment.

It is the job of the operating system to insure that these interferences do not affect the results of other sequential process.

6.1.2 PARALLEL SEQUENTIAL PROCESSES

When more than one sequential process runs at the same time, they are called parallel sequential processes. A multiprocessor computer allows this feature for programs, but it should be pointed out that even for a monoprocessor computer, there might be several sequential processes running (I/O channels) simultaneously with the CPU.

In both cases, several sequential processes may have to cooperate with each other to perform a given task.

The usual high level programming languages do not allow one to define parallel processing because they were designed to describe single sequential processes for applications programs.

Dijkstra proposes to extend this feature to ALGOL 60 by the use of a special block "parbegin parend." Thus a block:

- parbegin S1; S2; S3 parend

  will indicate that the statements S1, S2, S3 will be executed in parallel.

6.1.3 COOPERATING SEQUENTIAL PROCESSES

When two or more parallel sequential processes have to cooperate with each other, they must interact in order to exchange information. Each of the individual processes are independent of each other except during those phases of interaction where they must access the same common variables.

Whenever two or more parallel processes need to use the same resource (resource being taken in a broad sense,) they will have to make sure that the resource is available. If the resource is to be shared effectively by several processes, the operating system must provide a mechanism which insures harmonious cooperation of the processes. The part of the processes where this can happen is called a critical section, and at any moment, at most one process is allowed to be executed in this critical section.

In order to carry out this mutual exclusion, the parallel processes have to access the same common variables. To avoid the same problem of critical section in the process of intercommunication, the operation of testing and changing each common variable must be indivisible: in some computers, it can be executed in one instruction such as the "TEST AND SET" instruction, more generally it can be implemented by preventing the interruption of the processor when processes are inspecting or modifying shared variables so that two processes can never do it simultaneously,

but only in sequence.) Another type of interaction is given in the case of processes which produce data which are consumed by other parallel processes.

6.1.4 SYNCHRONIZATION AND MUTUAL EXCLUSION

The mutual exclusion problem has been studied and solved by a team of Dutch scientists and applied to the computer field by Dijkstra [DIJ68]. Although the concept of mutual exclusion is simple, it should be pointed out that a correct and general solution to this problem is not trivial. The advantage of using it is that one can be sure that if competition for a resource arises the conflict will be resolved properly.

6.1.4.1 SEMAPHORES

In the solution given by Dijkstra [DIJ68], the common variables used in the synchronization or mutual exclusion of parallel sequential processes are called semaphores. The semaphores are integer valued variables which will be modified by the processes in an indivisible operation before and after a process enters a critical section.

6.1.4.2 SYNCHRONIZING PRIMITIVES

The indivisible operations used to modify the semaphore are called synchronizing primitives. They are symmetric operations on the semaphores which provide a mechanism to signal either the entering and leaving of a critical section or the synchronization of a consumer process and a producer process. In both cases, they usually are implemented by incrementing and decrementing a semaphore variable.

The solution to these problems is described in details in the corresponding literature [DIJ68,HAN72].

6.1.5 INTERACTIONS AND DEADLOCK PROBLEMS

The solution is satisfactory if each process respects the following rules:

- Each process must not enter a critical section without setting and checking the semaphore. If another process is already using the same resource, it must wait until the other one leaves its critical section. If a process is waiting for an event to happen, it must not proceed until the event occurs.
- Each process has to signal when it leaves a critical section in order to clear the way for other processes. If it is a synchronization problem, it must signal the event by modifying the semaphore accordingly.
- Each modification of the semaphores variables must be uninterruptible.

The problem becomes more complicated if processes are allowed to have nested critical sections. For instance, suppose that process 1 has entered critical section 1 corresponding to resource 1 (associated with semaphore 1,) and inside this critical section, it needs to access another resource which is accessible through a second critical section with semaphore 2. If, at the same time, another process 2 has already set semaphore 2 and wants to access resource 1 via semaphore 1, the resulting situation is a deadlock (or "deadly embrace.") Each process is waiting for the other to proceed but none of them can go ahead unless some external process is willing to resolve the conflict. This situation might result in a local deadlock or a complete deadlock of the system if the corresponding resources are needed by most of the processes (such as a disc or a file system for instance.)

It has been shown [COF71] that deadlock situations occur when all the following conditions are true:

i) Processes claim exclusive control of the resources they require (mutual exclusive condition.)

ii) Processes hold the resource allocated to them while requesting and waiting for additional resources.

iii) There is no preemption of resources held by a process until the resource is used to completion by the process.

iv) A circular chain of processes exists such that each process holds one or more resources that are being requested by the next process in the chain.

If one of these necessary conditions is not satisfied then a deadlock can be avoided.

In the following, it is supposed that condition (2) is never satisfied for the processes. This is possible because nested critical sections are not allowed. In other words, a process cannot have the exclusive use of more than one resource at a time. This is not unrealistic since for a given application most of the users are sharing the same set of programs and therefore most of the programs are re-entrant. In some cases where there is a unique resource (such as a line printer) shared by several users, the deadlock is avoided by having a first-in, first-out queue at the resource. If it is an interactive program, the system signals the terminal user that he might have to wait so that he may optionally abort the desired task to pursue another type of activity.

6.2 A TRANSACTIONAL OPERATING SYSTEM

In this section, a real-time transactional system is presented and has been implemented to support frame selection applications.

6.2.1 OBJECTIVES

The objectives of the system can be defined as:

- multi-user capability and multi-access to the resources
- CRT oriented system
- fast response time for CRT display
- multi-access to patient's file
- support of a selection mechanism for data entry
- low cost implementation

None of the known existing commercial or experimental operating systems met all these requirements at the time the project was initiated.

Among the real-time operating systems developed by minicomputer manufacturers, the multi-tasked systems are well adapted for industrial real-time process control where most of the interactions of the system are with other machines but not with humans. However, these systems are not multi-user oriented and even less CRT oriented. Most of them have used designs oriented toward the use of a single teletypewriter for the interaction between the user and the system. In these systems, multi-access to the files are usually not provided because in most applications, each task has its own files which are not accessible by other tasks at the same time. The directories are usually sequentially organized and accessed, which results in an unacceptable access time when there are a large number of files. Therefore, the conventional real-time operating system was not an appropriate model for our goals.

The other existing models are the more classical type of operating systems (batch-oriented or time-sharing.) Their common characteristics

is that they are based on the concept of "job", the task of the system being to allocate the resources to a given job, and to make sure that the integrity of the data is protected within a given job.

In a system where most of the files are common and shared by several users working simultaneously in real-time, the concept of job becomes useless and must be replaced by the concept of transaction because the integrity of the data base is more important than the integrity of individual jobs.

A transaction can be defined as an action which requires at least one access to the data base (either reading or writing.) A transactional operating system is a system which is built around the concept of transaction and its major task is to preserve the integrity of the data base.

It must be emphasized here, that a job-oriented system is usually very cumbersome to use for multiple, real-time access to a common data base. Even a time-sharing system, which may have shared files, usually provides a locking system which is too constraining for the real-time use of the data base.

In practice, a transaction may be characterized by the following parameters:

- a user number (identifying the originator of the transaction)
- a message (the content of the transaction)
- a time of origin (data and hour)
- a priority

These four parameters define what is called a transaction descriptor:

User #	priority	time	pointer to message

The concept of transaction can be extended to request for the use of any resource: in this case the pointer to the message can be considered as a pointer to a parameter list or table describing the way the resource should be used. For instance, an input-output request can be considered as a transaction dealing with a particular peripheral device. Therefore, the concept of transaction can be used uniformly as the basic entity of the operating system. If the transaction is generated by an interactive user, it will be called an external transaction which may in turn generate internal transactions to the data base by means of an internal transaction to the disc I/O driver. The importance of this unified concept is that only one mechanism will be needed to deal with any type of transaction (internal or external.) In particular, selections can be considered as external transactions which generate internal transactions in the form of messages to the data base.

To conclude this section, it is clear that a real-time, transactional system must be a multiprogrammed system, but instead of managing jobs, the system manages transactions using a common data base.

6.2.2 TRANSACTIONAL PROCESSES

Each transaction is associated with a transactional process, which represents the actions to be performed by the transaction. These processes are implemented by procedures or programs which perform the actions defined by the transactions.

In order to insure the sharing of a procedure by several transactional processes, the procedure must be either re-entrant or a queuing mechanism must be provided. If the procedure is not re-entrant, it is associated with a semaphore (in this case, it is equivalent to consider a non-re-entrant procedure as involving a critical section.)

If the procedure is re-entrant, it can be associated with a generalized semaphore [DIJ68] which indicates the number of transactions using the procedure at a given time.

The following figure represents the basic structure of a software module implementing the concept of a transactional process:

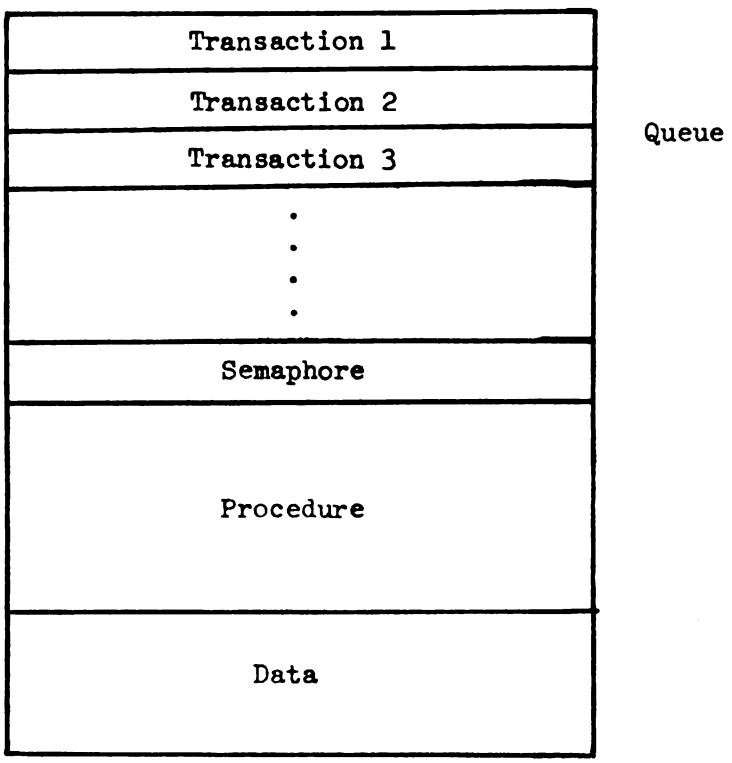


Figure 6.1

If the procedure is re-entrant, the procedure is called "pure" and the data area (impure section) must be divided into as many areas as there are users (each data zone being accessed via an index register.)

In the case of synchronization between two parallel processes, the procedures will be associated with a common semaphore as shown in the figure below:

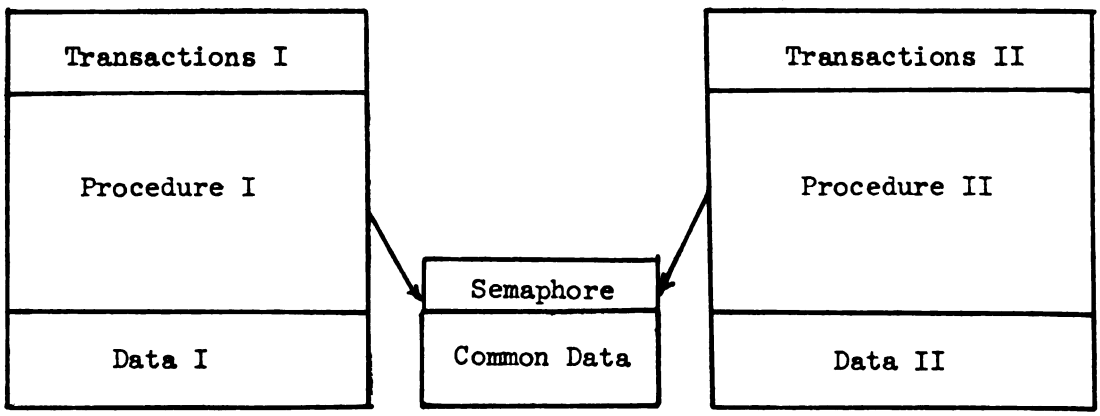


Figure 6.2

With these two basic software structures, a transactional operating system can be built so that several parallel processes can share common resources.

A process can be in one of three states:

- running (it is currently using the CPU)
- ready (the process can proceed as soon as the CPU is available)
- waiting (the process cannot proceed until it receives a signal or message from another process.)

When a process is initiated, it enters a queue of processes ready to run. When the CPU becomes available, the process becomes running. When a process is running, it may encounter a situation where it must wait for a signal or for the leaving of a critical section by another process.

In the case of mutual exclusion, a critical section is preceeded and followed by the two primitives:

- DECSEM (SEMAPHORE) (Decrement by 1 the semaphore variable)
- INRSEM (SEMAPHORE) (Increment by 1 the semaphore variable)

When no process is inside the critical section, the value of the semaphore is +1. If, after executing DECSEM, the semaphore has a negative value, then it means that one process is inside the critical section and the incoming process must wait. This is done by using the primitive QUEUE (process.) Before leaving the critical section each process must execute an INRSEM primitive.

If INRSEM is executed and the semaphore value is non-positive, then the primitive DEQUEUE (process) will be executed and the first waiting process in the queue will become ready and will be run if it has the highest priority.

The same primitives can be used for the synchronization problem.

6.3 THE ABSTRACTIONS AND LAYERS OF THE OPERATING SYSTEM

Following the hierarchical approach, defined by Dijkstra [DIJ71], the operating system has been designed by defining several abstractions which correspond to different layers of the operating system.

Each abstraction is intended to give to the user or the programmer a more flexible way to handle the machine.

For instance, a file system gives the user a flexible way to handle data records, items, keys, etc. but the operating system must perform the tasks of looking up the key in the directories, searching the indexes, issuing the I/O instructions to read disc sectors, etc.

Therefore, the operating system can be viewed as a set of concentric layers going from the hardware machine to the machine actually available to the end-users.

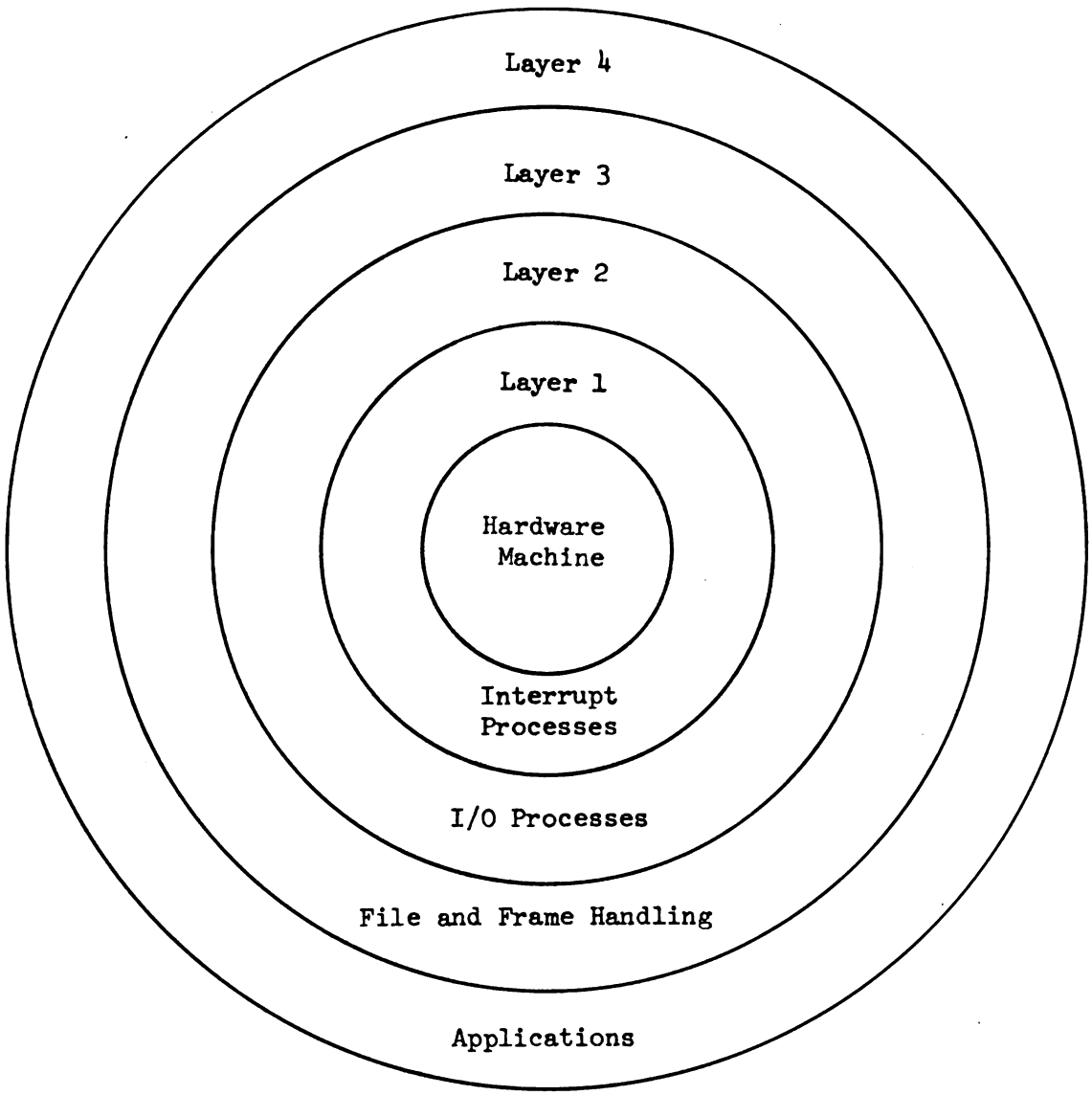


Figure 6.3

Each layer can communicate to the others by way of well defined primitives or macroinstructions.

6.3.1 INTERRUPT PROCESSES

The first layer, closest to the hardware, corresponds to the interrupt processes.

In the case of the machine used for the implementation described later (Four Phase,) the interrupt processes are:

- the clock
- the disc controllers
- the keyboards entry mechanism (including the selection mechanism.)
- the line printer controller

These processes recognize the nature and origin of the interrupts, save the context of the interrupted process (registers and program counter,) and take the appropriate action corresponding to the type of interrupt (increment the clock count, read a character, read or write a disc sector or output a line on the printer.) These processes are executed according to a hierarchical hardware priority which enables the preemption of the processes with a low priority (see Appendix). In addition, two levels of programmable interrupts are used for the execution of foreground and middleground tasks.

6.3.2 INPUT-OUTPUT TRANSACTIONAL PROCESSES

The next layer corresponds to the I/O transactional processes.

An input-output transaction is defined as:

- the input of a character string (the end of the string in the character EOM)
- the input or output of contiguous sectors on a given disc drive
- the output of a continuous string of characters on a given line printer.

With each of these type of transactions is associated a transactional process as defined above.

The format of the primitives and the tables used to access the procedures and queues implementing these processes are given in the Appendix.

These processes are, in fact, the front-end of the interrupt processes as far as the I/O functions are concerned. At this level, the user ignores the indeterminacy of the interrupts and the real hardware I/O instructions.

This layer also accomplishes the sharing of the I/O devices by multiple users in a way that is not apparent to the user. This is done by a queue associated with each input output procedure. The disc input output procedure can be accessed from the three priority levels (foreground, middleground, background.)

6.3.3 SCHEDULING AND MULTIPROGRAMMING

Before defining the next layer, it is necessary to describe the scheduling and multiprogramming mechanisms. According to HANSEN [HAN72], a monitor should be considered as an extension of the hardware to give a multiprogramming machine. Following this view, the priority scheduling function has been delegated to the hardware by using programmable interrupt levels.

There are three levels of priority:

- the foregorund (activated by programmable interrupt)
- the middleground (second programmable interrupt)
- the background (normal level of execution)

A transaction can be originated from any interrupt level and can be passed to any priority level by using a re-entrant procedure called STAPRO. Each priority level has a scheduling (first in, first out) queue for processes which are candidates for activation (either because they have been called by another process or because they are ready after a

wait for an I/O request.) The multiprogramming is done by switching from one user to another when a user process makes an I/O request. This is reasonable because the frames and files are stored on the disc, and therefore, each user transactional process is composed of one to several disc accesses for frames or files. This access time allows adequate CPU cycles to insure the multiprogramming between users and the frequency of frames accesses is such that the multiprogramming is effective.

Further assumption is made that if there is a longer computation to be carried out without an I/O request, then such a procedure is executed in the middleground or the background. So each type of transaction will be executed at different levels of priority depending on their expected execution time, the urgency of the response required, and the I/O pattern of the transactional process.

Since the priority scheduling is done at different hardware levels, the overhead required to switch from a task at a given level to another more urgent task is minimal. In the implementation of the frame selection system, the frame handling is done in the foreground in order to achieve a good response time, independently of the nature of the middleground and background transactions. In some cases, the need arises to communicate data between these levels and this is done by using a common message area (mailbox,) which can be filled or emptied, by two processes executed at different levels.

The following diagram represent the general architecture of the operating system:

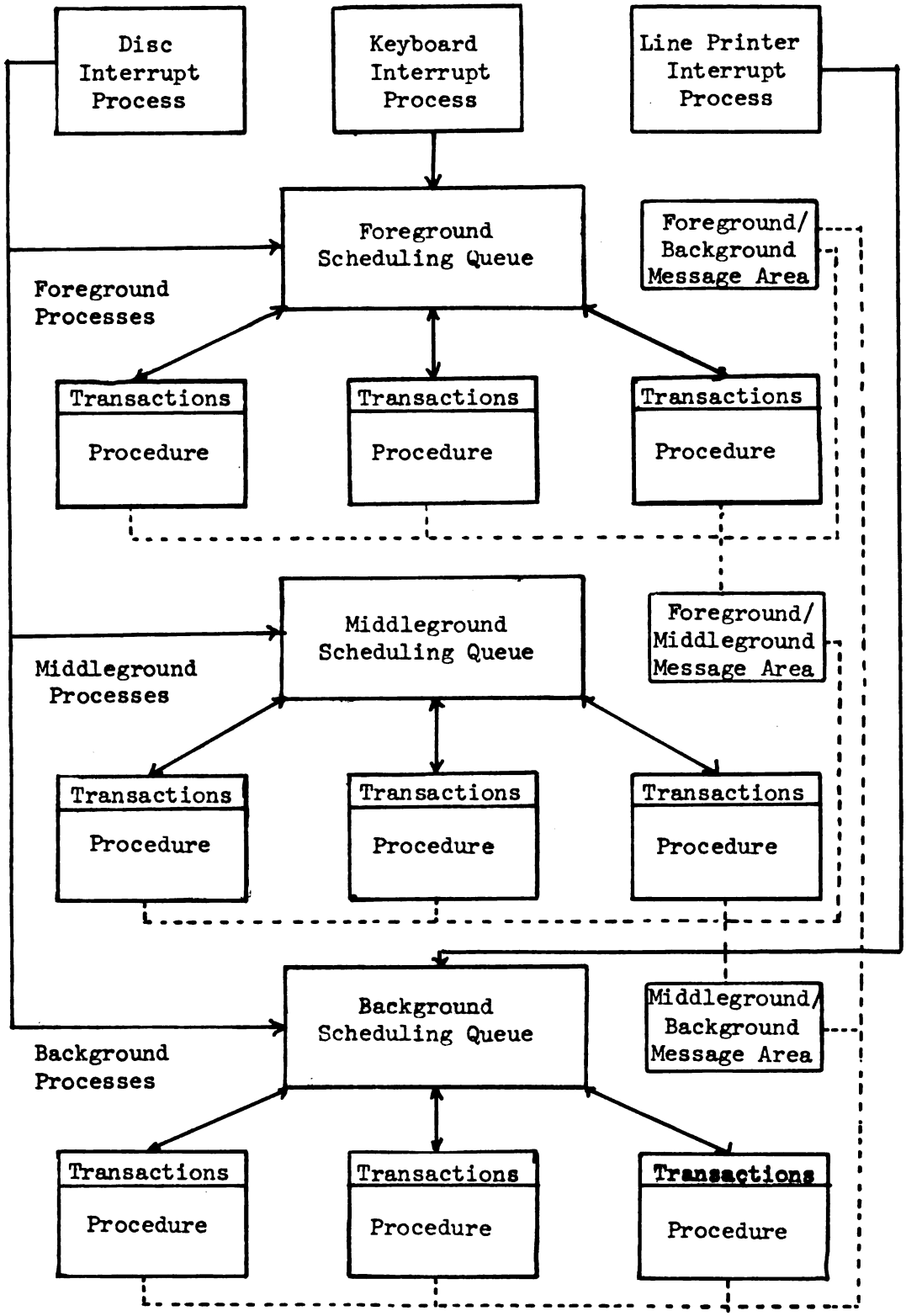

Disc Interrupt Process
Keyboard Interrupt Process
Line Printer Interrupt Process
Foreground Scheduling Queue
Foreground/ Background Message Area
Foreground Processes
Transactions
Procedure
Transactions
Procedure
Transactions
Procedure
Middleground Scheduling Queue
Foreground/ Middleground Message Area
Middleground Processes
Transactions
Procedure
Transactions
Procedure
Transactions
Procedure
Background Scheduling Queue
Middleground/ Background Message Area
Background Processes
Transactions
Procedure
Transactions
Procedure
Transactions
Procedure

Figure 6.4

6.3.4 THE FRAME SELECTION LAYER

The previous layers described were not specifically designed for the frame selection system and could have been used to develop any type of multiprogramming system.

The next layer implements the abstraction of frame selection systems. Although in this layer the structure of the hardware machine is hidden by the previous layers, one characteristic of the Four Phase minicomputer has been extensively used in the design of this layer, namely the fact that dedicated portions of the main memory are directly displayed on the CRT terminals. This characteristic is particularly well suited to the implementation of a frame selection system because it eliminates all the overhead necessary to support the CRT's. A frame can be read directly from the disc into the main memory display areas, and the frame image will appear automatically on the screen. From a system's viewpoint, a frame is composed of two contiguous disc sectors (768 characters each;) the first sector contains the text to be displayed and the second sector contains the selection interpretation table and the semantic restriction table.

The text of the frame is compacted in the first sector by ignoring the blank lines and conversely by using a decompaction mask the total screen image (1152 characters) can be displayed on the CRT. If it is an editing frame, the maximum number of text lines is 12, and the remaining of the sector is used to store field descriptors which indicate the starting address and the length of each field. If it is a selection or mixed frame, the maximum number of lines is ten, arranged in a 3 x 10 choice matrix. The selection interpretation table is the compiled version of the selection instructions generated by the frame programming language described in the previous chapter.

The next figure shows the content of two sectors composing a frame:

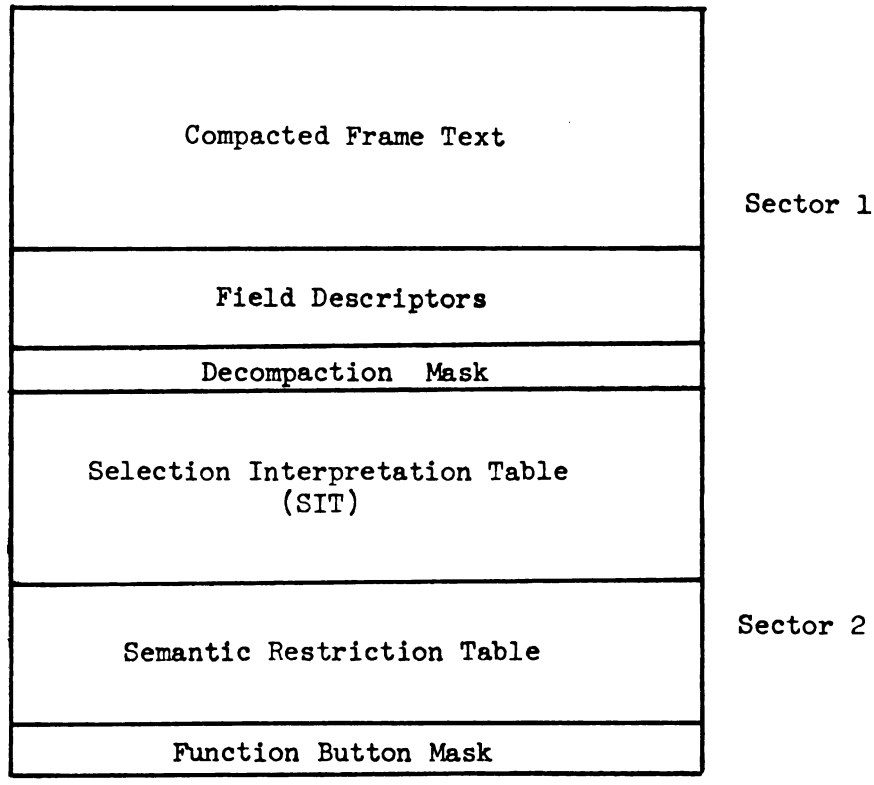


Figure 6.5

The function button mask is a variable associated with each frame and is used to specify the function keys which are permitted with a given frame.

When a frame is demanded by the frame selection system, it is read into a two sector buffer (512 words) corresponding to a screen area. Then the field descriptors and the selection interpretation table are moved into some other internal buffers and the frame text is decompacted according to the decompaction mask.

After decompaction, the frame content is structured as follows:

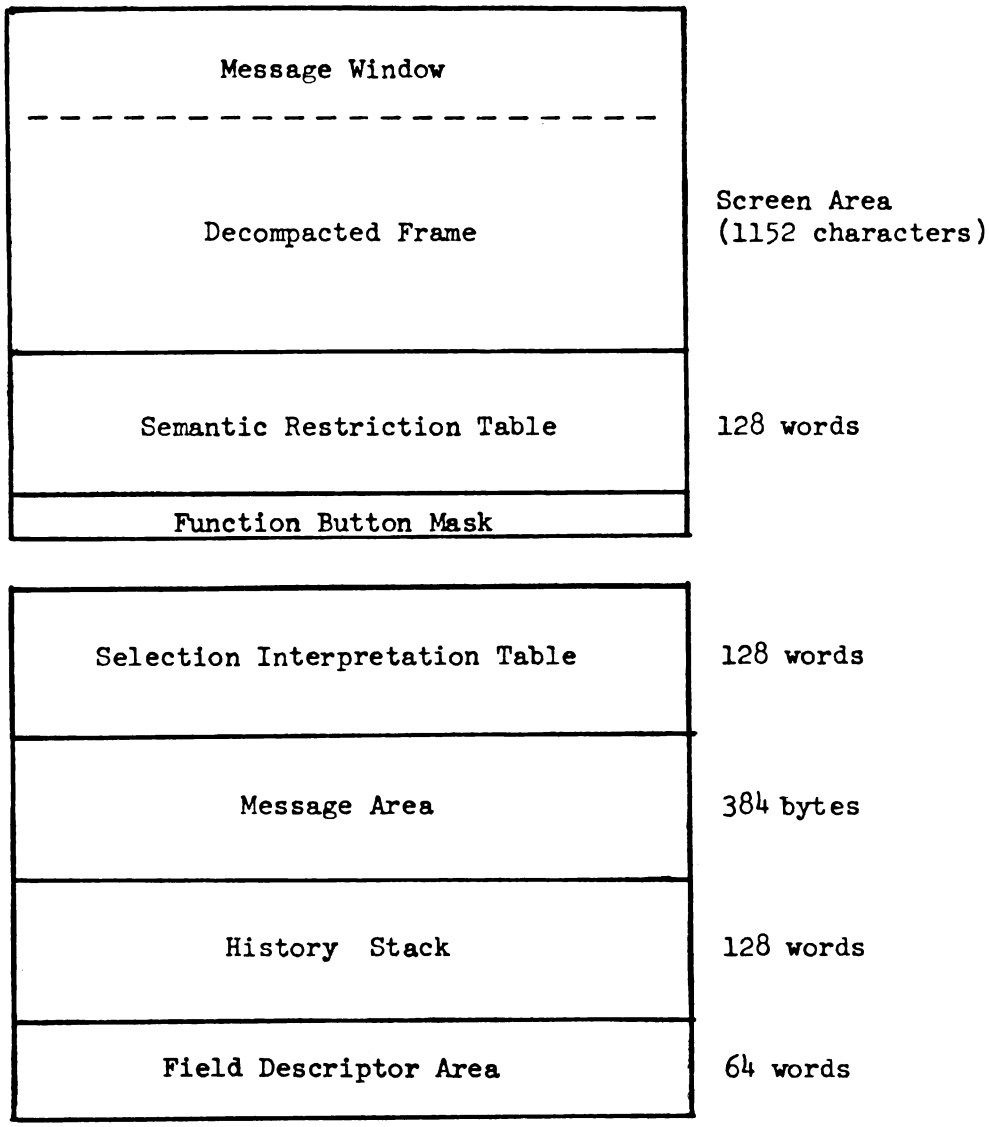


Figure 6.6
Users' Buffers and Tables used by the Frame Selection System

In addition to the content of the frame as stored on the disc, some other memory buffers are necessary to store the dynamic data generated by the frame selection system. These buffers are: the <u>message area</u> in which the text of the string generated by the successive selections will be stored and the <u>history stack</u> which keeps track of the previous frames and pointers referring to the message already generated so that it is

possible to reverse the selection process. Some other miscellaneous stacks and variables necessary for the operation of the frame selection system are not presented on the figure because they take only a few memory locations.

The message window seen on the top of the screen area is a duplication of a portion of the message area which corresponds to the last four lines (4 x 48 characters) generated by the selection process.

6.3.4.1 THE EDITING MODE AND THE SELECTION MODE

In the previous chapter, three types of frames were defined: editing frames, selection frames and mixed frames. Accordingly, there are two possible modes of operation associated with each type of frame:

- the editing mode or the selection mode.

With each of these modes is associated a number of function buttons and keyboard activated editing functions. When in the editing mode, a simple editor can be executed in the middleground. All the editing functions are programmed and special keys on the keyboard are used to move the cursor (one character left or right, one line up or down, at the beginning of a line or the top of screen,) to insert and delete characters, to erase a line or the total screen, to move to the next tabulation position, etc.

In addition, a protection mechanism is implemented for the editing frames in order to avoid the erasure of the text representing the prompts.

The editor is activated by a function button (F1) and the end of typing is signaled by the EOM (end of message) key.

The selection mode is the normal mode of operation, but as far as the software is concerned, a selection is received as an input of a character from the keyboard. This allows for the simulation of a selec-

tion device (light pen, touch-screen, magnetic wand, etc.) by a normal keyboard so that the addition of several type of selection devices does not require any reprogramming.

An alternative back-up mechanism is provided by using the normal keyboard to move a solid cursor on each selection square represented on a frame. This is done by using the key island represented below:

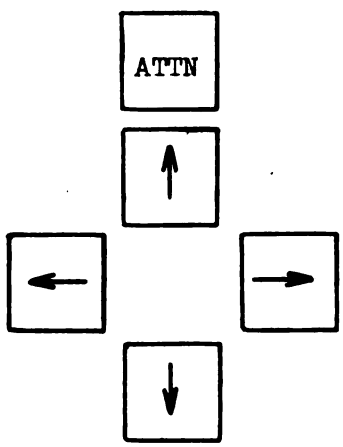


Figure 6.7

the ATTN key brings the cursor at the first selection point and the other keys move the cursor in the direction specified by the arrows. When the cursor is positioned on the desired selection point, the EOM key is used to confirm the selection.

The selection mode is associated with a set of procedures executed in the foreground level and which constitute the selection executor.

6.3.4.2 THE SELECTION EXECUTOR

Following a selection, the executor will decode the instruction contained in the selection interpretation table relative to the frame currently displayed.

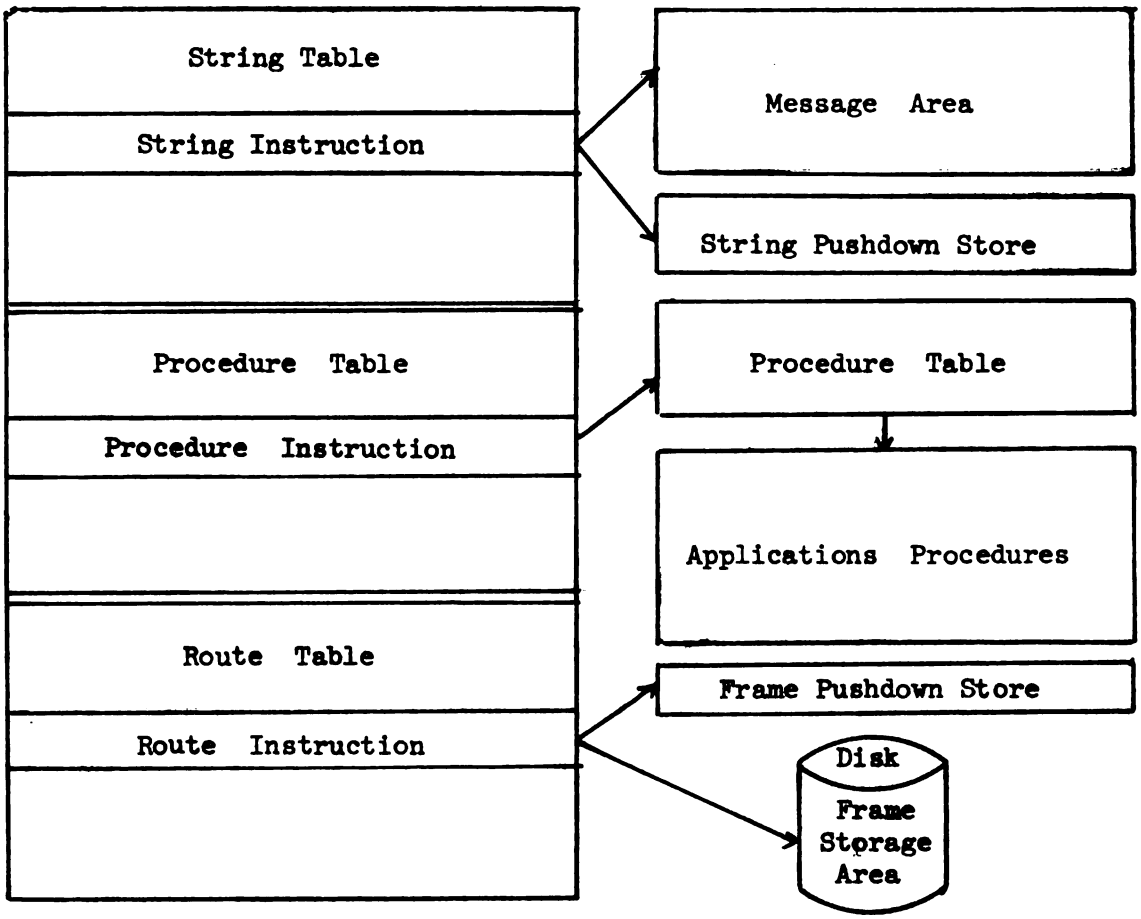
Each frame is associated with such a table which indicates the operation to be accomplished as a result of a selection:

- The string next to the selection point may be moved into a message area.

- An associate string (full text or abbreviation not displayed on the screen) may be moved into the message area.
- A string may be placed on a pushdown store and moved later into the message area.
- A procedure (resident or overlay) may be executed as a result of the selection (a foreground task.)
- A middleground or background process (resident or overlay) may be invoked and executed later in the corresponding priority level.
- A new frame might be called as a result of the selection.
- A frame address may be placed on a pushdown store and may be removed later.

All these actions corresponds to the application designed as a frame selection system: the executor function is to move strings, which can be used for building sentences of an application language or to call procedures and processes which perform some computation for the application.

The structure of the selection interpretation table used by the selection executor is presented below:



Selection Interpretation Table

Figure 6.8

Each time a selection is made, the executor keeps track of the previous frame address, together with the corresponding semantic restriction mask and the pointer to the last string moved to the message area. These data are stored into the history stack and constitute a history vector shown below:

frame address	semantic restriction mask	pointer to message area

History Vector

This vector can be used in relation with a special function button which enables the user to execute the selection process in the backward direction. This feature can be used to investigate some selection paths and then go back to a previous branching point or to make a correction in the message and then proceed forward again.

Some other function buttons are used to review the message already generated, to erase the total message and to store the data entered on an editing frame.

In order to avoid the misuse of the functions buttons, the mask associated with each frame will prevent the indiscriminate activation of functions which are not desired with a given frame.

The selection executor is the most important part of the operating system because it is directly related to the response time.

In order to maintain a good response time, it is desirable to limit the procedures invoked between two frames to an execution time not greater than the average seek time of the disc drive. The processes executed in the middleground or background levels do not cause a degradation of the response time because such tasks are pre-empted by the foreground level (the selection executor and function buttons.) Therefore, the response time on the CRT's depends only on the number of users actively involved in a selection process at a given time. Experiments have shown that with a rate of twenty selections per second involving the reading of a new frame , the response time was about half a second.

6.4 THE FILE SYSTEM

The frames are stored under a simple filing system, enabling the creation and deletion of frames via a hash coded directory [PRIII]. The name of the frame is used to generate a number taken as a sector

address in which the frame's name and address will be stored.)

Each frame is composed of two contiguous sectors; the first sector contains the compacted text of the frame, and the second sector contains the selection interpretation table (SIT.) Since the frames are the most frequently accessed files, they are stored in the middle of the disc area in order to minimize the seek time from one frame to another.

The application's filing system (patient's file) is characterized by a dynamic allocation of disc space with a quantum equal to one sector (768 bytes.) There is no limit on the length of a given file. The system maintains a triple directory access: by name, by medical record number, and by location (bed location.)

Each file is divided into pages and the pages are linked forward and backward. A page is itself divided into items which represent either a field entered on an editing frame or a message built by the frame selection system.

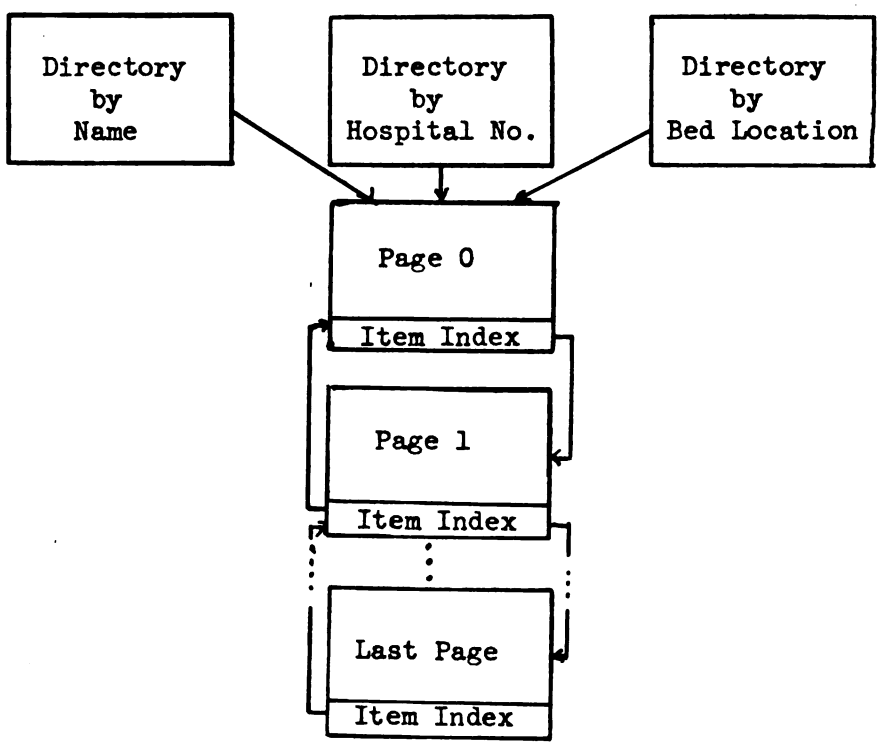


Figure 6.9 - Structure of the Patients' Files

The data items are retrieved within a page by a set of pointers called the item index which gives the position of the item within the page.

To conclude this Chapter a general overview of the system is shown below. From left to right, the different modules are:

- KEYB: the keyboard and selection device interrupt drivers.
- FQUEUE, MQUEUE, BQUEUE: are respectively the queues for the foreground, middleground, and background ready processes.
- SCH: is the scheduler which utilizes these queues
- SELEX: is the "selection executor"
- FUNC: is the "function button" handler for eleven function buttons.
- EDIT: is the CRT editor
- LINEPT: is the line printer and label printer drivers.
- DIR, FIL, ALLOC: are the modules of the file system and stand for directory handling, file handling and dynamic allocation of disc space.
- DISC$^{F}_{M}$: is the moving head disc routine (accessible by the three levels: foreground, middleground and background.)

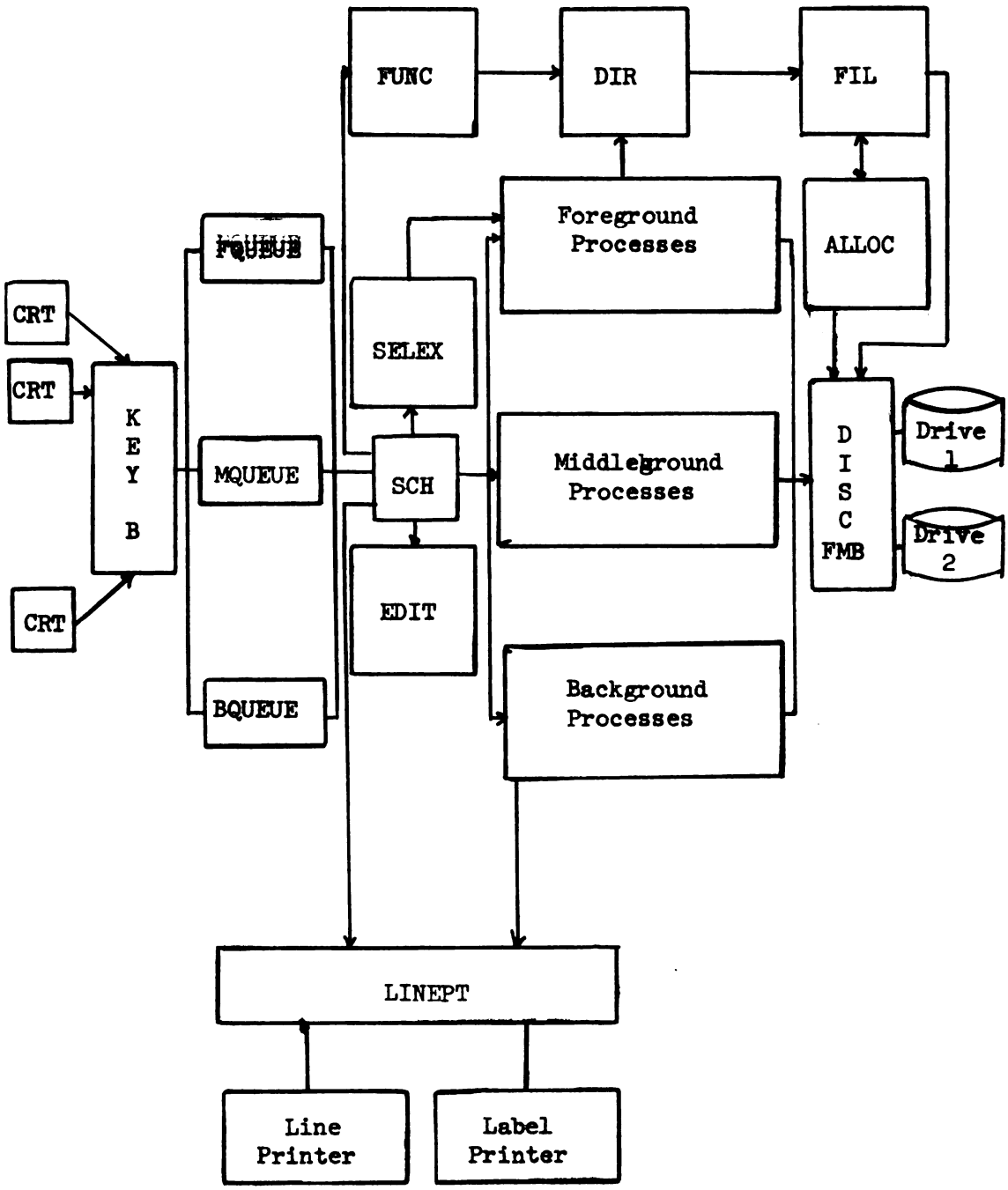


Figure 6.10 - Software System Block Diagram

Chapter 7

A Pharmacy Application, Utilizing the Frame Selection System

"Let us present our duties with some humility as it gets dark now. In case there is no paradise, we will not have soiled the curtain with our tugging, but actually I'd rather take a nothing I loved to my grave than a something I have every reason to hate."

Kenneth Patchen

7. A PHARMACY APPLICATION, UTILIZING THE FRAME SELECTION SYSTEM

This Chapter describes an inpatient pharmacy system developed as part of this research. It was implemented to prove the feasibility of the proposed language, operating system and the low cost modular concept.

Previous attempts to build automated pharmacy systems [GOUV69] have shown that one of the problems was the detection of typing errors due to the multiplicity and complex spelling of drug names; if one allows the use of completely free text typing, much computation and many file accesses must be done in order to check the validity of the order; on the other hand, if abbreviations or codes are used for drugs, the data entry cannot be done by medical professionals without awkward reference to manuals or lists. To document this assertion, the first section of the Chapter reviews the different types of automated pharmacy systems developed in the past, as well as those current systems under development.

7.1 REVIEW OF PHARMACY COMPUTERIZED SYSTEM

Comprehensive reviews of the literature concerning the use of computers in the hospital pharmacy, are found in [KNI73,GOU71]. In the following, only a few of the most significant systems are presented briefly.

The first type of systems implemented were off-line computerized cost accounting systems [DEE60], where coded pharmacy requisitions were keypunched, a tabulation of drug expenses was produced and, in some cases, statistics on drug usage were generated in monthly reports.

Other similar types of systems were designed to improve the management of the pharmacy department by automating administrative tasks such as inventory control, purchasing and patient billing. A paper by Winters and Hermandez [WIN72] describes an on-line inventory control system where CRT terminals were used to enter purchasing and inventory data while

purchase orders were automatically generated when the minimum re-order level was reached. Those systems were successful because they improved the management of the pharmacy and were easy to implement and economically justify. Such systems had the beneficial effect of standardizing and unitizing the drug items dispensed by the pharmacy.

At the same time the development of manual unit dose drug distribution systems was initiated. In such a system, a package containing a predetermined amount of drug (or supply) for one usual dose, application, or use, is prepared by the pharmacy service and presented, as such, to the nurse for direct administration. The advantages and disadvantages of unit dose systems are suggested in a paper by Miller and deLeon [MIL72]. Among the attempts to automate such systems (which are referred to as drug distribution and medication systems), Simon [SIM72] describes a semi-automated approach where each drug item is keypunched with patient data and used by technicians to fill unit dose carts and to generate patients' profiles. A similar technique was described by Johnston [JOHN70] where drug orders were transcribed on cards which were sorted four times daily and used in filling unit dose carts. The cards were then duplicated and used to tabulate the pharmacy charges for each patient.

The first attempt to build a totally automated and on-line medication system was described by Gouveia [GOU68]. The system was developed as a prototype on a time-sharing computer, using CRT's and teletypewriters as terminals. The drug orders were entered by typing and therefore, much of the programming effort was expended in writing programs that would check the accuracy and rationalness of the medication orders. In a later paper [GOU71] Gouveia gives a comprehensive analysis of the difficulties encountered.

The Automated Medication Order System (AMOS) is described by Melrose [MEL70]. This system is used by pharmacists who enter coded orders at the terminal and, in return, the decoded items are presented to the user for verification and validation. If the displayed data is confirmed, a label is printed automatically.

The system developed at the University of Southern California and described by Maronde [MAR72], was also designed for on-line data entry of coded drug orders. Each order was decomposed in five fields which were decoded and displayed for verification and a label was printed. The system is operational but due to the size of the computer used (IBM 360/40), the operating costs are high.

Another type of system which is operational at the Johns Hopkins Hospital in Baltimore is described by Zellers [ZEL73]. It uses the central computing facility of the hospital (IBM 370, Model 135 computer) and special terminals with matrix keyboards. The matrix keyboards hold "mats" which can be interchanged, thus changing the information which a particular key represents. An automatic sensing device distinguishes which mat is being used. This is used as a "drug selection keyboard" which offers a set of one-hundred selections on each mat (the most commonly used drugs are on a single mat.) Together with this matrix keyboard, an acknowledgement keyboard is used to enter the drug administration information into the patient's record.

A more elaborate system is the MEDIPHOR System (Monitoring and Evaluation of Drug Interactions by a Pharmacy-Oriented Reporting System), developed at Stanford University [COH72]. The system has been developed as a MUMPS application system and the entry of data is performed by responding to a series of prompts on the screen. A coding mechanism is used to enter the drug name. The pharmacist then enters the dosage

regimen. The most interesting aspect of the system is the automatic checking of drug-drug interactions by the programs developed for that purpose. The system is currently operating on a rather powerful computer (PDP 11/45 backed up by another PDP 11.) The cost effectiveness of the system has yet to be determined.

In addition, other pharmacy systems have been developed as part of overall hospital information systems which were described in Chapter 2. Usually in these sytems, the method used to generate the drug orders was the menu-tree selection or frame selection approach.

To conclude this brief overview of the literature concerning automated systems, it seems that most of the on-line systems are faced with the problem of avoiding the use of free text typing to enter the drug orders. Two alternatives seem to be used: the utilization of codes or a menu-tree selection approach. This survey shows that no stand-alone pharmacy system has yet used the frame selection approach. In the following pages a stand alone pharmacy system is presented; its implementation shows the feasibility of building such a system on a small dedicated computer.

7.2 THE REQUIREMENTS AND THE CONSTRAINTS OF THE PHARMACY APPLICATION

Since the pharmacy system was designed to be used in a real production environment, a certain number of requirements and constraints were to be considered.

7.2.1 OBJECTIVES

The pharmacy information system is designed to accomplish the following specific objectives:

i) Inpatient drug ordering system: the system provides a data entry mechanism to capture all types of drug orders generated in the hospital.

ii) Generation of a patient drug list for each hospital inpatient: for each hospital inpatient, this system records drugs names as they are ordered, together with the starting date, the stopping date, the route of administration, the dose and the dosing schedule. Various other patient data such as sex, age, weight, medical service, bed location, allergies, and diagnosis...etc. are captured and stored in the profile.

iii) Capture of drug billing items: the system automatically captures the drug information necessary for the preparation of the patient's drug bill.

iv) Pharmacy inventory maintenance: the system records inventory changes necessary for inventory update and maintenance, as well as assistance in placing orders for drug supplies.

v) Generation of medication schedules for Nursing service: the information system prints out medication schedules by room and bed number for each shift of the nursing service indicating the drug dose, route, and administration schedule for each patient.

vi) To provide a data base adequate to support drug allergy and drug-drug interaction warning systems: the initial system will involve clinical pharmacists who will check for possible allergies and drug-drug interactions, but as suitable drug allergy and drug-drug interaction programs become available, they can be incorporated into the system.

7.2.2 CONSTRAINTS ON SYSTEM DESIGN

The conception and design of the system was constrained by a number of factors:

i) the operation of the system, in particular, data entry, was to be performed by hospital pharmacy personnel. The existing data

source and formats in use in the pharmacy were to remain unchanged. At a later time, the entry of drug orders into the system may be performed by nurses or nursing clerks from the wards, and at this point, entry by physicians would be possible. These requirements in turn imposed some terminal constraints on the pharmacy system, since a homogeneous system was desired due to economic and backup factors.

ii) Terminal selection was to be made primarily on the basis of human factor considerations.

iii) Although our hospital was committed to the eventual use of a unit-dose drug dispensing method, this system had been only partially implemented. Approximately 170 beds were serviced by a unit-dose system and the remaining 380 beds continued under the more conventional individual patient order dispensing system.

iv) The economic requirements were such that the system must have the potential of realizing operational and development costs associated with the system. These constraints, or variations of them, result from the realities of hospital operations and are faced by most designers of hospital information systems.

7.2.3 HARDWARE SELECTION

The hardware was selected in an attempt to optimize the operations and performance of the major portions of a hospital information system, rather than the single pharmacy module alone. The hardware selection criteria reflected these constraints and other considerations which included:

i) Terminal and system flexibility

ii) Economy for modules having 8 to 16 CRT terminals apiece

iii) Reliability

iv) Ease of repair and/or replacement

v) Low power consumption in order to avoid the need for air conditioning and to make feasible emergency battery power

vi) State of the art design so that obsolescence would not be a factor during a development period of 3 to 4 years.

The system must be capable of supporting a CRT frame display system, and a selection mode of operation with a response time for frame replacement of less than one second. The reliability must reflect itself in a mean-time-between-failure of the hardware system of at least 5,000 hours. In order to facilitate rapid repair, the system components should be small and of light weight so as to be manually replaceable by a spare unit in approximately thirty minutes. The diagnostics should be adequate to allow identification and replacement of malfunctioning circuit boards in about 15 minutes.

The cost that could be justified for any given hospital subsystem varies, but the major applications (i.e., admitting, clinical laboratory, pharmacy, radiology, central supply, and ward systems) must all be cost-effective _in toto_. The most likely cost savings in the administrative areas are the capture of lost charges, keypunch cost recovery, and reduction of professional labor costs. The medical benefits of the system, although large and of major concern in the design, were not counted in the cost justification of the system due to the difficulty of quantitative economic evaluation of increases in the quality of patient care.

The Four Phase Systems, Model IV/70, an all LSI (large scale inte-

grated) CPU-terminal controller was selected for the implementation. This system is an integrated CPU and CRT system, which allows the direct and CPU independent refresh of video monitors from as many as 32 banks of LSI memory. The direct viewing of the memory does not in any way subtract cycles from the relatively slow 24 bit CPU which controls keyboard inputs and all other normal computer peripherals (i.e., printers, discs, tapes and communications controllers.) The CRT monitors connected by coaxial cables) can be located up to 2000 feet from the CPU controller, and could display 48 characters x 24 lines, as well as smaller subdivisions. The 132 symbol character set includes upper and lower case letters, and symbols providing for limited graphics. The unit is electronically well suited for the integration of a selection device, (i.e., a touchscreen, joystick, magnetic wand or special keyboard.) The unit is physically small, requires no special power supply (800 watts) and no air conditioning. Multiple discs from 2-1/2 million bytes to 58M bytes/drive are available, as well as a range of standard tape drives, printers and communications equipment. For the pharmacy system, eight terminals are supported by 48K bytes of memory, and two 2-1/2 million byte disc drives and a spare 2-1/2 million byte disc unit for backup. A 500 line per minute electrostatic, 132 column, 11-1/2 inch wide, printer is attached to provide fast silent printed output. Several (1 to 5) label printers can be used on the unit for labelling drug containers.

The light weight of the total electronics package and the manner in which connections were made, allows complete system replacement in a short time. The vendors diagnostic software was sufficient to isolate single card failure quickly.

7.3 THE DATA FILES IN THE PHARMACY APPLICATION

The data bases used in the pharmacy application are:

- the frames which provide the selection system
- the drug formulary
- the patient profiles

These data bases are stored on a unique disc cartridge of 2.5 million bytes. The cartridge is composed of 200 cylinders of 16 sectors and a sector contains 768 bytes.

7.3.1 THE FRAMES

The number of frames used in the pharmacy application is about 200 frames. Each frame, including the selection interpretation table, is stored in a two sector contiguous area. A frame directory gives the address of the first sector of each frame. Since most of the disc accesses will be for reading frames, the frame area has been placed in the central area of the cartridge in order to minimize the movement of the disc drive arm.

The total space required for the frames is about 400 sectors (25 cylinders.)

7.3.2 THE FORMULARY

The formulary contains the following data:

- an item name (usually a drug) associated with some code to be used for drug-drug interaction, drug utilization review and adverse reaction warning
- a set of dose forms and strength followed by the real cost and a price to be charged for a unit (if the dose is not unitized, the unit is either the volume unit (ml) or the weight (mg or g)
- a minimum inventory level to be maintained and the current inventory level

The formulary file is organized as an indexed sequential file and is also stored in the central area of the disc cartridge. The total number of name entries in the formulary is about 1200 with a total number of 2600 billing items.

The total formulary list occupies about 256 sectors on the disc. The formulary inventory is not updated in real-time, but on a daily basis when the billing program is executed.

7.3.3 THE PATIENT PROFILE

The patient profile is the only non-static file of the application. As previously described, the data is stored as a two-way linked sequential file with three ways to access the file: by the name of the patient, by the patient's hospital identification number, or by the patient's bed location.

The pharmacy profile is composed of two typed pages, entered via keyboard and associated with editing frames. The first page contains all the admission data: names, birthdate, sex, admission problem, physician name, admission diagnosis. The second page contains notes, to be entered by a clinical pharmacist, on allergies or adverse reaction to drugs.

The following pages are composed of the drug orders prescribed for the patient during his hospital stay. These pages are filled as a result of a message building selection process using the frame selection system approach. The directories which are updated in real time are stored next to the frame storage area in order to minimize the seek time.

According to the historical statistics kept by the hospital pharmacy, a typical patient's drug profile contains an average of 10 to 14 drug orders for a stay of 7 days. This amounts to an average of 3 disc sectors to store a typical patient profile.

For the University of California Medical Center, containing 550 beds, the total space required for the patient profiles is:

$\simeq$ 500 X 3= 1500 sectors

The following scheme details the organization of a disc cartridge:

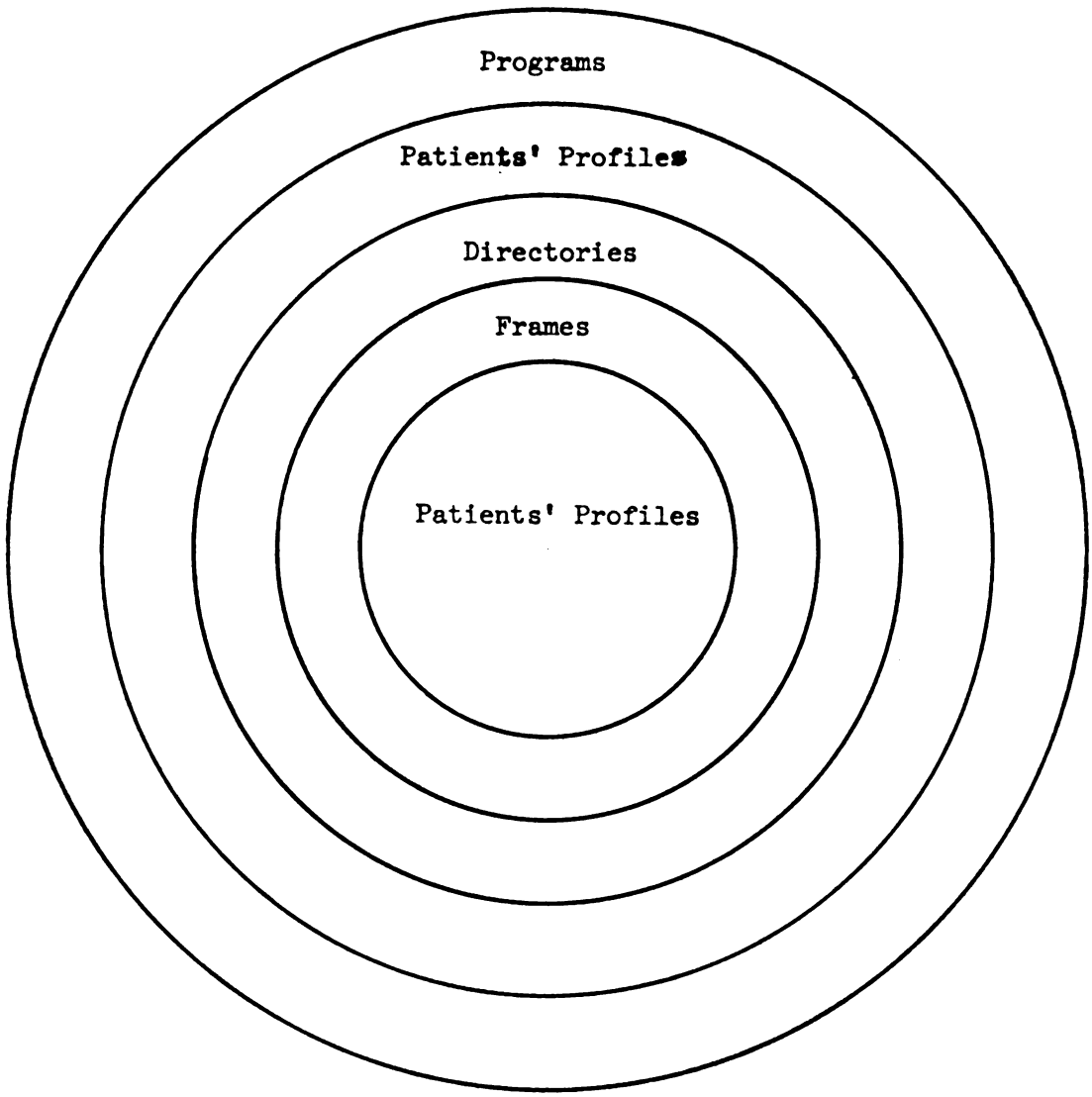


Figure 7.1

The Pharmacy Data Base Organization on the Disk

7.4 BACK-UP AND RECOVERY PROCEDURES

It has been already mentioned that a hardware failure may be located quickly and that the corresponding failing parts can be replaced by new identical spare parts in a short time. However, the most serious breakdown is the destruction of a disc cartridge and it is of the utmost importance to prevent the data contained on the cartridge from being lost. The only dynamic files are the patient's profile and the ward stock order files. This data can only be recovered if there are redundant versions of the data. This redundancy can be introduced by two methods: complete duplication of the dynamic files or maintenance of a log of the modifications to the dynamic files.

The first method requires a double copy of the data which implies three disc drives (two independent drives to support separate copies of the dynamic data and a third drive to use as back-up in case one of the others fails.)

The advantages of this first method is that the normal operations can continue with a minimal interruption (the time to switch to the two remaining drives.)

The inconvenience associated with this method is the overhead necessary to keep a double copy of the data. However, with a small system, this is acceptable because writing of data occurs only when a new order is stored in the file; on the average, there are one or two new orders per patient per day which amounts to approximately 500-1000 updates during the day (in fact, it is less because usually several orders are entered simultaneously and require only one update.)

In addition to the double copy a complete copy of the cartridge is made before every shift and kept on a shelf in case of a major breakdown of the system.

The second back-up method is a constant monitoring of the transactions generated during an eight hour shift. In this case, the third disc drive might be replaced by a tape drive (either a regular tape or a cassette drive.) To be completely secure, the tape drive should be backed-up by a second one, so that the economics of the two methods are just about equivalent. This method has been implemented during the testing period by using a disc cartridge as recording medium.

Since all the transactions are generated at the keyboard (or equivalently by the selection mechanism), a record of all the keystrokes is sufficient to enable the regeneration of the patient profiles.

The data is recorded as a triplet containing the keystroke, the user # and the time between two successive keystrokes.

User #	Time between events	key code

Since the keyboard interrupts are treated sequentially, the log of keystrokes can be considered as a time series:

k_i^u where i is the index corresponding to the keystroke and u is the user index.

Each k_i^u can be associated with a time to event (real-time) t_i or the time between events $\Delta t_i = t_i - t_{i-1}$.

The ti can be found from the Δt_i and t_o by the formula:

$$t_i = t_o + \sum_{n=1}^{i-1} \Delta t_n$$

The time between events can be measured in basic clock cycles (1/10 of a second in our case) and are, therefore, more advantageous in terms of memory space required for storage than the full-time in hours, minutes, and 1/10 second. Furthermore, by changing the unit of the clock cycles

the time series can be played back at an increased speed.

In case of the destruction of the data base, a monitored version of the time series can be played back by generating internally the keystroke interrupts and driving the software as if it was externally activated.

An experimental play back version was implemented with a speed increased by a factor of six (1/60 s), but even if the clock is speeded up, the playback is limited by the rate of disc accesses from the drive (with an average seek time of 70 ms, only 15 accesses per second can be made). Therefore, if the data base is to be generated by direct play back, it must be started with the previous back-up version and it may take several hours to reconstruct the data base depending on the length of the time series. The advantage of this method is a slightly smaller overhead for storing the keystroke series (as opposed to the double copy.) However, the introduction of a new device (magnetic tape or cassette tape) to be supported in real-time introduces more complexity in the software.

The main advantage of the second method is the potential use of the keystroke time series to study the statistical behaviour of the system for research purposes. During a testing phase the availability of monitored time-series would be adventageous since a malfunction of the software could be reproduced by playing back the monitored time series. Thus, the problem which caused the malfunction could be repetitively re-examined.

The disadventages of the method as a back up alternative are the time required to play back the keystroke series to regenerate the data base and the cost and aomplexity as previously mentioned.

Both methods have been implemented and the conclusion reached was that in a production environment, the double copy was preferable because it requires less down time and insures a quick recovery procedure.

A combination of the two methods would be ideal since both a quick recovery and a recording of the time series would be available for research purposes and for debugging.

7.5 THE DIFFERENT FUNCTIONS AND PROCEDURES OF THE PHARMACY APPLICATION

The remaining part of the Chapter describes the major procedures implemented for the pharmacy system.

7.5.1 THE LOG-IN PROCEDURE, THE MAIN INDEX, AND FUNCTION BUTTONS

The user activates the system by a special log-in procedure as follows: he must activate the editor (function button F1) and enter a password which is masked on the screen (see figure below.)

```
ENTER OPERATOR ID:***

■ PROCEED
```

Figure 7.2 "Log-in" Frame

When the password is entered, the "Proceed" selection will trigger a procedure to check the validity of the password. If it is recognized as valid, the next frame will be the "main index" presented below. Together

with the selection items on the screen, the operator has the choice of using one of the special purpose programmed function buttons which are:

F1: activate the editor (bring blinking cursor)

F2: store a typed page in a file

F3: reverse the selection process

F5: advance forward one page in the patient's profile

F6: back-up one page in the patient's profile

F7: review the message already built

F8: back to the selection process after F7

F9: erase all the message area

F10: bring the main index

The main index frame is displayed below:

```
PATIENT PROFILE                 BLOCK TIME FILLING
■ View   Profile                  ■ Dispense-Cred
■ Modify Profile                  ■ Check Dis-Cred
■ Create Profile                  ■ Fill Specials
BLOCK PRINTING
■ Print All Patients' Profiles
■ 8hr Medication Schedule
WARD STOCK ORDERING
■ Ward item from List A-K
■ Ward item from List L-Z         ■ SIGN OFF
```

Figure 7.3 Main Index

When this index is displayed, the frame selection system is in a neutral state; no message is in process, no file is open, and no procedure or process is in execution.

The index is divided into five types of actions:

- access to the patient's profile
- block printing processes: patient's medication profile and the eight hour medication schedule for each nursing shift
- block time filling: crediting and dispensing of unit dose drugs
- ward stock items ordering
- signing off

7.5.2 PATIENT'S PROFILE ACCESS

If a new drug order or a modification of the drug profile is desired, the first step is to display the existing profile ("View profile.")

A new profile is created by filling in the editing frame presented in Figure 5.3 and the use of the function button F_2 will generate the updating of the directories and the storage of the first page of the profile on the disc.

The access to a profile is done via a frame fork to access the alphabetized directory. The alphabet has been broken down into twenty-nine categories presented below:

AAA-ARI	GOL-HAR	PAM-POR
ARJ-BEC	HAS-HOL	POS-RIC
BED-BRA	HOM-JER	RID-SAN
BRB-CAN	JES-KIN	SAO-SHR
CAO-CLA	KIO-LED	SHS-STE
CLB-CRU	LEE-LUM	STF-THO
CRV-DOD	LUN-MCB	THP-VIN
* DOE-EVE	MCC-MIT	VIO-WIL
EVF-FRE	MIU-NEV	WIM-ZZZ
FRF-GOK	NEW-PAL	BY HOSP. NO _

Figure 7.4

Alphabetical Categories to Access Patient's Name

The thirtieth alternative is the access by hospital number which can be entered by typing. The selection of a category will bring the corresponding list of patients on a selectable frame.

Once a name has been selected, the first page of the profile is displayed. An example of the patients' profiles pages is given below:

```
Name (Last,First,Init):DOE JOHN A.          ←
Hosp #:000001  ← Bed:624/1 ← Birthdate:08-06-23←
Sex:M←Race:C←Wt: 72Kg←Adm:06-03-74←PhCh:$      ←
MD:DR SMITH              ← Last Surg:08-65←
Adm Prob:Back pain                              ←
DX:Herniated lumbar disc
                                                ←
Diet:regular← Low Fluid: ←   I/O: ←
Spec Orders:
```

Figure 7.5 First Page of Patient's Profile

Function button F5 displays the following page: it is another free text page for notes and allergies concerning the patient.

```
DOE JOHN A.              000001   624/1

Notes :complete bed rest,traction.
                                                ←
Allergies,Adv Reac (Date,Drug,Type,Severity)
:None known.
```

Figure 7.6 Second Page of Patient's Profile

The modification of these two pages can be done by selecting the alternative "<u>modifying profile</u>" on the main index and retyping the modification when the corresponding page is displayed.

The next pages represent the drug profile of the patient:

```
DOE JOHN A.                 000001   624/1

01 06/06 06/13  ASPIRIN TAB 325mg 2 po qid:
02 06/06 06/13  PHENOBARBITAL TAB 15mg 1 po qid:
03 06/06 06/07  CODEINE TAB 60mg 1 po prn Pain:
```

Figure 7.7 Medication Profile

NOTE: The top of the screen represents a message window from the message area and the first line contains the patient's name, hospital identification number, and location. When the end of the profile is reached the use of function button F5 will bring the drug order index.

7.5.3 THE DRUG ORDER INDEX

The drug order index indicates a state where a patient's profile has been selected and allows for new drug orders to be generated. The drug order index is the root of several selection processes represented in the following frame:

```
DOE JOHN A.                 000001   624/1

NEW DRUG ORDER   * By Common 30   ■ By CUDA List
 Full list &     ■ A-K            ■ Non Listed
   Specials      ■ L-Z            ■ Narcotics
■ IV Infusion/Additive
■ Preoperative Drugs
■ Change Duration/Use
■ Print Patient's Profile
■ Discharge (Close Profile)
                                  ■ Pharmacy Index
                                  ■ SIGN OFF
```

Figure 7.8 Drug Order Index

7.5.3 CREATION OF A NEW DRUG ORDER

A drug order can be represented by the following selection formula (see Chapter 4):

D.O.= □ Drug name [□ DOSE FORM □UNIT PER DOSE(□ ROUTE)]□ FREQUENCY

A frame fork is used to access the list of drug names, a dose form related to the drug name is selected, then a route of administration is chosen and finally a frequency of administration is specified.

In the order generation, several different paths can be taken as the starting point:

- a list of the most common thirty drugs
- a list of unit dose drugs for the Central Unit Dose Area (CUDA)
- a list of narcotic drugs
- the full list of drugs and items available in the pharmacy

The different paths have been created in order to shorten the selection process to make an order:

- for instance, the pharmacy statistics have shown that a large percentage of all drug orders can be taken from a list of 30 names which have been displayed as a separate frame providing rapid access
- the unit dose drugs are a subset of the full list for which every dose is unitized and distributed on a daily basis. This procedure is followed on approximately a third of the wards in the hospital.

If a drug is experimental and not yet in the formulary, it will be entered by selecting the "non-listed" option which will allow for the free text entry of the order. The full list is accessed by two indices (A-L, M-Z.)

The following is an example of the creation of a drug order from the list of the most common 30 drugs:

```
   DOE JOHN A.                 000001    624/1
04 06/17 06/24

■ ACETAMINOPHEN  ■ DIPHENHYDN Na  ■ PENTAZOCN Cl
■ ALLOPURINOL    ■ DOSS           ■ PENTAZOCN Lac
■ AMPICIL TRIHD  ■ Fe SO4 USP     ■ PENTOBARB Na
* ASPIRIN        ■ FLURAZEPAM Cl  ■ PHENOBARBITAL
■ BENADRYL       ■ LOMOTIL        ■ POT Cl
■ BISACODYL      ■ MAALOX         ■ PREDNISONE
■ CHLORAL HYD    ■ METHYLDOPA     ■ PROCLOPER ED
■ DEXAMETHASONE  ■ MOM            ■ PROPOXPHEN Cl
■ DIAZEPAM       ■ MULTIVITS UC   ■ RIOPAN
■ DIGOXIN        ■ PEN PHENOX K   ■ SECOBARB NA
```

Figure 7.9 "Most Common Thirty Drugs" Frame

Between the drug index and the present frame, a procedure has been executed to compute the drug sequence number and the start and stop date. A drug order is valid for seven days unless otherwise specified.

```
   DOE JOHN A.                 000001    624/1
04 06/17 06/24  ASPIRIN

* TAB 325mg      ■ CAP 325mg      ■ TAB 600mg pink
■ CAP 650mg      ■ SUP 65mg       ■ SUP 325mg
■ SUP 650mg
```

Figure 7.10 Dose Forms Available for Aspirin

Only dose forms corresponding to ASPIRIN can be selected.

The next frame is multiple selection frame to select the number of units to be given as a dose:

```
   DOE JOHN A.                000001   624/1
04 06/17 06/24  ASPIRIN TAB 325mg

UNITS PER DOSE  ■ 1            * 2

■ 3             ■ 4            ■ 5

■ 6             ■ 1/2          ■:

ROUTE OF ADMIN  ■ po

                               ■ ng
```

Figure 7.11

The selective mask for the route of administration permits the display of only those routes that are appropriate for the selected dose form. For example, only po (oral) and n_g (nasogastric tube) are displayed for a dose form such as TAB (tablet) as these are the only appropriate routes for this dose form.

The next frame represents the set of standard frequencies at which a dose can be given during the day:

```
   DOE JOHN A.                000001   624/1
04 06/17 06/24  ASPIRIN TAB 325mg 2 po

FREQUENCY OF ADMINISTRATION

■ qd            ■ q1h          ■ ac

■ bid           ■ q2h          ■ pc

■ tid           ■ q3h          ■ with meals

* qid           ■ q4h          ■ hs

■ STAT          ■ q6h          ■ prn

                ■ q8h

                ■ q12h         ■:
```

Figure 7.12

With the extended FPL version it would be possible to eliminate the frequencies for which the cumulative dose may be dangerous by executing a procedure and applying a mask pushed on the stack when the dose and number of units were selected. These safeguards could be established by medical staff regulations, but would contain provision for over ride if the attending physician specifically authorizes it.

The next frame is a procedural frame that allows the user to specify if the drug has been dispensed or to modify the start and stop dates:

```
   DOE JOHN A.                000001   624/1
04 06/17 06/24  ASPIRIN TAB 325mg 2 po qid

CHECK ENTIRE DRUG ORDER

 F7 reviews all orders.F8 returns

Change START and/or STOP Dates

                ■ YES            ■ NO

Was Drug Dispensed

                ■ YES            ■ NO

Do you want a Label?

                ■ YES            ■ NO
```

Figure 7.13 Procedural Frame at end of Drug Order

If the start or stop dates require modification, the following frame will be displayed:

```
DOE JOHN A.                    000001    624/I
04 06/17 06/24  ASPIRIN TAB 325mg 2 po qid

ADVANCE          START DATE     STOP DATE

SET BACK         START DATE     STOP DATE
```

Figure 7.14

```
                NB=

 7               8               9

 4               5               6

 1               2               3

 0               ERASE           PROCEED
```

The frame providing for the modification of start and stop dates is a good example of the concept of the software machine underlying the frame selection approach: the frame appears to be identical to a calculator keyboard (N= is the zone where the digits selected are displayed), and once a number is displayed, it can either add (advance) a chosen number of days to the date or subtract (set back) a chosen number from the dates displayed in the message. It is easy to see from this example, that a frame selection system can simulate any keyboard machine by changing the name and function of the keys.

7.5.5 INTRAVENOUS ORDER

Very few automatic pharmacy ordering system includes the intravenous solution orders [SOU73] because they have a complex structure.

The selection process involved for an intravenous (I.V.) order can be represented by following the selection formula:

□ BOTTLE NB □ FLUID □ VOLUME □ FLOWRATE (□ ADDITIVE □ AMOUNT)*

A bottle number is selected, a fluid and a volume must be specified together with a flow rate and then an unspecified number of additives may be added to the fluid (star operator.)

At the end of the selection process, a label for the I.V. bottle is printed automatically. A sample label print out is given below:

```
DOE JOHN A.
000001          624/1

D5W             1L
100 ml/hr

Kcl             30mEq
Vit B cpl-C     10ml
DISPENSED    06/17
EXPIRES AT   12h01
BOTTLE # 1
```

Figure 7.15 Example of I.V. Label

7.5.6 PREOPERATIVE DRUG ORDER

A separate path is also taken for preoperative drug order. A preoperative drug order is characterized by the following formula:

(□ PREOP DRUG □ STRENGTH)* □ ROUTE □ TIME

A preoperative order may contain several drugs, the frequency is replaced by a time of administration since a preoperative order is given only once. A label is also printed for preoperative orders and a sample is given below:

```
DOE JOHN A.
000001          624/1
Atropine SO4    0.4mg
Meperidine Cl   75mg
IM              6h00am
```

Figure 7.16 Preoperative Drug Order Label

7.5.7 NON-FORMULARY DRUG ORDER

When a new drug not listed in the present formulary is ordered the pharmacist will enter the order manually by using the following editing frame:

```
   DOE JOHN A.                 000001   624/1
04 06/17 06/24

 MANUAL INPUT OF NON LISTED DRUG

Drug Name:                    ←

Dosage form-size :            ←

No of Units per Dose:         ←

Route:        ←

Frequency of administration:                ←

                                  ■ PROCEED
```

Figure 7.17 Frame for Entry of Non Listed Drugs

7.5.8 MODIFICATION OF A DRUG ORDER

A given drug order may not be deleted from the patient's profile - it becomes part of the drug history of the patient. However, a drug order may be discontinued, temporarily stopped, reinstated and the stop date may be extended.

This operation is performed by selecting the drug order to be modified from the page of the profile. Then the modification is selected on a frame specifying the action to be taken. The corresponding frame is given below:

```
   DOE JOHN A.                 000001   624/1
01 06/06 06/13  ASPIRIN TAB 325mg 2 po qid:

■ DISCONTINUE    ■ TEMPORARY STOP

■ CHANGE USAGE   ■ REINSTATE

■ EXTEND STOP DATE by

■ CREDIT(Doses)  ■ DISPENSE(Doses)

                 NB=

■ 7              ■ 8              ■ 9

■ 4              ■ 5              ■ 6

■ 1              ■ 2              ■ 3

■ 0              ■ ERASE          ■ PROCEED
```

Figure 7.18 Modification of a Drug Order

7.5.9 THE BLOCK TIME FILLING PROCEDURES

The block time filling is a procedure which takes place daily and is part of the unit-dose pharmacy delivery system. The unit-dose system employs the concept of a twenty-four hour dosing period. A duplicate set of medication drawers is maintained so that one set contains sufficient doses of drugs for a twenty-four hour period and is in use on the floor, while the second set is being filled in CUDA by a pharmacy assistant. Daily at 7:00 P.M., after the pharmacist has checked the medication drawers, a pharmacy assistant exchanges the depleted drawers for newly-stocked drawers. The contents of the returned drawers are checked and any doses left may be credited to the patient. This work is done by pharmacy technicians, and the crediting and dispensing of drugs is a batch operation done ward by ward. A separate filling procedure takes place for special drug orders involving a longer preparation time and more complex preparations. ("Fill specials")

The block time filling procedure is interesting because it shows an example of a batch process being implemented in real time.

The selection process involved in the crediting and dispensing is described by the following selection formula:

□ WARD (□ CREDIT □ DISPENSE)*

and the following frame is used for the selection of a ward:

6N FCS	6S	60E
7N	7S	7E
8N	8S	8E
9N	9S	9E
10N	10S	10E
11N	11S	11E
14N	12S	14E
15N	14S	15E
15B	15S	UC HOSPITAL
15W	6W	NONE

Figure 7.19 List of Wards in the Hospital

At the completion of the block filling or checking of a ward, the corresponding selection is flagged by a letter F (filled,) S (specials) or C (checked.)

Once a ward is selected, a procedure will automatically look up the bed location directory and display the first patient file. Within each file only the active drug orders are presented to the user in sequence.

The following frame is used for the crediting:

```
   DOE JOHN A.                 000001   624/10
01 06/06 06/13  ASPIRIN TAB 325mg 2 po qid:

U/Day=08          D/Day=04

      Select number of units returned
```

Figure 7.20

	NO=	
7	8	9
4	5	6
1	2	3
0	ERASE	PROCEED

With each block-filled order, a dispense and credit zone is updated and is used as the basis for the billing program. For each order, a procedure computes the number of doses and the number of units to be dispensed.

```
   DOE JOHN A.                 000001   624/10
01 06/06 06/13  ASPIRIN TAB 325mg 2 po qid:

U/Day=08          D/Day=04

    Select number of units added or removed

1 Add units              =08

2 Remove units           =


                NB=8

7                8                9

4                5                6

1                2                3

0                ERASE            PROCEED
```

Figure 7.21 "Dispense" Frame

On this frame the number of units to be added is displayed so that the technician may select the number required from the drug stock. The "proceed" selection will bring the process back to the crediting frame with the next order to be filled. The filling of specials orders follows the same procedure, but only the special orders are displayed. The checking of the dispensing is done after the filling and the pharmacist may modify the number of doses dispensed by entering a new number.

```
   DOE JOHN A.                 000001    624/10
01 09/06 06/13  ASPIRIN TAB 325mg 2 po qid:

U/DAY=08          D/DAY=04

        CHECK LAST DOSE DISPENSED

If incorrect enter new nb of dose to dispense

New dose/day:


                NB=

7                8                9

4                5                6

1                2                3

0                ERASE            PROCEED
```

Figure 7.22 "Checking" Frame

7.6 THE WARD STOCK ORDERING

This type of ordering is a bulk dispensing of pharmacy items to the wards.

The ordering of an item is done from the full list and the orders are stored in a temporary sequential file which is deleted after the billing program has been run.

The selection process for this type of order is simpler than a patient's drug order, because it does not involve the individual patients and there is no need for a route and frequency of administration.

The selection formula is as follows:

□ WARD (□ ENTRY NAME [□ ITEM] □ ITEM NB)

The entry name refers to an entry in the full list (it may be a drug or sundry item) and the item refers to the particular form or size within

that entry. The next selection specifies the number of items dispensed to the ward. This type of ordering is identical to a classical inventory system, therefore the replacement of the pharmacy inventory list by another list (for instance, central supply) will not require any reprogramming.

It should be noted that a real time inventory could be maintained easily if the system was built as a pure inventory system. In the pharmacy application this was not necessary because the billing charges are computed on a daily basis and at the same time the inventory level can be updated.

7.7 THE MEDICATION ADMINISTRATION SCHEDULES

Every eight hours a medication administration schedule is generated for the nursing personnel. This task is executed as a background process and reviews every file; every active order is listed with the corresponding times at which a dose must be given to a patient. The program is table-driven, so that a given frequency may be interpreted differently for a given ward.

This process is executed as a background task and does not interfere with the foreground and middleground processes. An example of a medication schedule is given in the Appendix.

Discussion:

The pharmacy application as described in this Chapter has several features not available in previous pharmacy systems:

i) The generalized frame selection approach allows flexibility in the system design (it was possible to implement a unit dose distribution system as well a classical system.)

ii) The existence of the frame programming language enables the easy

modification of frames and procedures which will allow for future expansions into a system for automatic drug-drug interaction checking.

iii) The use of a new technology adapted to the frame selection approach gives a better response time than a more conventional systems (from less than 1/10s to 1/2s depending on the load and type of operation executed.)

iv) The low cost of the hardware used to support the application can be economically justified.

Chapter 8

Conclusion

"When Earth's last picture is painted and the tubes are twisted and dried,
When the oldest colours have faded and the youngest critic has died
We shall rest, and faith, we shall need it - liedown for an aeon or two
Till the Master of all Good Workmen shall put us to work anew."

R. Kipling

CONCLUSION

The previous experiences of developing medical information systems have shown that it is wise to avoid the use of text typing, and this is made possible by the use of a frame selection system. This is particularly important because of the nature of medical language, which is descriptive and encyclopedic and therefore, must be expressed by linguistic terms.

This dissertation has included an attempt to study such systems in a more formal way by relating them to the theory of information, the theory of automata, and the theory of formal languages. This was done by the definition of a model called the selecton which can be viewed as a finite state machine where inputs are selections displayed on a CRT, and organized as frames. Instead of considering the model as an acceptor of strings, it was used for its generative properties of output strings to build messages by the concatenation and manipulation of phrases and choices selected. From this perspective, the selecton model is more like a formal grammar, and a regular selecton generates regular expressions while a pushdown selecton generates sentences of a context-free grammar.

By introducing the possibility of executing procedures as a result of a selection, the model becomes more powerful, and can in particular allow the introduction of context-sensitive rules and use transformations on the strings already generated.

These results show that a frame selection system can be as powerful and flexible as any other software approach, but instead of parsing or interpreting strings, the frame structure and the procedures are designed to generate syntactically correct strings. This is useful, not only for building application systems where the user may generate orders or observa-

tions by selecting phrases from frames, but it is also useful for generating statements of a programming language.

In particular, the concept of a frame programming language was developed and implemented as a frame selection system. A frame programming language is a high level language which enables the construction of frame selection systems: for each frame and each selection on a frame, a statement can be generated to indicate what will be the result of the selection (output of string, execution of a procedure, storage on a pushdown store, or branching to a new frame). A particular example of a frame programming language was described, and used to build a pharmacy ordering system. A real-time operating system was developed on a minicomputer (Four Phase Model 70) to support frame selection system applications.

Although the system as developed meets the expectations of a rapid response time (less than 1s for 8 CRT's working simultaneously), some improvements are suggested, which could be implemented as an external layer on the current system or by extension of the frame programming language.

First, the procedures invoked as a result of a selection were written in assembly language and it would be helpful if a higher level language were available to write applications procedures, which might be linked by using the frame language approach.

Second, the file system could have an external layer realizing the concepts of data base schema, and a simple data base query language could be implemented by means of the frame selection system.

Third, the implementation of a more comprehensive frame programming language would allow for more complex semantic relations to be specified when other applications are designed. This could be useful in pathological or radiological descriptive statements.

Finally, with respect to the modular approach advocated previously, a communication software could be developed. A simple protocol should be devised to insure the local communication of data and messages between the modules constituting the nodes of the communication network. In particular, the use of identical hardware modules connected to a central communication processor will greatly simplify the task of communicating data between modules representing activities in different hospital department.

In conclusion, other avenues of research are opened: a further analysis of the frame selection approach for the implementation of interactive programming language designed for specific applications. For instance, a simulation language like GPSS (general purpose simulation system) would definitely be easier to use with such an approach, the command and control languages used in time sharing systems will also benefit from such an approach, especially for small systems. It is even possible that a COBOL equivalent language could be implemented as a frame selection system for small applications. It is also likely that a frame selection approach would be a good teaching tool to learn programming languages by enabling the user to build syntactically correct sentences instead of compiling the statement to see if it is correct. A frame selection approach can be used to remove all the syntactical idiosyncrasies of programming languages, and the automatic parsing will considerably reduce the effort to produce a multiple user incremental compiler. Furthermore, the standardization of a language will no longer require the use of specific keywords; only the grammar will be standardized and the frames could be translated in any language.

In the medical information field, frame selection systems are well suited for applications such as history taking, aids to diagnosis, medical

order generating, report generation, inventory systems, ward systems, and computer assisted instruction. Most of the research needed is to discover the logic and the appropriate grammars to express the medical language and practice. This is not a simple task and no unique solution exists, but if progress is to be made, one must start with the simple, well defined problems first and then undertake the more difficult problems later.

The advantages of the frame selection system approach are its extensibility, and the direct and intuitive understanding it offers to medical professionals as well as computer specialists.

References and Bibliography

[AHO72] A.V. AHO and J.D. ULLMAN
"The Theory of Parsing, Translation and Compiling"
Prentice Hall, 1972

[ARB68] M. ARBIB
"Algebraic Theory of Machine Languages and Semigroups"
Academic Press, 1968

[BAC59] J.C. BACKUS
"The syntax and semantics of the proposed international algebraic language"
Proceedings of IFIP Congress, Paris, 1959, pp 125-131

[BAL71] M.J. BALL
"An overview of total medical information systems"
Methods of Information in Medicine, Volume 10, No. 2, 1971

[BAL74] M.J. BALL
"Medical data processing in the United States"
Proceedings of AFIPS National Computer Conference, 1974, Volume 43

[BAR67] G.O. BARNETT and P.A. CASTLEMAN
"A time-sharing computer system for patient care activities"
Computers and Biomedical Research, Volume 1, (March) 1967, pp 41-51

[BAR70] G.O. BARNETT and R.A. GREENES
"High level programming languages"
Proceedings of Conference on Medical Information Systems, San Francisco, 1970

[BAR71] G.O. BARNETT
"The use of computers in clinical data management: the ten commandments"
American Medical Association Conference, Las Vegas, (February) 1971

[BAR74] G.O. BARNETT
"The modular hospital information system"
Chapter 11, Computers and Biomedical Research, Fourth Edition, Stacy and Waxman (Editors), Academic Press, 1974

[BARH64] Y. BAR-HILLEL
"Language and information"
Selected essays on their theory and application
Addison-Wesley, 1964

[BEK72] G.A. BEKEY and M.D. SCHWARTZ
"Hospital information systems"
Biomedical Engineering Monograph,
Marcel Dekker, 1972

[BLO71] M.S. BLOIS and R.R. HENLEY
"Strategies in the planning of hospital information systems"
Proceedings of Journees d'Informatique Medicale,
IRIA, St. Lary, France, 1971

[BRE71] G. BRECHER and H. LOKEN
"The laboratory computer - is it worth its price?"
American Journal of Clinical Pathology, Volume 55, No. 5,
1971

[BREI68] G.Y. BREITBARD and G. WIEDERHOLD
"The ACME compiler"
Information Processing 68, North-Holland Publishing Co.,
Amsterdam

[BRI62] L. BRILLOUIN
"Science and Information Theory"
Second Edition, Academic Press, 1962

[BRU71] S. BRUNJES
"An anamnestic matrix toward a medical language"
Computers and Biomedical Research, Volume 4, 1971,
pp 571-586

[CAR56] R. CARNAP
"Meaning and Necessity"
University of Chicago Press, 1956

[CHE66] C. CHERRY
"On Human Communication"
MIT Press, 1966

[CHO58] N. CHOMSKY and G.A. MILLER
"Finite state languages"
Information and Control, Volume 1, 1958, pp 91-112

[CHO65] N. CHOMSKY
"Aspects of the Theory of Syntax"
MIT Press, Cambridge, Massachusetts, 1965

[COD70] E. CODD
"A relational model for large, shared data banks"
Comm. ACM, Volume 13, No. 6 (June) 1970, pp 377-387

[COF71] E.G. COFFMAN, M.J. ELPHICK, and A. SHOSHANI
"System deadlocks"
Computing Surveys, Volume 3, No. 2, (June) 1971

[COF73] E.G. COFFMAN, Jr., and P. DENNING
"Operating systems theory"
Prentice Hall series in Automatic Computation, 1973

[COH72] S.N. COHEN, M.F. ARMSTRONG, L. CROUSE, and G. HUM
"A computer based system for prospective detection and prevention of drug interaction"
Drug Information Journal, (January/June) 1972

[COL70] M.F. COLLEN
"General requirements for a medical information system"
Computers and Biomedical Research, Volume 3, 1970, pp 393-406

[CRO69] L. CROUSE and G. WIEDERHOLD
"An advanced computer system for real-time medical applications"
Computers and Biomedical Research, Volume 2, No. 6, (December) 1969, pp 582-592

[DAH72] O.J. DAHL, E.W. DIJKSTRA, and C.A.R. HOARE
"Structured Programming"
Academic Press, 1972

[DAV69] L.S. DAVIS, M.F. COLLEN, L. RUBIN, and E.E. VAN BRUNT
"Computer stored medical record"
Computers and Biomedical Research, Volume 1, 1968, pp 452-469

[DAV73] L.S. DAVIS
"A system approach to medical information"
Methods of Information in Medicine, Volume 12, No. 1, 1973

[DIJ68] E.W. DIJKSTRA
"A constructive approach to the problem of program correctness"
BIT 8, 1968, pp 174-186

"The strucutre of THE multiprogramming system"
Comm. ACM, Volume 11, No. 5, (May) 1968

"Cooperating sequential processes"
Programming Languages
edited by F. Genuys
Academic Press

[DIJ71] "Hierarchical ordering of sequential process"
Acta Informatica, Volume 1, 1971, pp 115-138

[DEE60] A.E. DEEB
"Electronic data processing system for a hospital pharmacy"
Amer. Jour. Hosp. Pharm., Volume 17, (December) 1960, pp 745-749

[FEL68] J. FELDMAN and D. GRIES
"Translator writing systems"
Comm. ACM, Volume 11, No. 2, (February) 1968

[FRI72] J. FRIES
"Time oriented patient records and a computer databank"
JAMA, Volume 222, No. 12, (December 18) 1972, pp 1536-1542

[GIN68] A. GINZBURG
"Algebraic theory of automata"
Academic Press, 1968

[GOU68] W.A. GOUVEIA, P.B. HOFMAN, and G.O. BARNETT
"Computers - basic principles and hospital pharmacy implications"
Amer. Jour. Hosp. Pharm., Volume 25, (January) 1968, pp 4-11

[GOU69] W.A. GOUVEIA, C. DIAMANTIS, and G.O. BARNETT
"Computer applications in the hospital medication system"
Amer. Jour. Hosp. Pharm., Volume 26, (March) 1969

[GOU71] W.A. GOUVEIA
"Computer applications in the hospital pharmacy
Hospitals J.A.H.A., Volume 45, (January) 1971

[GOR67] G.A. GORRY
"A system for computer-aided diagnosis"
Project MAC, Technical Report MAC-44, MIT, Cambridge, Massachusetts, 1967

[GPSS] "General purpose simulation system", 360 Users' Manual, IBM Manual (H20-0326)

[GRE69] R.A. GREENES, A.N. PAPPALARDO, C.W. MARBLE, and G.O. BARNETT
"Design and implementation of a clinical data management system"
Computers and Biomedical Research, Volume 2, 1969, pp 469-485

[GRI71] D. GRIES
"Compiler Construction for Digital Computers"
John Wiley & Sons, New York

[HAB72] A.N. HABERMANN
"Synchronization of communicating processes"
Comm. ACM, Volume 15, No. 3, (March) 1972

[HAN70] P. BRINCH HANSEN
"The nucleus of a multiprogramming system"
Comm. ACM, Volume 13, (April) 1970, pp 4

[HAN73] P. BRINCH HANSEN
"Operating System Principles"
Automatic Computation
Prentice Hall, 1973

[HAR65] F. HARARY, R.Z. NORMAN, and D. CARTWRIGHT
"Structural Models: An Introduction to the Theory of Directed Graphs"

[HOP69] J.E. HOPCROFT and J.D. ULLMAN
"Formal Languages and Their Relation to Automata"
Addison Wesley, 1969

[ICDA68] "International Classification of Disease - Adapted"
D.H.E.W., Volume 1, U.S. Public Health Service, 1968

[ISN72] D.W. ISNER
"An inferential processor for interacting with biomedical data using restricted natural language"
Proceedings of AFIPS Spring Joint Computer Conference, 1972

[JOH71] J.L. JOHNSON
"Clinical Laboratory Computer Systems - A Comprehensive Evaluation"
J.L. Johnson Associates, 1971

[JOHN70] L.J. JOHNSTON
"Problems of implementation of unit-dose dispensing"
Amer. Jour. Hosp. Pharm., Volume 27, (October) 1970, pp 815-821

[KNU68] D.E. KNUTH
"The Art of Computer Programming"
Volume 1, Addison Wesley, 1968

[KNI73] J.R. KNIGHT and W.F. CONRAD
"A literature review of computer oriented pharmacy services in hospitals"
Jour. Clin. Computing, Volume 3, No. 3, (November) 1973

[KRO68] K. KROHN, J.R. RHODES, and B.R. TILSON
"The prime decomposition theorem of the algebraic theory of machines"
Algebraic Theory of Machines, edited by Arbib, Academic Press, 1968

[LED69] R.S. LEDLEY
"Practical problems in the use of computers in medical diagnosis"
Proceedings of IEEE, Volume 57, No. 2, (November) 1969

[LIN70] D.A. LINDBERG
"A statewide medical information system"
Proceedings of Conference on Medical Information Systems, January, 1970

[MAR72] R.F. MARONDE and S. SEIBERT
"Electronic data processing of prescriptions in hospital information systems"
Biomedical Engineering Monograph, Marcel Dekker Inc., 1972

[MEL70] J.P. MELROSE
"Automated Medication Order System - Hospitals"
J.A.H.A., Volume 44, (September) 1970, pp 68-73

[MIL72] R.A. MILLER, R.F. DELEON, E.T. HERFINDAL, and J.L. HIRSCHMAN
"The frame system: a second generation unit-dose distribution system"
Hospital Progress, (February) 1972

R.A. MILLER and R.F. DELEON
"Development of a computerized pharmacy control system"
Amer. Jour. Hosp. Pharm., Volume 29, (November) 1972, pp 963-966

[NAU60] P. NAUR et al
"ALGOL 60 Report"
Comm. ACM, (May) 1960

[PRA69] A.W. PRATT and M.G. PACAK
"Automatic processing of medical English"
International Conference on Computational Linguistics, September, 1969, Stockholm

[PIE61] J.R. PIERCE
"Symbols, Signals, and Noise: The Nature and Process of Communication"
Harper & Bros., 1961

[PRI71] C.E. PRICE
"Table look-up techniques"
Computing Surveys, Volume 3, No. 2, (January) 1971

[RHO73] J. RHODES
Class Notes, "Automata Theory" (Course 226 B/C)
Department of Mathematics, University of California, Berkeley, 1973

[SCH71] J.R. SCHULTZ, S.V. CANTRILL, and K.G. MORGAN
"An initial operational problem-oriented medical record system for storage manipulation and retrieval of medical data"
Proceedings of AFIPS Spring Joint Computer Conference, 1971

[SCHW72] M.D. SCHWARTZ
"Status of hospital information systems"
Hospital Information Systems, edited by Bekey and Schwartz, Marcel Dekker, Co., 1972

[SHA49] C.E. SHANNON and W. WEAVER
"The Mathematical Theory of Communication"
University of Illinois Press, 1949

[SHU63] M.P. SCHUTZENBERGER
"On context-free languages and push-down automata"
Information and Control, 1963

[SHU70] S.A. SHUMAN and P. JORAND
"Definition mechanisms in extensible programming language"
Proceedings of AFIPS Fall Joint Computer Conference, 1970

[SIB73] E.H. SIBLEY and R.W. TAYLOR
"A data definition and mapping language"
Comm. ACM, Volume 16, No. 12, (December) 1973

[SIM72] G.I. SIMON, H.M. SILVERMAN, T.G. VETTER, and G. VOLPERT
"A semi-automated approach to unit-dose dispensing"
Amer. Jour. Hosp. Pharm., Volume 29, (June) 1972,
pp 491-495

[SIN70] S.J. SINGER
"Visual display terminals in a hospital information system"
Conference on Medical Information Systems, January 1970,
San Francisco

[SLA66] W.N. SLACK, G.P. HICKS, C.E. REED, and L.J. VAN CURA
"A computer based medical history system"
New England Journal of Medicine, Volume 274, 1966,
pp 196-198

[SNOP65] "Systematized Nomenclature of Pathology"
College of American Pathologists,
American Medical Association, Chicago, 1965

[SOU73] D.E. SOUNDER, W.A. GOUVEIA, D. SHERERTZ, R. ZIELSTORFF,
F.E. JONES and G.O. BARNETT
"A computer-assisted intraveinous admixture system"
Amer. Jour. Hosp. Pharm., Volume 30, (November) 1973,
pp 1015-1020

[VAN70] E.E. VAN BRUNT
"The Kaiser-Permanente medical information system"
Proceedings of Conference on Medical Information Systems,
January 1970, San Francisco

[VAL68] C. VALLBONA et al
"An on-line computer system for a rehabilitation hospital"
Methods of Information in Medicine, Volume 7, No. 1, 1968

[WEE68] L.L. WEED
"Medical records that guide and teach"
New England Journal of Medicine, Volume 278, 1968,
pp 593-600 and 652-657

[WEE69] L.L. WEED
"Medical Records, Medical Education and Patient Care"
Case Western Reserve University Press, Cleveland, 1969

[WIE69] G. WIEDERHOLD
"An advanced computer system for medical research"
Proceedings of the IBM Japan Computer Science Symposium, Research and Development, and Computer Systems, Tokyo, November 1969, pp B1-B15

[WIE70] G. WIEDERHOLD, R. FREY, and S. GIRARDI
"A filing system for medical research"
Proceedings of Journees d'Informatique Medicale, IRIA, St. Lary, France, 1970

[WIE72] G. WIEDERHOLD
"A choice of language to support medical research"
ACM Conference, Boston, 1972

[WIR66] N. WIRTH,and C.A.R. HOARE
"A contribution to the development of ALGOL"
Comm. ACM, Volume 9, No. 6, (June) 1966

[WIR69] N. WIRTH
"A basic course on compiler principles"
BIT 9, 1969, pp 362-386

[WIN72] B.H. WINTERS and L. HERNANDEZ
"A computerized drug inventory control system"
Amer. Jour. Hosp. Pharm., Volume 29, (September) 1972, pp 780-785

[ZAD65] L.A. ZADEH
"Fuzzy sets"
Information and Control, 1965, pp 338-353

[ZAD71] L.A. ZADEH
"Quantitative fuzzy semantics"
Information Sciences 3, 1971, pp 159-176

"Similarity relations and fuzzy ordering"
Information Sciences 3, 1971, pp 177-200

[ZAD73] L.A. ZADEH
"Outline of a new approach to the analysis of complex system and decision processes"
IEEE Transactions on Systems, Man and Cybernetics Volume SMC 3, No. 1, (January) 1973

[ZEL73] D.D. ZELLERS
"A computer based medication system utilizing unit-dose distribution and medication administration"
Jour Clin. Computing, Volume 3, No. 3, (November) 1973

Appendices

APPENDIX 1

Syntax Description of the Frame Programming Language

The following is a description of the frame programming language using the Backus-Naur Form [BAC59,NAU60]. Since the language was implemented as a frame selection system, the following description is not meant to be used by an automatic parser:

```
<FRAME>:= <EDITING FRAME>|<SELECTION FRAME>|<MIXED FRAME>

<EDITING FRAME>:= <EDITING HEADER><EDITING FRAME BODY>

<EDITING HEADER>:= EDITING FRAME<FRAME NAME>

<SELECTION FRAME>:= <SELECTION HEADER><SELECTION FRAME BODY>

<SELECTION HEADER>:= SELECTION FRAME<FRAME NAME>

<MIXED FRAME>:= <MIXED HEADER><MIXED FRAME BODY>

<MIXED HEADER>:= MIXED FRAME<FRAME NAME>

<FRAME NAME>:= <CHARACTER>*

<EDITING FRAME BODY>:= <DATA ENTRY ITEM>*

<DATA ENTRY ITEM>:= <PROMPT>:<EDITING FIELD>

<SELECTION FRAME BODY>:= <SELECTION ITEM>*

<MIXED FRAME BODY>:= (<SELECTION ITEM>)*<DATA ENTRY ITEM>

<SELECTION ITEM>:= <SELECTION CHOICE><SELECTION INSTRUCTION>

<PROMPT>:= <STRING>

<EDITING FIELD>:= <BLANK>*

<SELECTION CHOICE>:= <SELECTION POINT><CHOICE STRING>

<CHOICE STRING>:= <STRING>

<SELECTION INSTRUCTION>:= <STRING INSTRUCTION><PROCEDURE INSTRUCTION>
                           <ROUTE INSTRUCTION><SEMANTIC RESTRICTION>

<STRING INSTRUCTION>:= CHOICE '<CHOICE STRING>' <CHOICE OUTPUT>

<CHOICE OUTPUT>:= OUTPUT {<EMPTY STRING>|<CHOICE STRING> AND <OUTPUT FORMAT>}
```

```
<OUTPUT FORMAT>:= CONCATENATION|CARRIAGE-RETURN LINEFEED|IDENTATION|
                  TABULATION
<PROCEDURE INSTRUCTION>:= EXECUTE <PROCEDURE IDENTIFIER>
                          <PROCEDURE CHARACTERISTICS><PROCEDURE BODY>
<PROCEDURE IDENTIFIER>:= <CHARACTER>*
<PROCEDURE CHARACTERISTICS>:= <LOCATION STATUS><PRIORITY><PARAMETER LIST>
<LOCATION STATUS>:= RESIDENT <BOOLEAN>|OVERLAY<BOOLEAN>
<PRIORITY>:= FOREGROUND|MIDDLEGROUND|BACKGROUND
<PARAMETER LIST>:= <VARIABLE IDENTIFIER>*
<VARIABLE IDENTIFIER>:= <CHARACTER>*
<PROCEDURE BODY>:= <STATEMENT>|<PROCEDURE BODY><STATEMENT>
<STATEMENT>:= <LABEL><assembly language instruction><OPERAND>
<OPERAND>:= <VARIABLE OPERAND>|<REGISTER OPERAND>
<VARIABLE OPERAND>:= <VARIABLE IDENTIFIER>|<VARIABLE IDENTIFIER>
                     <INDEX REGISTER>
<REGISTER OPERAND>:= <REGISTER>|<REGISTER><REGISTER>
<LABEL>:= <CHARACTER>*
<ROUTE INSTRUCTION>:= <EMPTY ROUTE>|BRING '<FRAME NAME>' AND
                      <STACK INSTRUCTION>
<STACK INSTRUCTION>:= <POP><EMPTY ROUTE|FRAME NAME>|<PUSH><EMPTY ROUTE|
                       FRAME NAME>|<EMPTY STACK INSTRUCTION>
<EMPTY STACK INSTRUCTION>:= NO STACK OPERATION
<EMPTY ROUTE>:= NO ROUTE
<POP>:= POP <BOOLEAN>
<PUSH>:= PUSH <BOOLEAN>
<SEMANTIC RESTRICTION>:= <EMPTY RESTRICTION>|<CHOICE SEMANTIC RELATION>*
<CHOICE SEMANTIC RELATION>:= <CHOICE STRING><SEMANTIC RELATION>
```

<EMPTY RESTRICTION>:= NO MASK

<SEMANTIC RELATION>:= <BOOLEAN>

<BOOLEAN>:= TRUE|FALSE

<STRING>:= <CHARACTER>|<CHARACTER><punctuation sign><CHARACTER>

<CHARACTER>:= <letter>|<digit>|<blank>

<BLANK>:= <blank>

<SELECTION POINT>:= □

<REGISTER>:= R0|R1|RA|RB|X1|X2|X3

<INDEX REGISTER>:= X1|X2|X3

<letter>:= A|B|C|D...|X|Y|Z|a|b|c|...|x|y|z

<blank>:=

<digit>:= 0|1|2|3|4|5|6|7|8|9

<punctuation sign>:= :|,|'|"|`|.|(|)|;

<assembly language instruction>:= <FOUR PHASE instruction set> (1)

(1) See FOUR PHASE User's Manual

APPENDIX 2

OPERATING SYSTEM PRIMITIVES

The primitives used in the operating system are written in assembly language. In the following, they are described by the calling sequence and the set of parameters necessary for the primitives to operate.

The mnemonic "BAL" refers to a branch and link instruction which keeps the program counter in a register so that it can be used to access re-entrant procedures.

The mnemonic "DCN" is a data definition pseudo-instruction.

The mnemonic "PZE" is used for pointers.

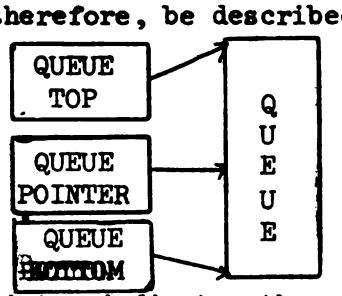
The mnemonic "BSS" is used to reserve a memory location.

The mnemonic "BRA" refers to a branch instruction.

1. Primitives QUEUE and DEQUEUE

Since the Four Phase computer[1] has instructions to push and pop elements from a stack and instructions to move blocks of words, the queues are implemented by using the stack mechanism but instead of removing the element on the top of the stack, the element on the bottom of the stack is removed first. This is acceptable because the maximum size of the queue is equal to the maximum number of users which will never exceed 16 on this machine.

A queue can, therefore, be described by three pointers as follows:



The current pointer indicates the position where a new item can be inserted in the queue.

[1] For more details, see Four Phase Systems, System IV/70 Reference Manual, Document No. SIV/70-11-1C

The primitive queue is implemented by:

```
BAL    QUEUE
PZE    QUEUE.POINTER
DCN    QUEUE CONTENT
BRA    QUEUE FULL
       NORMAL RETURN
```

The primitive DEQUEUE is implemented by:

```
BAL    DEQUEUE
PZE    QUEUE.TOP
BSS    1  RETURN CONTENT OF QUEUE
BRA    QUEUE EMPTY
       NORMAL RETURN
```

2. Primitive STAPRO

This primitive is used to put a process on the queue of ready processes from any hardware level to any priority level (foreground, middleground, background). It has the following format:

```
           BAL    STAPRO
           PZE    POINTER TO DESCRIPTOR
           BRA    ERROR RETURN
                  NORMAL RETURN

DESCRIPTOR DCN    LEVEL OF CALL
           DCN    LEVEL OF EXECUTION  (foreground,
           PZE    QUEUE.POINTER        middleground,
           DCN    PROCEDURE ADDRESS    background)
```

3. Primitives INRSEM and DECSEM

These primitives are used for incrementing and decrementing a semaphore variable and have the following format: if the variable SEMAPHORE is zero after the decrement instruction, skip the next instruction.

```
DEC    SEMAPHORE
BRA    QUEUE.PROCESS  (SEMAPHORE.QUEUE)

CRITICAL SECTION

INR    SEMAPHORE
NOP
BAL    DEQUEUE  (SEMAPHORE.QUEUE)
```

4. Input/Output Primitives

4.1 Disk Primitive

The disk I/O can be requested from the three primary levels (foreground, middleground and background).

```
                BAL     $DISCF    (foreground)
                        $DISCM    (middleground)
                        $DISCB    (background)
                PZE     DISK REQUEST
                BRA     ERROR RETURN
                        NORMAL RETURN

DISK REQUEST    BSS     1  STATUS WORD
                DCN     DRIVE #
                DCN     MEMORY ADDRESS
                DCN     Bit 0 = READ/WRITE + SECTOR COUNT +
                                DISK ADDRESS
```

4.2 Keyboard Primitive

When a message is requested from the keyboard, the following primitive is used:

```
                BAL     KEYIN
                PZE     KEY.REQUEST
                        NORMAL RETURN

KEY.REQUEST     DCN     KEYBOARD #
                DCN     RELATIVE STARTING ADDRESS ON SCREEN
                DCN     ADDRESS OF BUFFER
                DCN     MAXIMUM LENGTH OF MESSAGE
```

4.3 Printer Primitive

The printer driver includes the medium speed line printer (500 lines per minute) as well as the low speed label printers (60 characters per second). The primitives are identical for all the devices so that it is possible to use temporarily any of the low speed printers in case the others fail.

```
                BAL     LINEPT
                PZE     OUTPUT REQUEST
                        NORMAL RETURN

OUTPUT REQUEST  DCN     LINE LENGTH + INCREMENT + DEVICE NUMBER
                DCN     BUFFER
```

The device number refers to the printer device normally used. If it is not available, it can be replaced by another device number. The variable BUFFER refers to the starting address of the message to be printed. The line length divides this buffer into lines and the increment refers to the gap between lines (if a screen image is to be printed). The end of buffer is characterized by a zero marker.

5. File Handling Primitives

5.1 Allocation of Disk Space

Two primitives can be used which are either the allocation of one sector (256 words) or the allocation of two contiguous sectors:

```
BAL    ALLOCI  (or ALLOC2)
DCN    DRIVE #
BSS    1 SECTOR ADDRESS
BRA    DISK FULL
       NORMAL RETURN
```

Before the execution of the primitive, the sector address can be filled by a disk address at which the look-up for a sector should start in order to avoid unnecessary look-ups on already allocated zones on the disk. At the return of the allocation routine, this word contains the actual sector allocated.

The deallocation of disk space is done by:

```
BAL    DELOCI
DCN    DRIVE #
DCN    DISK ADDRESS
BRA    ERROR
       NORMAL RETURN
```

5.2 Directory Look-up

A general look-up subroutine is available for look-up of entries in a table:

```
        BAL    LOOKTB
        DCN    END OF TABLE CHARACTER + ENTRY LENGTH +
        PZE    TABLE ADDRESS                INCREMENT
        PZE    ADDRESS OF STRING TO COMPARE
        PZE    POINTER TO RESULT OF LOOK-UP
               NORMAL RETURN

RESULT  BSS    1  POINTER TO MATCHING STRING
        BSS    1  NUMBER OF NON-MATCHING CHARACTER
                  (0 if complete match)
```

A standard directory (frames, hospital number, bed location) can be examined by using the following primitive:

```
        BAL    DRFND 1
        DCN    DRIVE #
        PZE    POINTER TO ENTRY NAME
        PZE    POINTER TO ENTRY VALUE
        BRA    ERROR
               NORMAL RETURN
```

The entry name refers to a six character name and the value contains the returned disk address of the file being looked-up.

A new entry can be created or deleted by using the following primitive:

```
        BAL    CREATI  (DELETI)
        DCN    DRIVE #
        PZE    PNAME
        PZE    PVALUE
        BRA    ERROR RETURN  (NO ROOM OR NON-EXISTANT)
               NORMAL RETURN
```

5.3 File Page Descriptors

When a new page is created for a patient's profile, the following primitive is used to create a page descriptor:

```
                    BAL    SETPAG
                    PZE    PAGE DESCRIPTOR

PAGE DESCRIPTOR     DCN    PAGE ID + FRAME FORMAT ADDRESS
                    DCN    ITEM INDEX LENGTH AND RETATIVE ADDRESS
                    DCN    DATA STARTING ADDRESS
                    DCN    PAGE DESCRIPTOR ADDRESS WITHIN SECTOR
```

The look-up of a page within a file is done by using:

```
                    BAL    LOKPAG
                    PZE    PAGE DESCRIPTOR
```

5.4 Data Item Handling within a Page

Within a page, an item can be inserted or retrieved by an item index which can be accessed via the following primitive:

```
                    BAL    ITEM
                    PZE    ITEM DESCRIPTOR

ITEM DESCRIPTOR     DCN    ITEM # + READ/WRITE BOOLEAN
                    DCN    ADDRESS OF ITEM INDEX
                    DCN    ADDRESS OF ITEM BUFFER
                    DCN    LENGTH OF ITEM
```

In the case of a write operation, this primitive updates the item index and stores the data in the file. In the case of a read operation, the primitive searches the item index and moves the item in the buffer area specified.

DAY SHIFT 6N 06/07/74 * MEDICATION SCHEDULE *

624/1 DOE JOHN A. 000001

ASPIRIN	TAB 325mg	2	po	qid	09:00		13:00	
PHENOBARBITAL	TAB 15mg	1	po	qid	09:00		13:00	
CODEINE	TAB 60mg	1	po	prn				

624/2 DOE JANE M. 000002

METHYLDOPA	TAB 250mg	1	po	q12h	09:00			
PERCODAN	TAB	1	po	q3h	09:00	12:00		15:00
MEPIRIDINE Cl	INJ 75mg/ml	1	IM	q3h	09:00	12:00		15:00
QUINIDINE SO4	TAB 300mg	1	po	q8h			14:00	
TEDRAL	TAB	1	po	q4h	09:00		13:00	
FLURAZEPAM Cl	CAP 30mg	1	po	hs				
DIGOXIN	TAB 0.25mg	1	po	qd				

624/3 JONES JOHN R. 000003

TYLENOL	TAB 325mg	2	po	q4h	09:00		13:00	
Compazine	Inj 10mg/2ml	1	IM	q4h	09:00		13:00	
MAALOX	SUS	1	po	q4h	09:00		13:00	
DECADRON	TAB 1.5mg	3	po	qid	09:00		13:00	
CHLORAL HYD	CAP 500mg	1	po	hs				
MOM	SUS	1	po	hs				

624/4 JONES MARY J. 000004

TYLENOL	TAB 325mg	2	po	q4h	09:00		13:00	
SYNTHROID	TAB 0.1mg	1	po	qd				
DALMANE	CAP 30mg	1	po	hs				
MOM	SUS	1	po	prn				
DOSS	CAP 240mg	1	po	prn				