A RELATION ALGEBRA MODEL FOR DATA BASES

by

S.A. Bukhari

Research Report CS-75-11

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

April 1975

A RELATION ALGEBRA MODEL FOR DATA BASES

by

S.A. Bukhari

A Thesis

present to the University of Waterloo

in partial fulfillment of the requirements

for the Degree

DOCTOR OF PHILOSOPHY

Department of Computer Science

Faculty of Mathematics

University of Waterloo

April 1975

ACKNOWLEDGEMENTS

I gratefully acknowledge my debt to Dr. J. G. Linders whose constant encouragement and guidance made this work possible. I am deeply grateful to my wife, Attya, who in her own way contributed greatly towards the completion of this work.

DECLARATION AND AUTHORIZATION

SIGNATURE [illegible]

ABSTRACT

This thesis attempts to show the feasibility of modeling a data base in terms of the mathematical system known as relation algebra. Such a model uses a set of axioms to derive an automaton and its state transition diagram for the data base. The resulting diagram is a structural representation of the way in which the relations in the data base compose with one another subject to the constraints specified in the axioms. These constraints include, among others, assertions about the structural properties of the relations which are in the data base, (e.g. symmetry, equivalence). By stating these assertions as axioms, the associated properties of relations will remain invariant throughout the operation of the data base. In this way a measure of control and integrity of the data base is achieved.

The transition diagrams can be used to simplify query expressions. By controlling the transition from one state to another in the diagram, the problem of data security can be addressed. Furthermore, by requiring that any update procedure produce only those updated versions of data which continue to satisfy the basic set of axioms it follows that the state transition diagram also remains applicable after each valid update. This contributes to the reliability and consistency of the data base model.

TABLE OF CONTENTS

CHAPTER I INTRODUCTION

Intuitively a data base is a collection of data associated with facilities which provide the capability of performing certain operations on the data and of establishing interrelationships among them. The data base is therefore endowed with a modelling capability, and within its framework it becomes possible to organize the data so as to establish interrelationships which at least partially represent the universe from which the data are drawn. A data base so conceived provides a means of further study and to investigate the properties of the underlying universe with which the data base is associated.

Over the last decade the emphasis in computing has clearly shifted to a better understanding of the nature, representation, storage and manipulation of data. In the transition from manual operation to the use of computers the trend has been to an emulation of manual techniques. However, a point has now been reached where it is necessary to reexamine the foundations of all aspects of data processing and data analysis. As part of this reexamination there has emerged a parallel interest in data bases and the associated requirements of data security, data base integrity and correctness of operations, to challenge the existing technology.

The development in data bases in some ways is analogous to the metamorphosis undergone in programming languages, where the evolution was clearly from primitive unstructured machine language programming to very high level structured languages. In this case however the motivation was somewhat different. The need was a need for formalism, correctness and conceptual clarity. The process of development was from the concrete to the abstract with a striving for a high level of generalization.

Data should have embedded in them relations corresponding to the relations embedded in the universe from which they are drawn. These relations are not arbitrary or ad hoc relations among the parts but represent independent structural entities and as such must be so specified and explicitly represented in any structural description of a data base. If this is not done then structural information about the universe may be lost entirely or may be only implicit in the data. The result would be the loss of control over the integrity and modelling capability of the data base. It is desirable that the structure of a data base be invariant with respect to changes occurring in its underlying aggregate data as a result of update, delete and edit operations. This invariance can be achieved only if the structure is specified exclusively in terms of structural properties of the relations embedded in the universe or data from which they come.

This thesis is concerned with an investigation of the use of mathematical systems for modelling data bases. In particular an attempt is made to incorporate certain well known mathematical properties (e.g. symmetry, operators, functions, rules such as composition) into the structure of the data base and to demonstrate that this would facilitate achieving data base reliability, security, integrity, correctness and structural invariance relative to data operations. The approach has been to develop a model for a data base using algebraic properties of relation algebra. Whereas it is desirable to develop a working system of this model, this has not been the principal object. Instead this is an investigation into new ideas for developing data bases which should in time find practical realization perhaps in terms of other technologies.

CHAPTER II DATA BASE DEVELOPMENT

2.1.0. Data Collections

To collect and process data are very common activities in any field of endeavor. Of late these activities have grown tremendously both in scope and size [Westin, 1972], and computers are being used increasingly in this context.

The rapid growth of data collections has brought to the forefront many problems of data and their relation to the computer. Recently much attention is being devoted to the study of these problems and the need for their systematic solution is being recognized.

2.2.0. Data Directed Processing

In what may be called data directed processing of data, each datum carries with it information about the operation to be performed on it. This method reminiscent of plug board programming days of computing [Brooks and Iverson, 1969] is quite convenient for small data collections on which a few diverse operations have to be performed. The datum and its associated operation are presented to the computing mechanism, which performs the operation and proceeds to the next datum.

For larger collections of data unless the patterns and redundancies existing in data or operations are exploited to their full advantage, data directed processing becomes inefficient and cumbersome.

Furthermore, since data collections vary and each processing task is different, it appears that for processing fixed data with different operations or different data with identical operations both data and operations have to be prepared for processing repeatedly. This naturally leads to a shift in the emphasis from the study of data and their patterns to finding ways and means of separating data from their operations, so that either could be altered separately.

On the other hand if the data and operations are readily available together, it is unnecessary to consider access and ordering of data.

2.3.0. Separation of Data and Operations

As the need for handling very large collections of data grew, a gradual and almost complete separation of operations and operands took place in the form of stored program and data. Here ideally instructions are in one module called program and the data in a separate module. As the instructions are executed operations are performed on the data which are accessed and retrieved from the data module as required. This mode of data processing may be termed program directed. This method while solving some of the problems of efficiency inherent in data directed processing, has created its own peculiar problems. Among these, perhaps the most important, is the dependence of programs on the data on which they act.

2.3.1. Data Dependency

Data dependencies in a program arise when it uses those properties of data which result directly from ordering, indexing or methods of accessing them.

Problems due to data dependency in programs to some extent arise from the fact that in practice programs once written are not subject to continual structural change and in this sense are relatively fixed or invariant. On the other hand, collections of data are time varying, subject to changes of organization, deletion, editing, and updating. This mismatch of invariability in time results in valid programs becoming invalid due to changes in data. If an invalid program acts and causes changes in data, then the result is a corrupted data collection.

2.3.2. Data Dependency: Solutions

One solution is to ensure that dependencies exploited in a program remain intact for data as they are changed. However, this may not be always possible to do, specially when there are many programs which may be utilizing various data dependencies.

Another solution is to write programs which utilize structural characteristics of data collections rather than specific special properties of individual datum, some of which result from ordering and indexing. Hence as long as the structural characteristics of data collections are invariant there will be no need to change the programs which utilize these characteristics. For example if a collection of ordered pairs F is a function and a program P utilizes only the fact that F is a function, then as long as the functional character of F is preserved, P will remain applicable to ordered pairs in F, independently of their number, which will depend upon insertions or deletions over time.

2.4.0. Data Base

The term data base is applied to collections of data whose structure is specified. This is in contrast to the view that a data base is a large filing system [Knuth 1973, p. 389]. A data base is more than a filing system and it must provide its users adequate tools to model various aspects of reality as they perceive them. Furthermore, this facility must be provided in a data independent manner, i.e. free from the considerations of how the data is ordered, indexed, accessed, or stored.

The need for such data independence is clear. Data bases are collections of data which are time varying. If the data base is to remain structurally invariant in time, i.e. if its integrity is guaranteed then the integrity should not depend upon individual datum changes, such as insertions or deletions.

2.5.0. Notation

For specifying the structure of a system a notation is needed. Symbolic notation standing alone does not provide any indication of the meaning and usage to be attached to it in a particular application. Thus if M is a relation designed to model a functional relationship in some practical situation, then M standing alone does not give any indication that it is a function. In most applications meanings assigned to symbolic notation remain fixed. Conventionally to avoid repetition this meaning is asserted once, say by means of a declaration, and can be factored out. These statements and their representations are essential in order to make symbolic notational systems operationally and structurally complete.

The supporting assertions usually concern the structure of the notation and remain constant. This invariance is induced by the fixed and constant aspects of reality being modelled.

Notational devices which may be mentioned are Turski's notation [Turski, 1971] for modelling data structures, and the languages provided by sets and relation calculus. Notable also is the work of Galler and Perlis [1970]. Notation from logic, functional calculus [Levien and Maron, 1967] and the domain of algebra [Nero, 1969] have all found some use in data base work.

2.6.1. Information Algebras

In a CODASYL Development committee report [1962] an outline of an information algebra was proposed. This is an attempt to recognize the fact that a mathematical space provides a better model for data collections than that provided by an amorphous set.

Information algebra is based on the undefined notions of property, value and entity; there are three postulates regarding the existence of a value set from which an entity may be assigned a single value only and entity value pairs are always well defined. The concepts of lines, areas (files), and bundles characterize the operations of the algebra. Finally the glumps are provided as summarizing functions.

Recently interest has been revived in this type of approach, [Kobayashi, 1972].

2.6.2. File as a Formal System

Hsiao and Harary [1970] have proposed a formal structure for files. Starting with a set A of attributes and V of values, a record R is defined as a subset of A x V. The concepts of index, keywords, address, pointers, and pointer-lists are formalized. Finally a file is defined by closure of a set of records on pointer-lists, and the notion of a directory of a file is established.

A generalized file structure is a file and its directory. The authors have shown that commonly encountered filing schemes such as inverted, indexed, sequential, and multilist schemes are all special cases of the generalized file structure of their paper.

This paper establishes that it is possible to construct a formal model of a rather general nature and forms a basis of further explorations on algebraic lines [Wong and Chiang, 1971].

2.6.3. Files as Boolean Algebras.

Wong and Chiang [1971] starting with the formal basis provided by Hsiao and Harary [1970] have shown how to organize files as Boolean algebras and have indicated that there might be some merit in such explorations because they provide alternate means of storing basic data from which other information can be derived. For example when files are organized as finite Boolean algebras then theoretically it is sufficient to store only their atoms.

2.6.4. LEAP: An Associative Language

Feldman and Rovner [1969] have developed an extension of ALGOL60 by appending to it a special construct for handling associative data structures . As association is defined to be an ordered triple (a, o, v) and provides a model of " attribute a of object O has value v". By introducing the notions of items, association and associative context a framework is provided for associative data processing.

2.6.5. Relational Model of Data.

In a number of papers [1970, 1972a, 1972b] Codd has suggested that a data base be viewed as a collection of time varying relations of assorted degrees. The term relation is used in the usual sense that it is a subset of the cross product of domains on which it is defined. Thus if there are n domains $S_1, S_2, \ldots S_n$ then a relation R of order n is a set of n-tuples such that $R \subsetneq S_1 x S_2 x \ldots x S_n$.

Codd uses an array representation for relations. In terms of this representation it may be argued that each n-tuple is a small record and a relation is a file. Files have been viewed as relations by Langfors [1967]. Thus in conventional terms Codd's model of a data base is at least a collection of files containing different types of record. This latter part makes the relational model different from many existing data bases wherein very limited types of record formats are allowed.

Relations are sets and hence have a very rich algebraic and logical structure [Tarski, 1941]. Powerful operations can be performed on them and there exist both calculi and algebras for handling relations systematically [Carnap, 1958]. The systematization afforded by these tools allows Codd to develop a powerful language of specification, description and analysis.

Formalization of data bases in terms of relations naturally leads Codd to consider the question of form and organization of relation. He suggests a number of normal forms for organizing data bases. These normal forms exploit structural characteristics of data and thus serve as a means of control for the entire data base system. This feature of Codd's work must be emphasized, that is structure of data is exploited in a meaningful way.

2.6.7. Data Independent Accessing Model

Senko et. al. [1973] have presented another view of data and data bases. The basic theme of their model is a systematic use of names, set and subset construction for the purpose of describing real objects of all kinds. The various levels of indirection provided by the use of names allow them to handle associations and relation without the specific use of pointers or other data dependent devices.

Entities are distinct realities either concrete or conceptual. Entity sets may be constructed from entities by grouping similar entities. Entities can be named from which name sets may be constructed. A name can be an entity name, entity name set name, entity set name or role name. A set can be an entity set, entity name set, or a subset of an entity set. Using these constructs and devices it is possible to model objects as entities. Various associations between objects can be modelled by considering the role played by the association, and assigning it a suitable role name.

Senko et. al. [1973] use triplets in constructing description sets for entities. Description sets describe entities and their association with other entities. These associations can be exploited for establishing others by matching names in various positions of the triplets. In the model there is no constraint of order, i.e., how things are to be arranged or any use of pointers. Since associations are implicit via names, the model is relatively free from data dependencies and is rather powerful.

2.6.8. Set Model of Data

In a number of articles Childs [1968] has proposed to construct a data structure to model sets and relations. Such a data structure could then serve as a useful tool where notions of sets and relations can be applied. Childs [1968], introduces the notion of a complex in order to overcome the ambiguity inherent in the interpretation of products of more than two sets. For example for sets S_1, S_2, and S_3, the products $S_1 x (S_2 x S_3)$, $(S_1 x S_2) x S_3$ and $S_1 x S_2 x S_3$ are different.

Childs' complex of two sets can be used to model relations of various degree and also allow full use of set theoretic operations.

It is clear that the data base development has taken a turn towards formal and systematic development. Increasingly it is being pointed out that such an approach could be very rewarding [M. W. Wilkes, 1972], [Florentin, 1974].

CHAPTER III ALGEBRAIC PRELIMINARIES

In algebras operations play a fundamental role, in fact as soon as a set is endowed with an operation it qualifies as an algebraic system. A binary operation on a set S is a function defined on its cartesian product $S \times S$ into S. A unary operation is a function on S into S. Generally an n-ary operation is a function on the n-fold cartesian product $S \times S \times \ldots \times S$ (n factors) into S, and the integer n is called the arity of the operation. For completeness a nullary operation has arity 0, and in a set S on which it is defined the effect of this operation is to always select a fixed or constant element of the set. It thus provides a method of specifying the constants of a system. Finally when the arity is finite the operation is called finitary.

For data bases relation algebra is useful. A relation algebra is a Boolean algebra enriched by two additional operations [Chin and Tarski, 1950] and is formally defined elsewhere. It is assumed here that basic concepts of set theory and Boolean algebra are known [Halmos 1967] so many of the results in the following are stated without proof.

3.1.0 Definition. Algebraic System

An algebraic system, or an algebra, is an ordered pair $\langle A, \mathcal{F} \rangle$ where A is a set called the carrier set and $\mathcal{F}$ is a family of finitary operations such that the set A is closed under every operation in $\mathcal{F}$.

For convenience an algebra will also be denoted as $\langle A, O_1, O_2, \ldots\rangle$ where $O_i \varepsilon \mathcal{F}$, $i = 1, 2, \ldots$. Furthermore in some cases nullary operations may not be listed at all.

3.1.1 Algebraic System Development

An abstract algebraic system $\langle A, \mathcal{F}\rangle$ defined as above is too general to make it relevant to concrete situations such as data base work. It is necessary to suitably limit the scope of operations in $\mathcal{F}$. An algebra so restricted provides a powerful tool for modelling selected aspects of the real world, and its further development consists of establishing the necessary conclusion from the restrictions imposed on its operations.

Clearly the constraining hypotheses must be so formulated that the properties and the structure of the concrete world are faithfully reflected in the structure of the algebraic system. Thus for example in a concrete group, an identity is a unique element possessing very special properties. This feature is incorporated in the abstract group algebra by including a nullary operation in its family of operations. A set of axioms are chosen which are obeyed by this operation and guarantee that the effect of this operation is to always select that element of the carrier set which corresponds to the abstract concept of identity in a group. Due to its abstract nature a group algebra describes many concrete groups. It is this descriptive power of algebras which makes the study of algebras so desirable and important. This last point cannot be overemphasized, for it provides a firm logical basis for using algebras for studying concrete objects such as data bases.

3.1.2 Descriptive Power of Algebras

In algebras the concepts of operation, closure and homomorphisms play an important role in establishing similarity of structure. Similarity between two algebraic systems S_1 and S_2 established by means of a homomorphism is valuable for many reasons, and not the least is that if results are known for one of the systems say S_1, then these can be translated into corresponding results for the other system S_2. This combined with the observation [Weyl, 1949] that it is characteristic of algebraic systems that their structure can be described in a few axioms, provides a powerful mechanism for describing various systems.

A semigroup is a closed system with an associative operation defined on it. It is well known [Ginzburg, 1968] that from a set F of transformations a semigroup can be generated. Since transformations are commonly encountered in real life, it appears that semigroup structure should apply to many of these situations, and many of the results already known for semigroups can be applied to real life situations.

3.2.0 Boolean Algebras

A Boolean algebra will be denoted by $\langle B,+,-,1,0\rangle$ or briefly $\langle B,+,-\rangle$. The theory of these algebras is well known [Halmos, 1967] and many familiar results used here will be assumed without proof. A few pertinent facts are noted below.

3.2.1 Characterization as a Lattice

A Boolean algebra is a complemented distributive lattice [Gratzer, 1971] .

This characterization of a Boolean algebra as a lattice is sometimes taken as its definition, and the partial ordering relation will be denoted by $\leq$.

Lattices have been used by Hayes [1968] to construct a model for a thesaurus and to study its properties by decomposing it into the direct product of sublattices.

3.2.2 Algebra of Subsets

Let B be the set of all subsets of a given set A. An algebra can be constructed as follows. First distinguish two special elements of B, the empty set ϕ and the whole set A, and denote these by 0 and 1 respectively. Next, for any subset $x \varepsilon B$ take $\bar{x}$ to be the relative complement A-x. Finally, for any subsets x and y belonging to

B, take $x+y$ to be the set theoretic union and $x \cdot y$ to be the set theoretic intersection. Then $\langle B, +, \cdot, -, 1, 0\rangle$ is an algebra and is called the algebra of all subsets of A.

3.2.3 Theorem

Any algebra of subsets is a Boolean algebra.

Thus algebras of subsets provide a rich source of Boolean algebras.

A number of definitions of some concepts associated with two place relations follow.

3.3.0 Definition. Binary Relations

A binary relation is a set of ordered pairs.

The motivation for considering such sets is that many actual two place relations among various objects can be represented as sets of ordered pairs. Thus the set of ordered pairs

$$\{(1,2), (2,3), \ldots\}$$

may be used to represent

'is the successor of'

relation for natural numbers. Another example is the set

{(kitten, cat), (chicken, hen), (calf, cow)}

and each pair (x,y) in it is connected by the idea

'x is a young y'.

For convenience the terms 'binary relation' and 'relation' will be used interchangeably, unless otherwise stated.

3.3.1 Definition. Domain

The domain of a relation R, denoted by $D_0(R)$ is the set

$$D_0(R) = \{x | (x,y) \in R\}$$

The domain of a relation is the set of first coordinates of the ordered pairs which belong to it.

3.3.2 Definition. Range

The range of a relation R, denoted by $R_a(R)$ is the set

$$R_a(R) = \{x | (x,y) \in R\}$$

Again, the range of a relation is the set of second coordinates of the ordered pairs in it.

Sometimes the terminology first domain and second domain is used respectively for domain and range of a binary relation.

3.3.3 Definition. Field

The field of a relation R, denoted by $F_d(R)$ is the set

$$F_d(R) = D_0(R) \cup R_a(R)$$

From these definitions, it is clear that every relation R satisfies

$$R \subseteq F_d(R) \times F_d(R)$$

Sometimes the term concourse is used synonymously with field.

Several relations may have the same given field G. In G there are two special relations 1' and 0' called the identity relation and the diversity relation respectively.

3.3.4 Definition. 1' and 0'

The identity relation 1' on a given field G is

$$1' = \{(x,x) \mid x \varepsilon G\}$$

The diversity relation 0' on a given field G is

$$0' = \{(x,y) \mid x,y \varepsilon G \text{ and } x \neq y\}$$

An important point to note is 1' and 0' are unique relations with respect to a given field. Further it is also clear that

$$0' = \overline{1'}$$

where '_' the relative complement is taken relative to G×G, where G is the field.

Binary relations are sets of ordered pairs, hence set theoretic operations of union, intersection, and complementation can be defined for a collection of relations. Furthermore other operations, called relative operations, [Chin and Tarski, 1950] may also be defined on the class of relations. The relative operations are peculiar to relations and derive their strongest motivation from them. Two relative operations are defined below. Several others can be defined in terms of these two relative and the usual set theoretic operations.

3.3.5 Definition. Relative Product

The relative product of two relations R and T, denoted by R;T, is the relation

$$R;T = \{(x,y) \mid \text{for some } g,\ (x,g)\varepsilon R \text{ and } (g,y)\varepsilon T\}$$

The symbol ; (i.e. the relative product operation) is sometimes also referred to as the composition operation.

3.3.6 Definition. Conversion

The converse of a relation R, denoted by $\breve{R}$, is the relation

$$\breve{R} = \{(x,y) \mid \text{for each } (y,x)\varepsilon R\}$$

The symbol $\breve{}$ denotes the relative operation of conversion, sometimes also called the reverse operation.

3.3.7 Properties of Relations

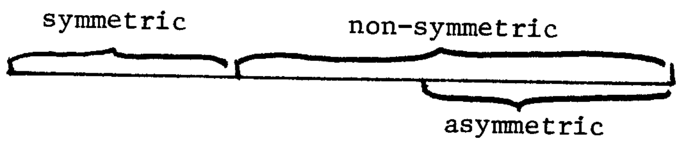
In this section relations are classified [Carnap, 1958] into various categories and certain of their properties enumerated. The identity relation on a field F will be denoted by 1', i.e. if $x\varepsilon F$ then $(x,x)\varepsilon 1'$.

(i) Symmetry. A relation R is called symmetric if $R = \breve{R}$.

(ii) Non symmetry. A relation R is called non symmetric if $R \neq \breve{R}$ i.e. there is at least one pair $(x,y) \varepsilon R$ and $(x,y) \notin \breve{R}$.

(iii) Asymmetry. A relation R is called asymmetric if $\breve{R} \subseteq \bar{R}$ i.e. there is no pair $(x,y) \varepsilon R$ and $(x,y) \varepsilon \breve{R}$.

(iv) These three relations, symmetric, non-symmetric and asymmetric classify all relations as follows.



Some examples may be noted. Sibling is a symmetric relation. Brother is neither symmetric nor asymmetric. Father is asymmetric.

(v) Transitivity. A relation R is called transitive if $R^2 \subseteq R$, where R^2 denotes $R;R$.

(vi) Non transitivity. A relation R is called non transitive if $R^2 \not\subseteq R$ i.e. a relation is non transitive if the condition for transitivity fails for it.

(vii) Intransitivity. A relation R is called intransitive if $R^2 \subseteq \bar{R}$ i.e. a relation is intransitive if R^2 and R are disjoint.

(viii) A three way classification of relations is induced by (v), (vi) and (vii). For example father is intransitive, friend is neither transitive nor intransitive and ancestor is a transitive relation.

(ix) Reflexivity. A relation R is reflexive if for each $(x,y) \varepsilon R$, $(x,x) \varepsilon R$.

(x) Non reflexivity. A relation R is non reflexive is there is at least one $(x,y) \varepsilon R$ such that $(x,x) \notin R$.

(xi) Irreflexivity. A relation R is irreflexive if there is no $(x,y) \varepsilon R$ such that $(x,x) \varepsilon R$.

(xii) Total reflexivity. A relation R is totally reflexive if $1' \subseteq R$.

(xiii) A three way classification of all relations is induced by (ix), (x) and (xi). Some examples may be noted. Father, sister, brother are irreflexive. The subset relation is reflexive.

(xiv) Connectivity. A relation R is called connected if for any two members in the domain of R either R or $\breve{R}$ holds.

(xv) Equivalence. A relation R is said to be an equivalence relation if it is reflexive, symmetric and transitive.

(xvi) Right Ideal Relation. [Chin, Tarski, 1950] A relation R is a right ideal relation if, $x,y \varepsilon F$ and $(x,y) \varepsilon R$ iff x is in the domain of R.

(xv) Closed Relation. A relation R is said to be closed if $R ; \breve{R} ; R = R$ [Ono, 1957]

(xvi) Function. A relation R is called a function if $\breve{R} ; R \subseteq 1'$ [Ono, 1957]

Asymmetrical relations play an important role in the logic of relations. The concept of asymmetry is also important in connection with ordering relations such as total order and partial orders. For a set to be algebraically interesting it must be at least partially ordered.

3.3.7 Arrow Diagrams, Relation Matrix, Similarity and Structure.

A binary relation R may be represented by an arrow diagram according to the following rules. If $(x,y) \varepsilon R$ then draw two circles and label them x and y, and draw an arrow emanating from the x-circle to y-circle thus $\overset{x}{\circ} \longrightarrow \overset{y}{\circ}$.

Thus if $R = \{(1,2), (2,4), (1,4)\}$ then its arrow diagram is

$$\begin{tikzcd}
& 1 \arrow[dl] \arrow[dr] & \\
2 \arrow[rr] & & 4
\end{tikzcd}$$

A relation R may also be represented by its relation matrix by constructing a square array with its rows and columns labelled by the elements of the field of the relation in some arbitrary manner. Then the array contains a 1 in x-row and y-column if $(x,y) \varepsilon R$ otherwise the entry is 0. Thus the relation matrix of $\{(1,2), (2,4), (1,4)\}$ is

$$\begin{array}{c|ccc} & 1 & 2 & 4 \\ \hline 1 & 0 & 1 & 1 \\ 2 & 0 & 0 & 1 \\ 4 & 0 & 0 & 0 \end{array}$$

Two relations R and S are similar if they have isomorphic arrow diagrams or relation matrices. Similar relations have exactly the same structure. [Carnap, 1958]

3.3.8 Some Further Properties of Relations.

Below some more properties of relations are listed. These are elementary theorems which can be easily proved. [Chin and Tarski, 1950] Proofs in general are omitted.

(i) If R is any equivalence relation then

(a) $R \; ; \; R = R$

(b) $\breve{R} = R$

(c) R is closed.

(ii) If R is any function then

(a) $\breve{R} \; ; \; R \subseteq 1'$

(b) R is closed

(c) $\breve{R}$ is closed.

(iii) If R and S are closed relations then

(a) $\breve{R}$ is closed

(b) $R \cap S$ is closed

(c) It is always possible to express R as $R = \breve{F} \; ; \; G$ where F and G are functions.

(iv) Some closed relations are listed below:

(a) If R is a function, R is closed

(b) If R is an equivalence relation, R is closed

(c) If F and G are any pair of functions then $\breve{F};G$ is closed

(d) If R is a right ideal relation then R is closed.

(e) For any function R , $R ; \breve{R}$ is closed i.e.

$$R ; \breve{R} ; R ; \breve{R} ; R ; \breve{R} = R ; \breve{R} .$$

In fact for any function R,

$$R ; \breve{R} ; R ; \breve{R} = R ; \breve{R} .$$

(v) Let $R^* = \bigcup_{n=1}^{\infty} R ; (\breve{R} ; R)^n$. Then

(a) R^* is closed

(b) $R^{**} = R^*$

(c) R is closed iff $R = R^*$.

(vi) A relation R is called dense if $(x,z)\varepsilon R$ implies the existence of an element $y\varepsilon F$ with $(x,y), (y,z)\varepsilon R$.

(vii) If $R = R ; R$ then R is

(a) transitive

(b) dense.

Every equivalence relation is transitive and dense but not conversely.

(viii) R is called left modular relation for $\cap$ under ; iff both

(a) $R ; R \subseteq R$

(b) $R ; \breve{R} \subseteq R$ hold.

Left modular relations are closely related to equivalence relations. Any equivalence relation is left modular for $\cap$ under ; . The converse is not true, e.g., consider the relation [Chin and Tarski, 1958]

$$\{(1,1), (1,2), (1,3), (3,1), (3,2), (3,3)\}$$

This relation is not an equivalence relation but is left modular for $\cap$ under ; . These left modular relations form a more comprehensive class than the class of equivalence relations.

(ix) If R is left modular relation for $\cap$ under ; then R is closed. Because we note that

(1) for any R , $R \subseteq R ; \breve{R} ; R$

(2) for R left modular for $\cap$ under ; $R ; \breve{R} \subseteq R$ and $R ; R \subseteq R$, hence $R ; \breve{R} ; R \subseteq R ; R \subseteq R$

Hence the result.

(x) If R is left modular relation for $\cap$ under ; and S and T are any other relations then

$$R \cap [S ; (R \cap T)] = (R \cap S) ; (R \cap T)$$

From relations we can obtain algebras of subsets. Hence collections of relations provide a source of concrete models of Boolean algebras. On the other hand Boolean algebras must be suitably extended by incorporating relative operations, if they are to adequately reflect the structure of the class of relations. A reason is that if a relation b belongs to the carrier set B of a Boolean algebra, then there is no way to obtain $\breve{b}$ using only b and Boolean operations, even though $\breve{b}$ may belong to B. This argument becomes much more transparent if we assume b to be an atom. Then $\breve{b}$ is also an atom, and certainly $\breve{b}$ cannot be expressed in terms of b using only Boolean operations.

Abstract Boolean algebras may be extended by adding new operations and axioms. If the relational operations composition, reverse, identity and diversity are models for these new operations, then a collection of relations can be used as a model for the extended abstract Boolean algebra. Abstract algebraic systems called relation algebras are such systems. These extended Boolean algebras are very complex and rich in their algebraic structure. This by itself provides the motivation for constructing and studying relation algebras.

3.4.0 Definition. Relation Algebra. [Chin and Tarski, 1950]

A relation algebra is an abstract algebraic system $<S, +, -, ;, \smile>$. The binary operations + and ; are called absolute addition and relative product respectively. The unary operations __ and $\smile$ are respectively called complementation and conversion. The carrier set S is closed under these operations, which obey the following postulates. For $a, b, c \in S$,

(i) $B = <S, +, ->$ is a Boolean algebra

(ii) $a;(b;c) = (a;b);c$

(iii) $(a+b);c = a;c+b;c$

(iv) there exists an element u in S such that $a;u = a$

(v) $\breve{\breve{a}} = a$

(vi) $(a+b)^{\smile} = \breve{a} + \breve{b}$

(vii) $(a;b)^{\smile} = \breve{b} ; \breve{a}$

(viii) $\breve{a} ; (a;b)^{-} + b^{-} = \bar{b}$

As an example of relation algebra, consider a set X. The set of all subsets of the cartesian product is the carrier set. For $+$ take set theoretic union , for $-$ take set complementation relative to $X \times X$. For $;$ take composition of relations , for $\smile$ take conversion of relations. The resulting system is a relation algebra and can be easily verified to be so. For example axiom (viii) may be verified for this system as follows. It is sufficient to prove that $\breve{a} ; (a;b)^{-}$ is a subset of $\bar{b}$. This is done below.

(1) Let $(x,y) \in \breve{a} ; (a;b)^{-}$ assuming that $\breve{a} ; (a;b)^{-}$ is not empty because then there is nothing to prove. Hence

(2) There exists an element $z \in X$, such that $(x,z) \in \breve{a}$ and $(z,y) \in (a;b)^{-}$. Hence

(3) $(z,x) \in a$.

(4) There exist elements of the form $(x,m) \in b$ and $m \neq y$ because if $m = y$ then $(x,y) \notin b$ and $(z,y) \in a;b$ contradicting $(z,y) \in (a;b)^{-}$ established in (2). Hence $(x,y) \in \bar{b}$.

Any collection of binary relations satisfying the requirements of a relation algebra is called a proper relation algebra. It is known [Henkin et al., 1971] that not all relation algebras can be represented by proper relation algebras.

Some results from relation algebra are listed below for reference. It is hoped that these results, which have not been used in this thesis, are indicative of the type of theorems which can be proved in this algebraic system and may in the future find application in the data base work, particularly in the composition of relations.

3.4.1 Relation Algebra. Some Results.

Let $R = \langle A, +, -, ;, \smile \rangle$ be a relation algebra, and $a, b, c, \ldots \varepsilon A$. Following notational convention will be used used. [Chin and Tarski, 1950]

(i) Parenthesis may be omitted if two additions are to be performed in the same order in which symbols follow from left to right.

(ii) Parenthesis may be omitted if two multiplications are to be performed in the same order in which symbols follow from left to right.

(iii) A multiplication symbol is regarded as taking precedence over an addition symbol.

For example $a ; b . c$ means $(a;b).c$

and $a + b ; c$ means $a+(b;c)$

(iv) The partial ordering relation of the Boolean algebra B is denoted by $\leq$.

Some results in relation algebra are listed below without proofs which may be found in [Chin and Tarski, 1950].

(i) The element u satisfying 3.4.0(iv) is the unique element, called the identity and is denoted by $1'$. For any element a,

$$a; 1' = 1';a = a$$

(ii) The element $0'$ is called the diversity element and is given by

$$0' = (1')^{-}$$

(iii) (a) $(1')^{\smile} = 1'$

(b) $\breve{0} = 0$ (0 is the Boolean zero)

(c) $\breve{1} = 1$ (1 is the Boolean unit)

(d) $1;1 = 1$

(iv) $a \leq b$ iff $\breve{a} \leq \breve{b}$

(v) $(a.b)^{\smile} = \breve{a} . \breve{b}$

(vi) $\breve{\bar{a}} = \bar{\breve{a}}$

(vii) If $a \leq b$, then $c;a \leq c;b$

(viii) If $a \leq b$, then $a;c \leq b;c$

(ix) If $a \leq b$, then $\breve{a} \leq \breve{b}$

(x) $(b;a)^{-} ; \breve{a} + \bar{b} = \bar{b}$

(xi) For any $a,b,c,\varepsilon A$, the following formulas are equivalent.

(1) $a;b.c = 0$

(2) $\breve{a};c.b = 0$

(3) $c;\breve{b}.a = 0$

(xii) $a;0 = 0$

(xiii) $a \leq a;1$

(xiv) $a;b.c \leq a;(\breve{a}; c.b)$

(xv) $a;b.c \leq a;\breve{a};c$

(xvi) $a \leq a;\breve{a};a$

(xvii) $a.b.c \leq a;\breve{b};c$

(xviii) For every $a\varepsilon A$,

(1) $\breve{a};\bar{a} \leq 0'$

(2) $\breve{a};(a;1)^{-} = 0$

(xix) $\bar{a} . \breve{a} \leq 0'$

(xx) If $a;c \leq a$ and $\bar{a};c \leq \bar{a}$ then $a.(b;c) = (a.b);c$

(xxi) If $a;c \leq a$ and $a;\breve{c} \leq a$ then $a.(b;c) = (a.b);c$.

CHAPTER IV A FORMAL MODEL OF DATA BASES

The general concept of a data base as a set of data on which relations are defined can be used to obtain a number of algebraic systems associated with it.

4.1.0. Subsets of Data

The properties of individual objects represented in a data base can be modeled by partitioning the data into subsets whose elements are identical in a given respect. Each property induces a decomposition of data into homogeneous subsets. For example the property 'color' would partition the data collection into specific color classes say 'red', 'green', or 'blue' if these were the three colors handled by the data base.

4.1.1. Boolean Algebra

The properties of objects represented in the data base determine a class of subsets. These subsets can be used to generate a Boolean set algebra.

The language provided by this Boolean algebra is then adequate to handle problems of classification and specification of represented objects relative to the class of properties handled by the data base.

4.1.2. Relation Algebra

The associations, interrelationships and interconnections which exist among data can in many cases be modeled by binary relations. The relational structure of a data base specifies how the data are interconnected to one another and what type (symmetrical, transitive, functional, etc.) of binary relations characterize these interconnections.

The domains and ranges of these relations are drawn from the collection of subsets which classify the data into various property classes. These binary relations can be used to generate a relation algebra.

4.1.3. Explicit Structure

It is characteristic of data bases that they have a set of profusely interconnected data. Clearly there is a strong need to formally control these interconnections. In this context the algebraically specified structure of the community of relations in the data base can play an important role. If such information is available then it can form the basis of consistent data base operations. Without such regulatory control it becomes quite easy to execute inconsistent updates or deletions.

4.2.0. A Formal Model of a Data Base

The definitions and discussion in the following sections lead up to a formal definition of a data base.

4.2.1. Data Universe

Definition. The aggregate of data representing the individual objects of a concrete universe, in a data base is called the data universe (DU).

Semantics. A DU is a set of abstract entities which provide representations of concrete situations. In a data base DU is the basic underlying collection of data on which it is built.

4.2.2 Data Base Domain

Definition. A data base domain (DBD) is a subset of the data universe.

Semantics. A DBD is abstractly defined by enumerating the properties of the objects which may belong to it. Concrete DBD's are obtained by searching the data universe D for objects which satisfy the DBD's membership requirements.

For example consider a collection H of data about human beings as DU. Then a DBD could be

$$M = \{x \mid x \in H \text{ and } x \text{ is a male}\}$$

modelling the set of males and also the property of being 'male'. Similarily

$$F = \{x \mid x \in H \text{ and } x \text{ is a female}\}$$

models the set of females in H. M and F will be a realistic description if also $M \cap F = \phi$.

4.2.3. Data Base Relation

Definition. A subset of D x D where D is a data universe is a data base relation (DBR) if and only if both its domain and range are subsets of data base domains.

Semantics . A DBR is a set of ordered pairs and provides an abstract model for two place relations which may exist among objects represented in a data base.

For example the relation "is the father of" among the individuals in a set of human being H can be modeled as

$$Fa = \{(x,y) \mid x \in H \text{ and } y \in H \text{ and } x \text{ is the father of } y\}$$

It is important to note that if Fa is to represent the relation " is the father of" as it is understood in every-day life, then Fa is subject some further restrictions such as Fa is asymmetrical or if $(x,y) \in Fa$ then x must be a male and so on. These properties when explicitly noted in a data base where Fa appears serve to provide regulatory mechanisms for it.

4.2.4. Data Base Axiom

Definition. A data base axiom is an assertion which remains true at all times with respect to a data universe.

Semantics. Data base axioms provide a mechanism for finer modeling of entities modeled. The aggregate of DU and the axioms provide a closer description of the modeled object than that provided by DU alone.

4.2.5. Data Base

Definition. A data base is a quadruple $\langle D, B, \hat{R}, A \rangle$ where

(i) D is a data universe.

(ii) B is a collection of data base domains.

(iii) $\hat{R}$ is a collection of data base relations.

(iv) A is a set of data base axioms.

4.3.0. Data Base Boolean Algebra.

Definition. For a data base $\langle D, B, \hat{R}, A \rangle$, the data base Boolean algebra (DBBA) is the Boolean algebra generated by B.

4.3.1. Data Base Relation Algebra.

Definition. For a data base $\langle D, B, \hat{R}, A \rangle$, the data base relation algebra (DBRA) is the relation algebra generated by $\hat{R}$.

4.4.0. Data Base Model: Relevance

If the proposed model of data base is to be useful, it must provide solutions to basic data base problems. These problems include at least the following:

(i) language (query and manipulation)

(ii) integrity

(iii) security

(iv) control

(v) consistency

(vi) data independence

(vii) structural invariance

4.4.1. Data Base Languages.

The DBBA and DBRA provide useful languages for specifying precisely and unambiguously the various requirements of a data base. Furthermore these algebras also connect data bases with the algebraic structure theory.

4.4.2. Data Base Integrity.

The most important requirement on a data base is that of integrity. It is essential that in a data base there be a guarantee that the basic data, the DBDs and the DBRs shall remain syntactically and semantically consistent and intact throughout the existence of a data base and yet allow operations such as update, access, or system failure.

The syntactic structure of the domains and relations of a data base can be guaranteed and regulated by means of the data base axioms. This may be done by constructing axioms which specify the structure of various entities such as relations. On the other hand semantic structure is very difficult to guarantee mainly because different users may understand the same data differently. Once again, however, some measure of regulation may be achieved via the axioms.

4.4.3. Data Base Control.

In a data base the relative or association structure is of great importance. Merely a collection of data may not be as useful as when various associations and interconnections among data are also specified. And merely specifying these interconnections is not sufficient and means must be provided to maintain the given interconnection and allow new interconnections to be introduced in a systematic and precise way. It should be possible to create new links or delete existing links or prohibit access to certain associations. And all this must be done within the framework of the data base without doing any violence to its structure.

It may be mentioned here that the notions of accessibility and creation of new links or associations among data must be kept separate. Merely because two items can be associated in a data base does not also mean that the associated pair is automatically accessible. For example consider:

$d = \{(\text{account } \#, \text{ salary})\}$

$a = \{(\text{name}, \text{ account } \#)\}$

$b = \{(\text{name}, \text{ salary})\}$

Now b may be obtained as $b = a;d$. However even though d and a may be accessible separately, it may not be desirable for b to be accessible to every user. Separation of these two notions opens up many control situations. Thus system may prohibit explicit construction of inaccessible items but may allow access to these items in coded, implicit, or indirect form. This clearly has relevance to implementation of security in a data base. The logical principle may be stated as follows: object access is logically independent from object construction.

In the following chapter the problem of data base control from the point of view of finite state systems is studied.

CHAPTER V DATA BASE FINITE STATE SYSTEMS

5.1.0 Introduction

The problems of control, security and integrity in a data base have been previously mentioned. Many important aspects of these problems find a most clear statement in the setting of finite state systems.

The theory of finite automata and its relation to algebraic structures and their decompositions is well known [Ginzburg, 1968].

5.1.1 Data Base Semigroup and Monoid

Let $<D, B, \hat{R}, A>$ be a data base. The relational structure of the data base is contained in $\hat{R}$ and D. Let G be given by

$$G = \{x \mid x = y \text{ or } x = \breve{y} \text{ where } y \in \hat{R}\} \quad .$$

That is G is a collection of relations which are in $\hat{R}$ or their reverse relations are in $\hat{R}$. G under composition operation ; generates a semigroup S which may be extended to a monoid M [Ginzburg, 1968]. If the relations in G are finite then so are S and M. Now data base semigroup and data base monoid are defined as follows.

5.1.2 Definition. Data Base Semigroup.

A data base semigroup (DBS) is the semigroup generated by the elements in G and in which all the axioms in A are satisfied.

5.1.3 Definition. Data Base Monoid.

A data base monoid (DBM) is the DBS to which an identity element has been appended.

Semantics. The significance of S (or M) is that it is the free semigroup (or free monoid) which can be obtained from G. On the other hand DBS (or DBM) may be thought of as being derived from S (or M) by identifying certain elements according to the dictions in A, hence it reflects the composition structure of relations in G (hence in $\hat{R}$) subject to the axioms in A. Accordingly, DBS is S in which requirements specified in A are always satisfied. In concrete and specific instances this distinction may not be apparent, due to the fact that any object inherently carries with it all information that structures it. It is only at the abstract level where objects are referred to by names that the distinction becomes clear and significant. For example

$$\{(1,a),\ (2,b),\ (3,c)\}$$

is a concrete instance of a function. If this is denoted by F then it is necessary to impose the restriction on F that it is a function, otherwise essential structural information is lost. On the other hand given only the concrete entity

$$\{(1,a),\ (2,b),\ (3,c)\}$$

a test can be carried and it can be determined that it is a function. In this sense it carries its own structural information.

In the following section only finite DBS or DBM are considered.

5.1.4 Data Base Semiautomata.

Let $<D, B, \hat{R}, A>$ be a data base and G defined as before. Let G* be the transitive closure of G under the operation of concatenation, i.e. G* is the set of words obtained by concatenating elements of G in all possible ways. The empty word $\lambda \varepsilon G^*$. Let $w = w_1 w_2 \ldots w_m$ and $g = g_1 g_2 \ldots g_n$ be two words in G*, and $w_i, g_j \varepsilon G$ and $i=1,2 \ldots m$; $j=1,2,\ldots n$. The equality of two words w and g is defined as follows.

Definition. Two words w and g in G* are equal and denoted by w=g if and only if $w_1; w_2; w_3; \ldots ; w_m = g_1; g_2; g_3; \ldots ; g_n$ where $w_i, w_j \varepsilon G$ and $i=1,2,\ldots m$; $j=1,2,\ldots n$.

It is then easy to establish the following theorem.

Theorem:

(1) Equality of words in G* as defined above is an

equivalence relation and

(2) It has a finite index.

Let the equivalence classes of G^* mod $=$ be represented by a word of minimal length in each equivalence class. A list of such representatives is finite. With each member in this list associate a unique member $s \varepsilon S$ where S has exactly the same number of elements as in the list of representa tive of equivalence classes. Furthermore the empty word is in the list and hence has a corresponding representation in S.

A finite state system $<S, N, \Lambda>$ over alphabet G can be constructed as follows.

(1) Take G as the alphabet

(2) Take S as the set of states

(3) Take N as state transition function, where $N : G \times S \rightarrow S$ and $N(g,s) = s'$ where $sg = s'$.
$\Lambda \varepsilon S$ is initial state.

Definition. The finite state system constructed as above is called the data base semiautomaton (DBSA).

It may be noted that S may be interpreted as a set of unique names associated with the distinct words of minimal length in G^*.

5.1.5 Properties of DBSA

(1) DBSA is unique

(2) DBSA exists

(3) Number of states is bounded

(4) DBSA is a generator of DBM

(5) DBSA is reduced

(6) DBSA is connected.

5.1.6 Data Base Automata

An automaton can be obtained from data base semi-automaton by attaching a suitable output function. Such an automaton is called data base automaton (DBA).

The output function of DBA can be used to control the system in a variety of ways.

The DBSA and DBA can play an important role in the control and regulation of data bases, especially as far as the relational aspects are concerned. Furthermore if this last remark can be supported, then there is the interesting possibility of realizing such control through hardware realization of DBA as a sequential circuit. In fact the direction to take would that be of microprogramming.

There are certain practical problems which must be mentioned. If $\hat{R}$ contains many relations then the number of states in DBA will explode. This number evidently must be controlled. This control furthermore must be achieved in a data independent way, i.e. in controlling the number of states one must not appeal to specific and particular data but to other features and characteristics of the data base such as the structure of relations in $\hat{R}$. By structure of a relation the reference is to those properties of the relation which remain invariant in time. Thus the fact that a relation is a function is a structural fact and remains so whether it contains one pair or a thousand pairs.

Some examples of data base semiautomata follow.

For convenience the following conventions are adopted. The members of $\hat{R}$ will be denoted by upper case letters. The state corresponding to $R \varepsilon \hat{R}$ (hence $R \varepsilon G$) will be denoted by $r \varepsilon S$, or by some other straightforward mapping or coding as the situation demands.

The members of A will be denoted a_i, $i=1,2,\ldots$,(or simply by i) and usually a_i will stand for a true statement.

5.2.0 DBSA: Single Equivalence Relation.

In the data base $<D, B, \hat{R}, A>$, let

(i) $\hat{R} = \{R\}$

(ii) $a \varepsilon A$ where a stands for the statement: R is an equivalence relation.

(iii) $G = \{R, \breve{R}\}$. Then:

(iv) The distinct words of G^* are computed as follows.

λ

$\mathbb{R}$

$\breve{R} = R$ because R is an equivalence relation

$RR = R$ because $R;R = R$ for equivalence relations.

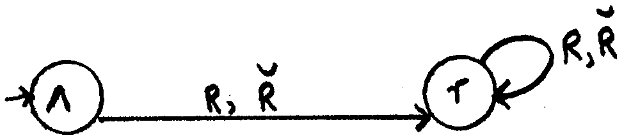
Hence λ and R are the two distinct words.

(v) The set of states $S = \{\Lambda, r\}$

(vi) The state transition table is

	R	$\breve{R}$
Λ	r	r
r	r	r

(vii) The state transition diagram is



In the following presentations of semiautomata, much of repetitive computational details will be omitted, where possible.

5.2.1 DBSA: Single Symmetric Function

A data base $<D, B, \hat{R}, A>$ with $\hat{R} = \{R\}$ where $a \varepsilon A$ asserts that R is a symmetric function. This means $R = \breve{R}$. Thus $G = \{R\}$ and distinct words of G^* are computed as follows.

λ

R

$\breve{R} = R$

RR

RRR = R, because R is a symmetric function.

The set of states $S = \{\Lambda, r, r^2\}$

The transition diagram is shown below.

$$\begin{tikzcd} {} \arrow[r] & \Lambda \arrow[r, "R"'] & r \arrow[r, shift left, "R"] & r^2 \arrow[l, shift left, "R"] \end{tikzcd}$$

5.2.2 DBSA: Right Ideal Element.

Let $\hat{R}$ in a data base $<D, B, \hat{R}, A>$ consist of a single relation R and A contains the statement R = R;1 , i.e. R is a right ideal element. Final results of the computations are summarized in Fig. 1 and Fig. 2.

(Note: The 1 in R = R;1 is the Boolean identity element.)

States	Words of G*
0	λ
1	1
2	R
3	$\breve{R}$
4	1R
5	$R\breve{R}$
6	$\breve{R}1$
7	$\breve{R}R$
8	$1R\breve{R}$
9	$R\breve{R}1$
10	$\breve{R}1R$
11	$1R\breve{R}1$
12	$R\breve{R}1R$
13	$\breve{R}1R\breve{R}$
14	$R\breve{R}1R\breve{R}$

	1	R	R
0	1	2	3
1	1	4	3
2	2	2	5
3	6	7	3
4	4	4	8
5	9	2	5
6	6	10	3
7	7	7	3
8	11	4	8
9	9	12	5
10	10	10	13
11	11	4	8
12	12	12	14
13	6	10	13
14	9	12	14

Figure 1

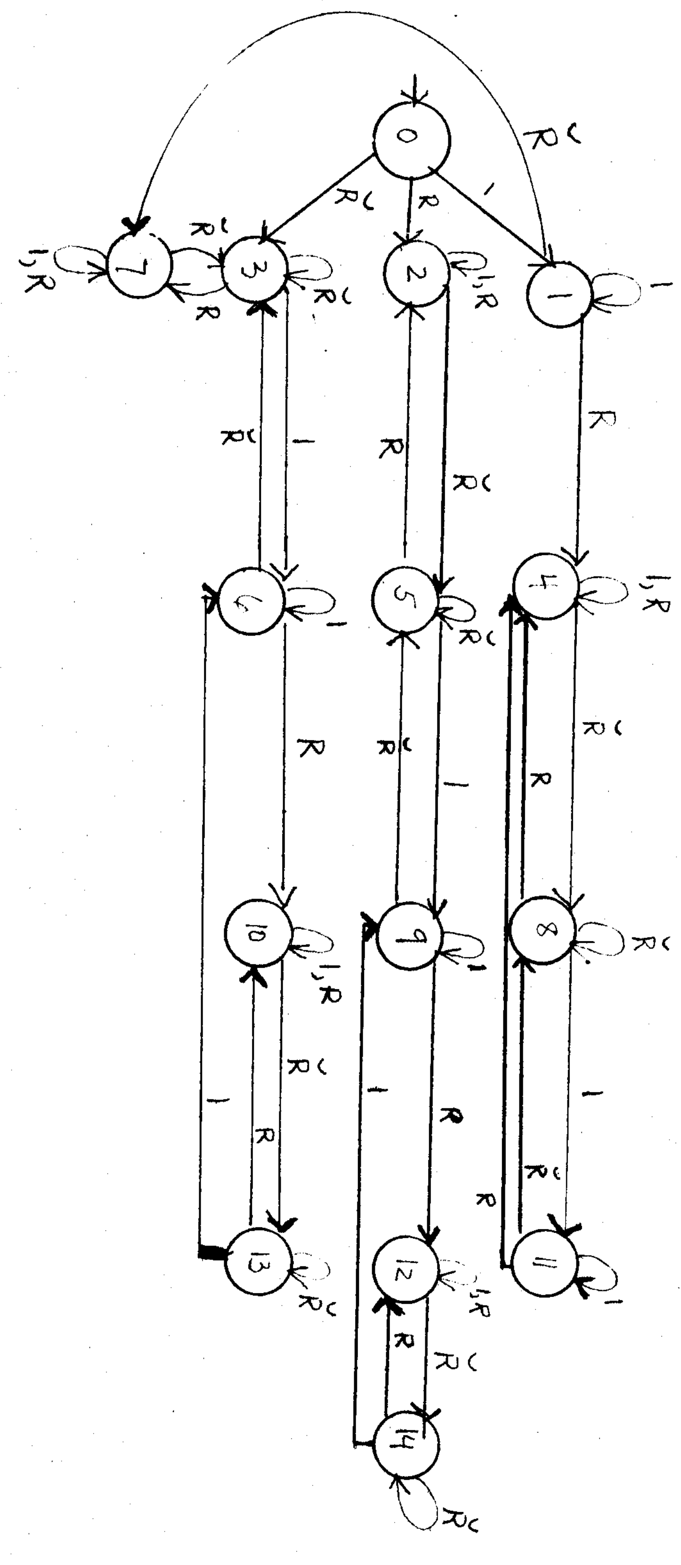


Figure 2

5.2.3 DBSA: Single Closed Relation; Third Power Empty.

Let the data base $<D, B, \hat{R}. A>$ be such that $\hat{R} = \{R\}$ and A consists of (i) $R^3 = \phi$, (ii) $R\breve{R}R = R$. Results of computations are given in Fig. 3 and Fig. 4.

States	Words of G*
0	λ
1	R
2	$\breve{R}$
3	RR
4	$R\breve{R}$
5	$\breve{R}R$
6	$\breve{R}\breve{R}$
7	$RR\breve{R}$
8	$R\breve{R}\breve{R}$
9	$\breve{R}RR$
10	$\breve{R}\breve{R}R$
11	$RR\breve{R}\breve{R}$
12	$R\breve{R}\breve{R}R$
13	$\breve{R}RR\breve{R}$
14	$\breve{R}\breve{R}RR$
15	$RR\breve{R}\breve{R}R$
16	$R\breve{R}\breve{R}RR$
17	$\breve{R}RR\breve{R}\breve{R}$
18	$\breve{R}\breve{R}RR\breve{R}$
19	$RR\breve{R}\breve{R}RR$
20	$\breve{R}\breve{R}RR\breve{R}\breve{R}$
21	ϕ

	R	R
0	1	2
1	3	4
2	5	6
3	1	7
4	1	8
5	9	2
6	10	21
7	3	11
8	12	21
9	21	13
10	14	6
11	15	21
12	16	8
13	9	17
14	21	18
15	19	11
16	21	4
17	5	21
18	14	20
19	21	7
20	10	21
21	21	21

Figure 3

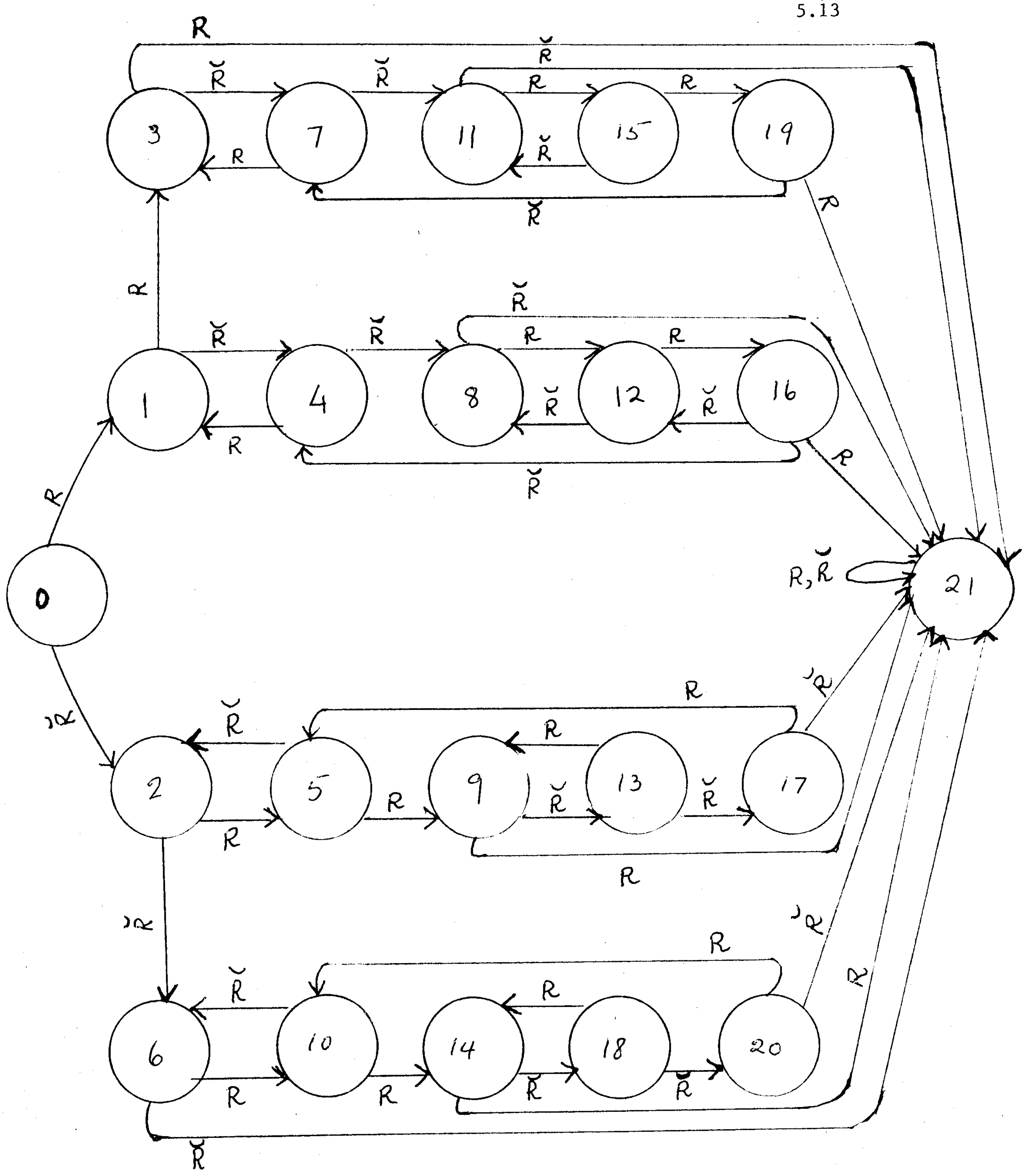

R
R̆
R̆
R̆
R
R
R
R̆
R̆
R
R
R̆
R̆
R
R
R̆
R̆
R̆
R
R
R,R̆
¬R
R
R̆
R
R
R̆
R̆
R
R
¬R
¬R
R
R̆
R
R
R
R̆
R̆
R̆
¬R
R
3
7
11
15
19
1
4
8
12
16
0
21
2
5
9
13
17
6
10
14
18
20

Figure 4

It may be noted that in constructing these DBSA only the structural properties of relations involved have been used. The property that R is closed is a rather general property possessed by many different kinds of relations. The DBSA associated with closed relations merits further study. Again, in real life situations it is likely there would be many more constraints on the finite state system. These restrictions will arise from semantic considerations, and their effect is to reduce the number of distinct states in the DBSA. This point is important if the number of states in the DBSA are to be controlled.

5.2.4. DBSA: Simple Kinship Relations

The following discussion of simple kinship relations is presented in three parts. In part one, an abstract system is developed, in part two it is interpreted and part three is a general discussion of the model and its utility.

PART I: ABSTRACT SYSTEM

Let $\langle D, B, \hat{R}, A \rangle$ be a data base where

D is a set of basic data

$B = \{Ma, Fe\}$

$\hat{R} = \{F, M\}$

A contains the following statements

1. $Ma \subseteq D$
2. $Fe \subseteq D$
3. $Ma \cup Fe = \phi$
4. $F \subseteq Ma \times D$
5. $M \subseteq Fe \times D$
6. $Ra(F) = Ra(M)$
7. $\breve{F};F = \breve{M};M$
8. $F;\breve{M}$ and $M;\breve{F}$ are 1-1 functions
9. $F;F = M;M = \breve{F};\breve{F} = \breve{M};\breve{M}$

   $= F;M = M;F = \breve{F};\breve{M} = \breve{M};\breve{F}$

   $= \emptyset$
10. $\breve{M};F = \breve{F};M = \emptyset$

11. $\breve{F}$ and $\breve{M}$ are functions. Hence it maybe noted that

(i) $F;\breve{F};F = F$

(ii) $M;\check{M};M = M$

As a result of this specification of the data base, $G = \{F,\check{F},M,\check{M}\}$ and the distinct words of G^* and their coding as states is given in Figure 5.

States	Words of G^*
Λ	λ
f	F
$\breve{f}$	$\breve{F}$
m	M
$\breve{m}$	$\breve{M}$
$f\breve{f}$	$F\ \breve{F}$
$f\breve{m}$	$F\ \breve{M}$
$\breve{f}f$	$\breve{F}\ F$
$m\breve{f}$	$M\ \breve{F}$
$m\breve{m}$	$M\ \breve{M}$
$\emptyset$	$\emptyset$

Figure 5

The DBSA associated with the data base consists of

(i) The alphabet

$$G = \{F, \breve{F}, M, \breve{M}\}$$

(ii) the set of states

$$S = \{\Lambda, \emptyset, f, \breve{f}, m, \breve{m}, f\breve{f}, f\breve{m}, \breve{f}f, m\breve{f}, m\breve{m}\}$$

(iii) the initial state Λ, and

(iv) Figures 6 and 7 give the state transition table and the state transition diagram respectively.

	F	$\breve{F}$	M	$\breve{M}$
Λ	f	$\breve{f}$	m	$\breve{m}$
f	$\emptyset$	$f\breve{f}$	$\emptyset$	$f\breve{m}$
$\breve{f}$	$\breve{f}f$	$\emptyset$	$\emptyset$	$\emptyset$
m	$\emptyset$	$m\breve{f}$	$\emptyset$	$m\breve{m}$
$\breve{m}$	$\emptyset$	$\emptyset$	$\breve{f}f$	$\emptyset$
$f\breve{f}$	f	$\emptyset$	$\emptyset$	$\emptyset$
$f\breve{m}$	$\emptyset$	$\emptyset$	f	$\emptyset$
$\breve{f}f$	$\emptyset$	$\breve{f}$	$\emptyset$	$\breve{m}$
$m\breve{f}$	m	$\emptyset$	$\emptyset$	$\emptyset$
$m\breve{m}$	$\emptyset$	$\emptyset$	m	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Figure 6

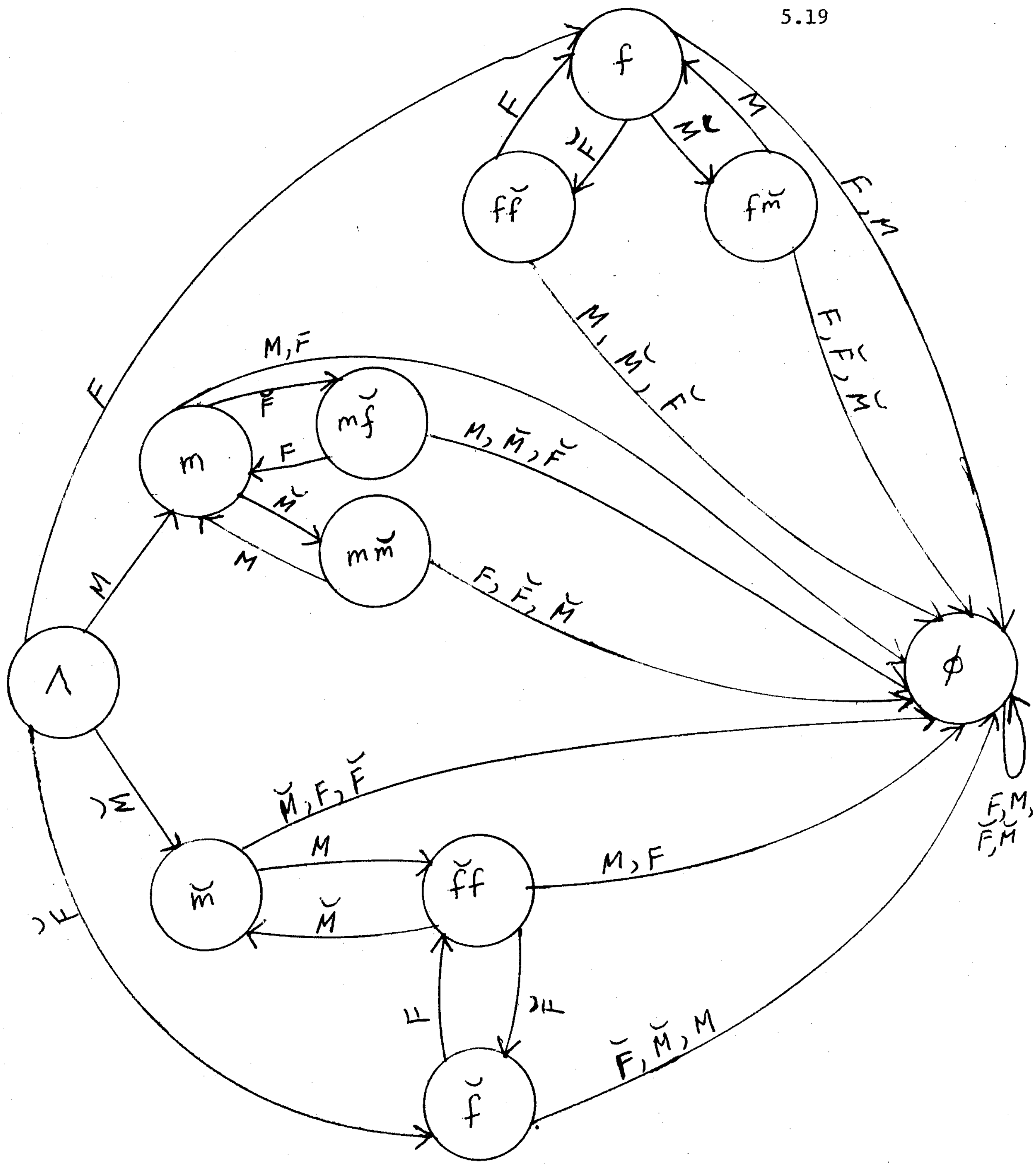


Figure 7

PART II: INTERPRETATION

The DBSA constructed in Part I can be interpreted as follows:

Take D to be a set of human beings consisting of both males and females. Let Ma be the subset of males in D and Fe be the subset of females in D. Let F be the relation: "is the father of" and M: "is the mother of". Now the assertions 1,2,3,4, and 5 in A are straight forward and must be obeyed by F and M and Fe, Ma if the interpretations attached to these symbols are to hold in the biological sense of their meanings. Thus 11 states that an individual can have exactly one father and exactly one mother. Biologically this is what is required. Again 10 states a mother cannot also be a father and vice versa.

The statements 6 and 7 may be interpreted as children who are legitimate and parents and their children all are in D and are recorded as such.

Item 8 will be valid in a monogamous society.

The assertions in 9 effectively mean that second generation information is not available in the data included in D.

As a concrete example, let 0, 1, 2, 3 be names of individuals. Then:

$$D = \{0, 1, 2, 3\}$$
$$B = \{Fe, Ma\}$$
$$Fe = \{3, 1\}$$
$$Ma = \{0, 2\}$$
$$F = \{(0, 1), (0, 2)\}$$
$$M = \{(3, 1), (3, 2)\}$$

satisfies the requirements of the abstract DBSA.

PART III: DISCUSSION

The first point to note is that the DBSA was obtained in a very general and abstract way without any appeal to the contents of the sets D, B, $\hat{R}$. (Here the term content is being used to mean what a relation in $\hat{R}$ may actually contain). Thus the derivation of DBSA is clearly data independent and depends upon the structural properties specified by A.

The DBSA summarizes the constraints and restrictions (at least some of them) included in A. It would be rather awkward to work directly with the axioms in A. The DBSA obtained not only reflects the axioms in A but also what can be deduced under composition and inversion of relations. Thus DBSA is a powerful tool of control and regulation. Furthermore as a finite state mechanism it is a convenient system to work with. This above justifies studying these special semi-automata .

The DBSA as constructed will serve as a model for any system of actual relations and data which satisfy the constraints specified in A. It may be noted that in the actual derivation of the DBSA only those constraints in A which were in the form of equations were utilized. The constraints which are in the form of inequalities were not fully utilized.

The DBSA so constructed can serve as a tool for control and regulation of access and updating of information. For example, if the relation $F;\check{M}$ were to be updated, then from the transition diagram it is clear what other relations must also be updated, since the state $f\check{m}$ (corresponding to $F;\check{M}$) is connected to other states which correspond to other relations. The significance of this really lies in the fact that DBSA summarizes these interconnections in such a remarkably clear and precise way.

The interpretation of the abstract system as a very simple kinship model points out the general thrust of such an approach. By including in A sufficient number of axioms, it is likely that this method will succeed in providing means of modeling complex situations in a realistic way, hence the construction of useful data bases for the same.

In the above example a possible implementation might be to maintain only two relations M and F as well as F_e, M_a, D and B sets. It could be deemed advisable to maintain further redundant relations such as $M;\breve{F}$ should there be sufficient activity within the data base usage.

Consider a simple update which may involve the addition of a new pair (x, y) to the set M . The pair (x, y) can be one of four possible types listed below.

a) x is an item which is new in the sense that it does not already appear in the domain of M , and similarly y is also new in the sense that it does not appear in the range of M .

b) x is a new item but y is an old item in the sense that y already does appear in the range of M .

c) x is an old item and y is a new item.

d) x is an old item and y is an old item, i.e. (x, y) is already in M .

Any data item incorporated in the relation M by an update procedure must result in updated versions of M and F which must satisfy exactly the same axioms which were used to generate the state transition diagram for the data base. Only then the state transition diagram will remain applicable to the updated relations. If the above conditions are satisfied then the update procedure will leave the composition structure of the data base invariant and the use of the transition diagram will continue to yield correct results.

The procedure for updating M is given below. For convenience P denotes $\{(x,y)\}$ and M' and F' denote the updated versions of M and F respectively.

It may be remarked at the outset that the set of axioms listed is not irredundant. Thus if $M;M = \phi$ (axiom 9) is satisfied then $\breve{M};\breve{M} = \phi$ is also satisfied. Hence, even though a rather complete set of axioms are listed, due to redundancy it is not necessary to test them all. The procedure for the update is to validate the updated data in conjunction with the axioms.

1. Compute $M' = M \cup P$.

If $M' = M$ then stop, because the pair (x,y) must be of type (d) and is already in M; otherwise, continue with step 2.

2. Compute $M';\breve{M}'$.

Since axiom 11 asserts that $\breve{M}$ is a function, its updated version $\breve{M}'$ must also remain a function and must satisfy $M';\breve{M}' \subseteq 1'$.

If $M';\breve{M}' \nsubseteq 1'$ then the update must be rejected. This situation will occur if (x,y) is of type (b): if such an update was allowed, then y will appear twice in first positions of two ordered pairs in $\breve{M}'$ and $\breve{M}'$ would no longer be a function.

If $M';\breve{M}' \subseteq 1'$ then proceed to step 3.

3. Compute $M';M'$.

If $M';M' \neq \phi$ reject the update and stop (axiom 9); otherwise proceed to step 4.

4. Compute $F;\breve{M};P$.

Since some axioms such as (7) or (8) establish connection between M and F, hence a change in M may imply a change in F which must also be changed to ensure the consistency of the data base.

If $F;\breve{M};P = \phi$ then the update cannot be completed solely in terms of information available in (x,y), F and M, because $F;\breve{M};P = \phi$ indicates the attempt to associate a father with y through y's mother has failed. This can happen if (x,y) is of type (a) or (b). However, type (b) would be rejected at step 2, hence (x,y) must be of type (a).

If $F;\breve{M};P \neq \phi$, then update F by $F' = F \cup (F;\breve{M};P)$. This will be the case if (x,y) is of type (c) and hence information about the father of x's children is in the data base and hence a father can be associated with a new child of x.

It should be remarked that if a pair of type (a) was entered after obtaining additional information then to guarantee the integrity of the data base some other tests similar to those in step 3 (i.e. drawn from axioms (9) and (10)) will have to be carried out.

This update procedure may be illustrated as follows:

$M = \{(3,1), (3,2), (4,5)\}$

$F = \{(0,1), (0,2), (6,5)\}$

$F;\breve{M} = \{(0,3), (6,4)\}$

Pair (x,y)	Type	M'	$M';\breve{M}'$	M';M'	$F;\breve{M};P$	F'	Remarks
(7,8)	(a)	{(3,1),(3,2),(4,5),(7,8)}	{(3,3),(4,4),(7,7)}	ϕ	ϕ		Request more information: who is the father of 8?
(7,2)	(b)	{(3,1),(3,2),(4,5),(7,2)}	{(3,3),(3,7),(4,4),(7,7),(7,3)}				Reject update. Attempt to associate a second mother with 2.
(3,7)	(c)	{(3.1),(3,2),(4,5),(3,7)}	{(3,3),(4,4)}	ϕ	{(0,7)}	{(0,1),(0,2),(6,5)(0,7)}	Valid update. Note 0 has been asso-ciated with 7 as father
(1,7)	(a)	{(3,1),(3,2),(4,5),(1,7)}	{(3,3),(4,4),(1,1)}	{(3,7)}			Reject. Attempt to introduce second generation information
(3,1)	(d)	{(3,1),(3,2),(4,5)}					Redundant update

The first case illustrates that (7,8) , a type (a) pair, should not be incorporated in M unless further information enabling the update of F is also made available. In such cases where partial information is available the update is not rejected but only its completion is delayed till more information becomes available.

In contrast in the fourth case (1, 7), a type (a) pair, and in the second case (7, 2), a type (b) pair are rejected because in each case the updated versions of M entail a negation of an axiom. Such updates can only be completed if the underlying axiom system is changed, which would imply a change in the state transition diagram. In this sense such changes may be classified as those which imply a structural change to the system.

In the remaining third and the fifth cases there is sufficient information available and no structural changes are implied. Hence updates pose no problems and are accepted.

By prohibiting updates which imply changes to the structure of the data base and delaying those updates which require further information for completion, the invariance of the structure of the data bases as specified by the underlying axioms and represented by the state transition diagram can be guaranteed. This invariance implies that data base operations which were correct prior to an update remain correct after the update. Furthermore data incorporated into the data base by appealing to the axioms, guarantees that its consistency in so far as specified by the axioms is preserved and the new data does not conflict with any data that was already in the data base. It is in this sense that the integrity of the data base is realized in this model.

As an example of the use of the state transition diagram (or the table) consider a query formulated as

$$F;\breve{M};M;\breve{F};F;\breve{M}$$

There are at least three possibilities to process this query.

1. By direct computation of the query expression.
2. By first using the axioms to reduce the expression to a simpler form (i.e. $F;\breve{M}$) and then computing it. In this method the reduction of the length of the expression by axioms is essentially a pattern matching operation which can involve heuristics and adhoc procedures.
3. By using the state transition diagram (or table) to reduce the expression to $F;\breve{M}$ and then computing it. This method requires that the transition diagram be known, but once known it can be used in a direct way to simplify query expressions by reducing their length. This method becomes impractical if the state diagram is too large. It should be stated here that the axioms when used as equations will result in a reduction in the number of states in the data base automaton. In this sense the size of the automaton depends upon the number of reduction resulting from the use of the axioms. To illustrate this point the axiom "$\breve{M}$ is a function" is expressed by

   $$M;\breve{M} \subseteq 1' \quad .$$

   However this implies that $M;\breve{M};M = M$. This equation implied by the axiom serves to reduce the number of states in the automaton, hence in the state transition diagram. If the set of axioms are such that no reduction in the number of states is possible, then this method is inapplicable. A class of such applications in which the set of axioms results in a significant reduction of states are those data bases where the relations in it are functional in character.

5.2.5. DBSA: Real Estate Directory.

In the following a data base model for a real estate directory is considered. A real estate directory is essentially a listing of owners and the properties they own or persons living at various residences in a city. It is essentially a many-many relation.

The relations normally termed as many-many type are rather algebraically indigent and hence as will be seen special effort has to be made to introduce some algebraic structure in the system.

PART I: ABSTRACT SYSTEM.

Consider a data base $\langle D, B, \hat{R}, A\rangle$ such that B is a collection of sets $\tilde{A}$, I, and N all subsets of D. $\hat{R}$ contains two relations P and R and A is the set of following assertions.

1. $I \cap N = \emptyset$
2. $R \subseteq I \times \tilde{A}$
3. $P \subseteq \tilde{A} \times N$
4. $R;R = \emptyset$
5. $P;P = \emptyset$
6. $R;\breve{R};R = R$
7. $P;\breve{P};P = P$
8. Both R and P are functions.

The set G is $\{R, \breve{R}, P, \breve{P}\}$ and the distinct words of G^* may be computed in the usual manner. However here their derivation is shown and arranged for convenience in the form of tables. In effect, each of these tables is a composition table. In each table elements to be composed are elements obtained in a previous table. The derivation is started by constructing a table with elements in set G. The process stops with the table in which no new elements are introduced.

(a)

;	R	$\breve{R}$	P	$\breve{P}$
R	$\emptyset$	$R;\breve{R}$	$R;P$	$\emptyset$
$\breve{R}$	$\breve{R};R$	$\emptyset$	$\emptyset$	$\emptyset$
P	$\emptyset$	$\emptyset$	$\emptyset$	$P;\breve{P}$
$\breve{P}$	$\emptyset$	$\breve{P};\breve{R}$	$\breve{P};P$	$\emptyset$

(b)

;	R	$\breve{R}$	P	$\breve{P}$
$\breve{R};R$	$\emptyset$	$\breve{R}$	$\breve{R};R;P$	$\emptyset$
$R;\breve{R}$	R	$\emptyset$	$\emptyset$	$\emptyset$
$R;P$	$\emptyset$	$\emptyset$	$\emptyset$	$R;P;\breve{P}$
$\breve{P};P$	$\emptyset$	$\emptyset$	$\emptyset$	$\breve{P}$
$P;\breve{P}$	$\emptyset$	$P;\breve{P};\breve{R}$	P	$\emptyset$
$\breve{P};\breve{R}$	$\breve{P};\breve{R};R$	$\emptyset$	$\emptyset$	$\emptyset$

(c)

;	R	$\breve{R}$	P	$\breve{P}$
$\breve{P};\breve{R};R$	$\emptyset$	$\breve{P};\breve{R}$	$\breve{P};\breve{R};R;P$	$\emptyset$
$P;\breve{P};\breve{R}$	$P;\breve{P};\breve{R};R$	$\emptyset$	$\emptyset$	$\emptyset$
$\breve{R};R;P$	$\emptyset$	$\emptyset$	$\emptyset$	$\breve{R};R;P;\breve{P}$
$R;P;\breve{P}$	$\emptyset$	$R;P;\breve{P};\breve{R}$	$R;P$	$\emptyset$

(d)

;	R	$\breve{R}$	P	$\breve{P}$
$P;\breve{P};\breve{R};R$	$\emptyset$	$P;\breve{P};\breve{R}$	$P;\breve{P};\breve{R};R;P$	$\emptyset$
$R;P;\breve{P};\breve{R}$	$R;P;\breve{P};\breve{R};R$	$\emptyset$	$\emptyset$	$\emptyset$
$\breve{P};\breve{R};R;P$	$\emptyset$	$\emptyset$	$\emptyset$	$\breve{P};\breve{R};R;P;\breve{P}$
$\breve{R};R;P;\breve{P}$	$\emptyset$	$\breve{R};R;P;\breve{P};\breve{R}$	$\breve{R};R;P$	$\emptyset$

(e)

;	R	$\breve{R}$	P	$\breve{P}$
$R;P;\breve{P};\breve{R};R$	$\emptyset$	$R;P;\breve{P};\breve{R}$	$R;P$	$\emptyset$
$\breve{R};R;P;\breve{P};\breve{R}$	$\breve{R};R;P;\breve{P};\breve{R};R$	$\emptyset$	$\emptyset$	$\emptyset$
$P;\breve{P};\breve{R};R;P$	$\emptyset$	$\emptyset$	$\emptyset$	$P;\breve{P};\breve{R};R;P;\breve{P}$
$\breve{P};\breve{R};R;P;\breve{P}$	$\emptyset$	$\breve{P};\breve{R}$	$\breve{P};\breve{R};R;P$	$\emptyset$

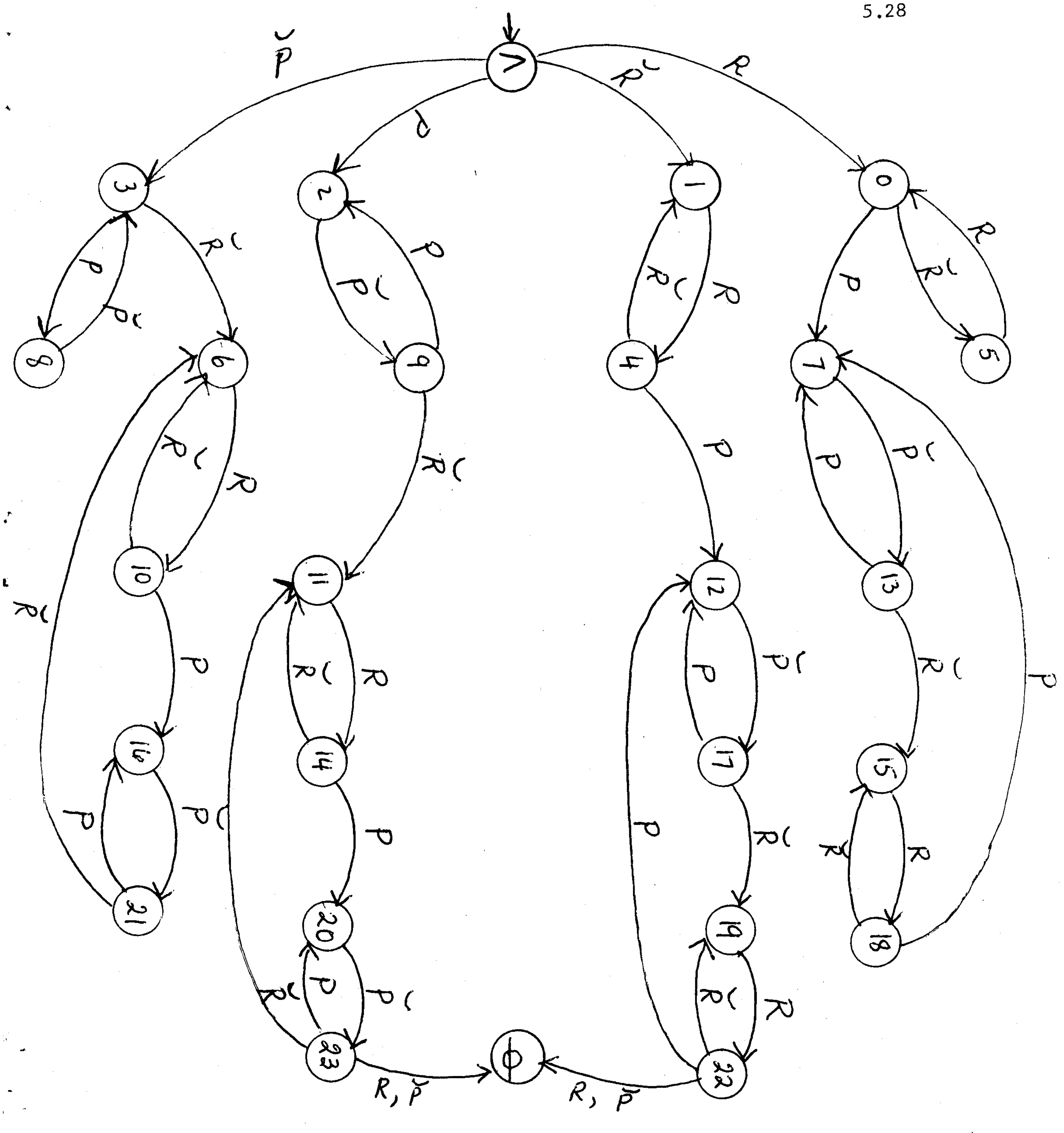
(f)

;	R	$\breve{R}$	P	$\breve{P}$
$\breve{R};R;P;\breve{P};\breve{R};R$	$\emptyset$	$\breve{R};R;P;\breve{P};\breve{R}$	$\breve{R};R;P$	$\emptyset$
$P;\breve{P};\breve{R};R;P;\breve{P}$	$\emptyset$	$P;\breve{P};\breve{R}$	$P;\breve{P};\breve{R};R;P$	$\emptyset$

The information in these tables is summarized in the table in Figure 8. The first column of this table lists the words of G^*, which are coded in the second column which constitutes a list of states. The remaining part of the table with column 2 constitutes the transition table for the DBSA associated with the data base under consideration. The transition diagram appears in Figure 9.

Words of G^*	State	R	$\breve{R}$	P	$\breve{P}$
R	0	∅	5	7	∅
$\breve{R}$	1	4	∅	∅	∅
P	2	∅	∅	∅	9
$\breve{P}$	3	∅	6	8	∅
$\breve{R}\ R$	4	∅	1	12	∅
$R\ \breve{R}$	5	0	∅	∅	∅
$\breve{P}\ \breve{R}$	6	10	∅	∅	∅
$R\ P$	7	∅	∅	∅	13
$\breve{P}\ P$	8	∅	∅	∅	3
$P\ \breve{P}$	9	∅	11	2	∅
$\breve{P}\ \breve{R}\ R$	10	∅	6	16	∅
$P\ \breve{P}\ R$	11	14	∅	∅	∅
$\breve{R}\ R\ P$	12	∅	∅	∅	17
$R\ P\ \breve{P}$	13	∅	15	7	∅
$P\ \breve{P}\ \breve{R}\ R$	14	∅	11	20	∅
$R\ P\ \breve{P}\ \breve{R}$	15	18	∅	∅	∅
$\breve{P}\ \breve{R}\ R\ P$	16	∅	∅	∅	21
$\breve{R}\ R\ P\ \breve{P}$	17	∅	19	12	∅
$R\ P\ \breve{P}\ \breve{R}\ R$	18	∅	15	7	∅
$\breve{R}\ R\ P\ \breve{P}\ \breve{R}$	19	22	∅	∅	∅
$P\ \breve{P}\ \breve{R}\ R\ P$	20	∅	∅	∅	23
$\breve{P}\ \breve{R}\ R\ P\ \breve{P}$	21	∅	6	15	∅
$\breve{R}\ R\ P\ \breve{P}\ \breve{R}\ R$	22	∅	19	12	∅
$P\ \breve{P}\ \breve{R}\ R\ P\ \breve{P}$	23	∅	11	20	∅
λ	Λ	0	1	2	3
ϕ	∅	∅	∅	∅	∅

TRANSITION TABLE
Figure 8



State Diagram

Figure 9
(arrows not shown all go to the state ϕ)

PART II: INTERPRETATION AND DISCUSSION

A real estate directory is a compilation of the following two associations:

(1) real estate property and its owner,

(2) individuals and their addresses,

which could be represented by two binary relations P and R respectively.

A real estate directory is a many-many relation. Normally this type of data is organized in a dictionary fashion. A dictionary implementation is a suitable ordered representation of the many-many relation. Now this representation of the relation does not use any redundancy that may be existing in any significant way. If a DBSA is constructed without further structural examination of this relation, the number of states in it tends to increase rapidly. If the number of states in DBSA is to be controlled the relation involved must be carefully examined to discover and use any redundancy which may serve to structure it. If the portion of directory denoted by R is interpreted to be the association between an individual and the address of his principal residence, then R becomes a function. However, P if interpreted to have as its domain a subset of individuals then P becomes a many-many relation and contributes to an "explosion" of states in DBSA. A solution is to consider one or several owners of a real estate property to be a single entity - a set - and attach a name to it. Then P also becomes a function and with such a P the DBSA has a manageable number of states. R, P, I, N and $\tilde{A}$ be interpreted as follows:

I	set of names of individuals
N	set of names of property owners
$\tilde{A}$	set of names of properties
R	xRy means individual x resides at property y
P	xPy means property x is owned by y.

The axioms of A are satisfied, provided the owners of a property are viewed as a single entity with a name.

In this interpretation it is important to note that control over the number of states in DBSA has essentially been achieved by clearly separating syntactic and semantic primitives. I, N, $\tilde{A}$, R and P without interpretation provide names and relations on these names. What these names stand for is established by a semantic process. For example, an $n \in N$ is the name of a set. However, at the syntactic level it is just a name similar to $i \in I$ which is not the name of a set but that of an individual. That that is the difference in the nature of these two names is established by appealing to their meaning or intended use. Clearly, at the implementation level an $n \in N$ will have a different type of representation than that of $i \in I$. It is in this sense the model being considered is more at a logical level than at the implementation level. Thus in a model of a data base as $\langle D, B, \hat{R}, A\rangle$ the logical and algebraic relationships play a paramount role and details of implementation are suppressed.

The system under discussion is capable of modeling the following real estate relations encountered in practice.

xRy:	x resides at y
$x\breve{R}y$:	x is the residence of y
xPy:	x belongs to y
$x\breve{P}y$:	x owns y
$xR;\breve{R}y$:	x resides at the same residence as y, i.e. x,y are occupants of a common residence
$x\breve{R};Ry$:	identity on residences
$xR;Ry$:	x's land lord is y
$x\breve{P};\breve{R}y$:	x's tenant is y.
$x\breve{R};R;Py$:	residence x belongs to y.
$x\breve{R};R;P;\breve{P}y$:	residence x belongs to owner of y , i.e. x,y belongs to the same owner and x is owner's residence.

CHAPTER VI CONCLUSIONS

For data bases there are a number of important advantages which result from adopting an algebraic point of view. It allows the structure of data base to be succinctly specified. This is so because it is usually possible to characterize algebraic systems by only a few axioms. The structure of a data base may be displayed explicitly using algebraic methods thus facilitating better control and regulation. This in turn allows an examination and understanding of the consequences of the possible changes in the basic structure of a data base. Furthermore algebra provides a powerful language and other formal tools to describe and study the properties of data base systematically. It also enhances clarity, systematic growth and provides a scheme for data base semantics. Again through the mechanism of homomorphisms it provides links with other algebraic systems. These advantages if fully realized make the algebraic approach a very powerful tool for investigating data bases as abstract entities. Finally it must be noted that for most algebraic systems a well developed decomposition theory already exists. If the connection between data bases and such algebras is successfully established, this aspect will provide valuable insights into the problem of data base organization and its structure.

To specify the structure of relations an appropriate language is required. There exist a number of languages capable of handling relations as separate entities. Two major categories may be noted. On the logical level there are relation calculi and on the algebraic level there are algebras such as relation algebras or cylindric algebras [Henkin et al. 1971]. There languages provide formal means of describing the structure of relations and hence also of data bases.

Relation algebra is an algebraic system designed to model the class of binary relations. The language provided by this algebra is adequate to specify the structure of such relations and describe operations on them. Accordingly a data base containing binary relations provides an application area for these algebras. If a data base contains more complex relations such as n-ray relations then cylindric algebras are appropriate. In any case algebraic approach to data bases merit further study and investigation.

A data base as a system consisting of a collection of data on which a community of relations is defined, models the interrelationship among parts of the physical reality represented by the data through the relations embedded in it.

The properties of the objects represented must also be modelled. This is done by the mechanisms of subsets. Each property is used to induce a decomposition of data into subsets. These subsets then provide adequate attribute modelling capability. Thus in a data base there are data, their subsets and relations defined on them.

The development of a data base consists of specifying the explicit structure of the relations and the decomposition of the data induced by the subsets in it, then using these to provide facilities for performing data base operations and achieving data base objectives of security, integrity and correctness algebraically. This notion of data base development when exploited, as is done in this thesis, provides a formal model of data bases with its own algebras and finite state systems.

The concept of data base as an abstract entity and its relation to finite state systemssuggests the possibility of direct hardware realization as sequential circuits. In view of the fact that in a data base there may only be a few relations but that each relation may have a very large underlying aggregate of data which will come into play when performing data base operations, some form of associative processing will be necessary. This aspect of the realization of data base must be carefully examined because if the structure of data base is to remain invariant when data base operations are performed, the underlying aggregate of data will play a part in preserving the structure of the data base. If this implies that large portions of this aggregate data have to be examined, then associative or parallel processing becomes a mandatory requirement, and requires further study.

The security of data in a data base may be achieved by using its state transition diagram as a labelled graph. The access to various states in the diagram may be controlled by label - ing the arcs connecting the states. These labels may be used to determine who may access a state and in what manner. This aspect of the data base requires further study and exploration.

BIBLIOGRAPHY

1. Brooks F. P. and Iverson K. E., (1969) Automatic Data Processing, Wiley, New York, p. 466.
2. Carnap R. (1958) Introduction to Symbolic Logic and Its Applications, Dover, p. 241.
3. Childs D. E. (1968) "Feasibility of a set theoretic data structure" in Proc. of the IFIP Congress 1968, V. 1, p. 420 - 430, North-Holland, Amsterdam.
4. Chin L. H. and Tarski A. (1950) "Distribution and modular laws in the arithmetic of relation algebras" in Univ. of Calif. Publications in Maths. V. 1, No. 9, p. 341 - 384.
5. CODASYL Development Committee, (1962) "An Information Algebra - Phase I report", Comm. ACM, V. 5, p. 190 - 204.
6. Codd E. F., (1970) "A relational model of data for large shared data banks", Comm. ACM, V. 13, p. 377 - 387.
7. Codd E. F. (1972a) "Further normalization of the data base relational model" in Data Base Systems, R. Rustin (Ed). Prentice Hall New Jersey, p. 33 - 64.
8. Codd E. F. (1972b) "Relational completeness of data base sublanguages" in Data Base Systems, R. Rustin

9. Feldman J. A. and Rovner P.D. (1969) "An ALGOL-based associative language" Comm. ACM. V. 12, p. 439 - 449.

10. Florentin J. J. (1974) "Consistency auditing of data bases" The Computer Journal, V. 17, No. 1, p. 52 - 58.

11. Galler B. A. and Perlis A. J. (1970) A View of Programming Languages, Addison - Wesley p. 282.

12. Ginzburg A. (1968) Algebraic Theory of Automata. Academic Press New York, p. 165.

13. Gratzer. G. (1971) Lattice Theory, W. H. Freeman and Co. San Francisco, p. 212.

14. Halmos P. R. (1967) Lectures on Boolean Algebra, Van Norstrand, p. 147.

15. Hays R. M. (1968) "The decomposition of vocabulary hierarchies" in Mechanized Information Storage, Retrieval and Dissemination, K. Samuelson (Ed.), North-Holland Amsterdam, p. 160 - 191.

16. Henkin L., Monk J. D., and Tarski, A. (1971) Cylinderic Algebras, Part I, North-Holland Amsterdam, p. 508.

17. Hsiao D. and Harary F., (1970) "A formal system for information retrieval from files" Comm, ACM, V. 13, p. 67 - 73.

18. Kobayashi I., (1972) "An algebraic model of information structure and information processing", Proc. ACM Annual Conference, August 1972, Boston, p.1090-1104.

18a. Knuth, D.E. (1973) The Art of Computer Programming. V. 3, Sorting and Searching, Addison-Wesley, Reading, p. 722.

19. Langfors B. (1967) Theoretical Analysis of Information Systems, Studentlitteratur Lund, V. 1, p. 220, V. 2, p. 180.

20. Levien R. and Maron M. (1967) "A computer system for inference and execution and data retrieval" Comm, ACM, V. 10, p. 715 - 721.

21. Nero C. D. (1969) "An approach to the formal description of the lamda system" in Information Processing 1968: Proc. of IFIP Congress 1968, V. 1, A. J. H. Morrel (Ed.), North-Holland Amsterdam.

22. Ono K. (1957) "On some properties of binary relations" Nagoya Math. Jour. V. 12, p. 161 - 170.

23. Senko et al. (1973) "Data structures and accessing in data base systems" IBM Systems Jour. V. 12, No. 1, p. 30 - 93.

24. Tarski A. (1941) "On calculus of relations" Jour. Sym. Logic, V. 6, p. 73 - 89.

25. Turski W. M. (1971) "A model for data structures and its applications", Acta Information V. 1, p. 26 - 34.

26. Westin A. F. (1972) Data Banks in Free Society, Quadrangle Books, p. 522.

27. Weyl H. (1949) Philosophy of Mathematics and Natural Science, Princeton University Press, Princeton, p. 311.

28. WILKES, M. V. (1972) "On preserving the integrity of data bases" The Computer Journal, V. 15, No. 3, p. 191 - 194.

29. Wong E. and Chiang T. C. (1971) "Canonical Structure in attribute based file organization" Comm, ACM, V. 14, No. 9, p. 593.