SUGGESTIONS FOR A UNIFORM REPRESENTATION OF QUERY AND RECORD CONTENT IN DATA BASE AND DOCUMENT RETRIEVAL

by

Gerard Salton*

TR79-363

Computer Science Department
Cornell University
Ithaca, NY 14853

*This study was supported in part by the National Science Foundation under Grant DSI-77-04843.

SUGGESTIONS FOR A UNIFORM REPRESENTATION OF QUERY AND RECORD CONTENT IN DATA BASE AND DOCUMENT RETRIEVAL

G. Salton*

Abstract

A standard approach is introduced for the representation of information content in data base and document retrieval environments. The use of composite concept vectors representing individual information items leads to a uniform system in different retrieval situations for the identification of answers in response to incoming information requests.

---

*Department of Computer Science, Cornell University, Ithaca, New York 14853.

This study was supported in part by the National Science Foundation under grant DSI-77-04843.

1. Introduction

Data-base retrieval systems are designed to process fixed-format, standardized records of the kind prevalent in many business applications. Typical examples are employee records in a personnel file, or components and parts records in an inventory control system. In such a situation, a small number prespecified terms, or descriptors is used to identify each record, and items are retrieved whenever a complete match is obtained between query formulations and the corresponding record specifications.

In document retrieval, the stored items normally represent journal articles, books, and other bibliographic items. The identifying vocabulary is large and often open-ended, consisting of objective terms to denote authorship, publisher, and the like, as well as content terms to reflect the information content. Because the content description may be tentative and uncertain, the retrieval of bibliographic items may depend on approximate, or partial similarity computations between queries and record identifiers.

While the conventional implementations of data base management on the one hand, and document retrieval on the other appear quite different on the surface, the problems faced by the respective system designers are closely related: in both environments, it is necessary to define a system of knowledge representation, including the choice of information identifiers and of data structures embodying the data under consideration. Common problems arise also in the formulation of information requests, the strategies used to search and traverse the data base, and the choice of items to be displayed in response

to individual queries. In particular, it is always necessary to compare a tentative description of information need with the description of the stored information items, and the reach a retrieval decision based on the degree of similarity between query and stored information. Furthermore, similarities can be detected in the representation of the information items leading to the definition of "clustering" properties among the items. These clustering effects can in turn be used to generate special file organizations and sometimes also efficient search strategies.

The basic components of knowledge representation are briefly summarized in the remainder of this study. A vector representation is then introduced leading to a common treatment of stored information items and information requests in the various retrieval environments.

2. Knowledge Representation in Data Base Systems

The conventional data base management systems are capable of manipulating three kinds of constructs: entities, attributes of entities, and relationships between entities. An entity may represent some notion existing in the real world which is of interest in a given processing environment; for example, a customer or an employee in a business organization, an object being sold or bought in a sales ledger, or a flight number or city in a travel reservation situation. It is generally possible to represent groups of similar entities (entity sets), such as the set of all customers of a given business, as well as individual instances of entities. Attributes represent properties of entities, such as the address of a person, the job classification of an employee, the length or color of a certain component part, the carrying capacity of a

given airplane. With each instance of a given entity it is convenient to associate the values of the corresponding attributes, chosen from a domain of permitted values. Thus a given employee might be characterized by a particular employee number, a specific address, a given job classification, and a particular salary figure.

Two kinds of relationships between entities are often distinguished:

a) The hierarchical inclusion relations which are independent of content and always hold in every processing environment. Typical examples are the relations between individual cities and the states in which they are located, or between children and their parents, or between individual employees and the class of all human beings.

b) The contextual relations between entities which are valid only in a particular situation -- for example, between individual airplanes and the pilots that fly them at a particular time, or between individual sales persons and their customers.

To characterize a given data base, it is necessary to choose the entities and their attributes, as well as the hierarchical and contextual relations between them. This may be a difficult task because the boundaries between the various descriptive components are uncertain. Thus, a particular concept might be represented as a separate entity, or alternatively as an attribute of other entities -- for example, the courses offered in a given school may be separately listed, or they may be included as attributes of the individual students enrolled in them, or of the instructors that teach them. The relationships may also receive varying treatments: they may be explicitly define

and represented in the data base; alternatively they may be implicit such as when a common attribute appearing in the definition of distinct entity sets is used to relate the corresponding entities.

Different conventions exist for representing the elements in a data base system. Normally, a file is represented by exhibiting the corresponding entity set as well as the attributes used in the entity descriptions. A file of employee records may thus appear as

Employee (Name, Address, Jobcode, Salary)

indicating that each employee is represented by specific values of the corresponding attributes. Relations may be indicated by arrows or lines on a diagram connecting the corresponding entity vectors. Hierarchical relations in particular, are always represented by a vertical displacement on a page, where the hierarchically governing entity set appears higher on the page than the corresponding dependent set. Typical examples are shown in Figure 1.

3. Extensions of Knowledge Representation

The representational features described in the previous section, consisting of entities, attributes of entities, and relations between entities are all included in the three well-known abstractions of data base processing: the relational, hierarchical, and network models. [1,2] In recent years, efforts have been made to extend the application of data base technologies to other more complex information environments. Thus experimental systems have been devised in which free-form, English-like query statements can be used to address data files covering a wide assortment of different subject areas, the idea being

to render the data base environment accessible to untrained users who might find the use of English-like query languages more congenial than the standard algebra or calculus languages usually available in data base retrieval.

When English-like query languages are utilized for purposes of query formulation, it becomes necessary to generate formal statements from the free-form input. The formal version can then in turn be transformed into a sequence of search and retrieval commands to be executed by the processing system. Of the many possible formalizations, some type of predicate calculus is often used because most indicative mood statements in the natural language can be unambiguously expressed in predicate logic given the right choice of variables and predicates*.

From the point of view of knowledge representation it now becomes necessar to extend the conventional data base models in two directions:

a) on the other hand, methods must be provided for the representation of statements in predicate logic including in particular Boolean connectives and quantification;

b) on the other hand, many of the components included in standard language understanding systems must be represented, including in particular a specification of the world knowledge covering the area of discourse under consideration; this additional information is often presented in the form of semantic maps and constructs such as frames, scripts and schemas.

---

* A predicate is a logical function of one or more variables producing a truth value (true,false). Each variable can represent an object, or a set of objects and the predicates are used to formulate the relationships and attributes betwe objects. Typical statements in predicate logic are (predicates are underlined)

"x is a prime number"
"y is the mother of z"
"for all x, predicate P holds"
"there is an x for which predicate P holds."

A number of different methodologies have been proposed in an attempt to satisfy these dual requirements. [3,6] Normally the solution takes the form of a graph structure in which the nodes represent concepts including objects, events, assertions, actions, abstractions and functions, as well as properties and characteristics of objects. The links between the nodes identify the hierarchical and contextual relations between the concepts, as well as other special information about concepts. Links may thus be used to represent the syntactic and/or semantic function of the concepts; to distinguish classes of objects from individual instantiations; to carry Boolean and other connectives between concepts as well as quantifiers; to represent functions and predicates for the objects; and to supply certain timing and inference-making capabilities.

It is impossible in the present context to describe in detail the morphology and structure of existing semantic graphs. A few examples taken from the literature must suffice. [7-11] In Figure 2(a), the "case" labels help in specifying the semantic context by supplying syntactic or semantic roles for the nodes under consideration; typical case labels are "agent", "instrument", "characteristic", "object", "result", and so on. [6] In Figures 2(b), the branch labels are used to identify hierarchical inclusion relations.

Special nodes are introduced in Figure 2(c) to represent arithmetic and other operators, including also quantification. [5, 9] These quantifiers could also be identified by special branch labels as shown on the right-hand side of Figure 2(c). Sequences of events and plots, or story lines can be represented by flowchart-like sequences of graphs in which the time factors is represented explicitly by special nodes or implicitly by the chaining of nodes and branches. Simplified examples are shown in Figures 2(d) and 2(e).

The semantic graph representation has the advantage of producing a fairly transparent picture of a given situation. Unfortunately, the graphs are difficult to construct and to manipulate because they lack homogeneity in the sense that both the nodes and the branches can represent a widely differing array of constructs. Thus in retrieval, the matching operations between the graphs representing the information requests and the graphs characterizing the stored objects are not generally straightforward. The graph transformation and restructuring operations necessarily depend on the particular graph structure and the individual application, as demonstrated by the variety of examples in Figure 2.

Labelled graphs can be transformed into linear structures such as compound vectors where vector sets represent concepts as well as attributes and relationships. Vectors are easier to manipulate than graphs and they can be utilized in many different environments of varying degrees of semantic complexity, including both document and data retrieval. A typical vector representation of semantic content is presented in the remainder of this study.

4. Knowledge Representation in Document Retrieval

The basic content identification problem is the same in document retriev as in other information processing environments: appropriate content identifi and relationships between them must be chosen to represent the information it under consideration. In document retrieval the identifiers may be objective terms, representing for example author names, title words, publishing data, a so on; alternatively the terms may be content descriptors used to represent document content. The content description is often subjective and the number of possible descriptors may be very large, running to several thousand in certain subject areas.

Consider a collection D of stored items D_i, and a set of t distinct descriptors or identifiers A_k representing the information content in D. For each descriptor A_k, it is possible to define a number of specifying characteristics as follows:

a) A value a_{k_i}, chosen from a number of different values $a_{k_1}, a_{k_2}, \ldots, a_{k_{n_k}}$ may be assigned to each record D_i for which descriptor A_k applies; these values might for example represent individual dates of publication, or languages of publication, or book measurements.

b) A weight w_{k_i} may be used in each record to represent an importance factor for descriptor A_k, the assumption being that not all identifiers are equally important in all records.

c) A role indicator r_{k_i}, chosen from a number of possible indicators $r_{k_1}, r_{k_2}, \ldots, r_{k_{p_k}}$, can be used to specify a syntactic or semantic specification for descriptor A_k in D_i; the role indicators could represent normal case labels.

d) A link indicator b_{k_i} chosen from a number of possible indicators $b_{k_1}, b_{k_2}, \ldots, b_{k_{q_k}}$, can serve to denote relationships between descriptor A_k in D_i and one or more other descriptors of D_i.

A standard indexing operation, consisting in the assignment to the stored records of descriptors or identifiers chosen to represent information content, will then produce for each stored record an index vector of the following kind [12]:

$D_i = (a_{i1}, w_{i1}, r_{i1}, b_{i1}; a_{i2}, w_{i2}, r_{i2}, b_{i2}; \ldots\ldots, a_{it}, w_{it}, r_{it}, b_{it})$

where a_{ij}, w_{ij}, r_{ij}, b_{ij} represent the various characterizing indicators for descriptor A_j in record D_i. When a given descriptor A_j is absent from D_i, the values of the corresponding indicator labels may be assumed to be null or zero.

In practice most operational document-retrieval systems operate without links and roles, and value indicators are used for the objective identifiers only and not normally for the content terms. This reduces the index vectors to term weight indicators of the form

$$D_i = (w_{i_1}, w_{i_2}, \ldots, w_{i_t}),$$

where a weight of zero denotes a descriptor absent from D_i.* Simple vector computations can be used to obtain similarity coefficents between pairs of different vectors, representing the degree of similarity between them. [12] Thus given records D_i and D_j, typical similarity functions might be the inner product g of the index vectors, or the cosine h of the angle formed by the vectors in t-space. Specifically,

$$g(D_i, D_j) = \sum_{k=1}^{t} d_{i_k} d_{j_k} \qquad (1)$$

and

$$h(D_i, D_j) = \frac{g(D_i, D_j)}{\sqrt{g(D_i, D_i) \cdot g(D_j, D_j)}} \qquad (2)$$

---

*In some practical systems the situation is further simplified by admitting binary descriptor weights only, restricted to 0 or 1 values.

Vector similarity computations are used to obtain similarity values between incoming queries and stored documents and records, permitting a retrieval discipline in which items are presented to the user in decreasing order of query-document similarity. In that case, a query vector $Q_i = (q_{i_1}, q_{i_2}, \ldots q_{i_t})$ replaces the record vector D_i in the similarity computation of equations (1) or (2). Retrieval of a given item does not then depend on the presence or absence of a particular identifier, but on a global match between the query vectors and the corresponding document vectors.

Similarity computations between pairs of document or record vectors may also be used to generate clustered file organizations in which the records are grouped into similarity classes according to the affinities in the respective term vectors. In a clustered file, classes of similar records can be retrieved efficiently as a single group. [13] Document similarity computations may also lend themselves to an automatic determination of the importance, or weight of individual record identifiers by postulating that a good content term -- one that must receive a high weight -- will help distinguish the individual records in a given collection by decreasing the average pairwise record similarity (while poor content terms will on the contrary increase pairwise record similarity). [12]

Since vector similarity computations are versatile and easy to use for term weight computations, file organization, and retrieval, it is useful to extend their application to complex retrieval environments comprising documents as well as data items.

5. Extended Vector Representation

It was seen earlier that many different retrieval environments are potentially of interest in information processing. A versatile method for information representation is then of special interest because it could lead to a common approach to different retrieval requirements. The previously mentioned vector model may be used in these circumstances.

Consider as an example of a typical document a well-known play by Shakespeare. The principal content identifiers or descriptors of the play might be personages occurring in the play such as, for example, 'Hamlet', 'Claudius', 'Gertrude', 'Polonius', 'Laertes', 'Ophelia', 'Horatio', and so on. An importance indicator, or weight, might be attached to each descriptor as a function of the degree of importance of the character in the play. Thus, the descriptor 'Hamlet' would receive a higher weight than the descriptor representing Hamlet's father Claudius. Structural information may also be attached to each descriptor. For the descriptor 'Hamlet' this could include

a) numeric data such as Hamlet's age or his weight, or height;
b) family relationships such as "son of Claudius", "son of Gertrude";
c) additional relations with other descriptors, such as "friend of Horatio", "lover of Ophelia", "enemy of Laertes", etc.
d) characteristics attached to Hamlet as a person, such as the clothes he wears, and the implements he carries;
e) the actions characterizing Hamlet such as "fights with Laertes", "kills Polonius", etc.
f) other characteristics such as Hamlet's profession ("prince of Denmark"), and so on.

Information of this kind can be specified in a variety of ways. One possibility consists in using concept vectors such as those utilized in the Satin retrieval system.* [14] In Satin each descriptor is characterized by a a set of triplets of the form (X_i, Y_i, Z_i) where the X component represents a logical characteristic in each case, Y is a semantic characteristic corresponding to the logical characteristic X, and Z is a link indicator making it possible to relate the given triplet to additional triplets attached to other descriptors. The logical characteristics are similar in spirit to the predicates of the predicate logic, and are chosen to reflect the subject matter for the collection under consideration. To describe, for example, the properties of individual persons and their relationships, a number of numerical characteristics may be needed such as age, height, weight; the semantic characteristics of the corresponding triplets will then represent the precise values of these characteristics. Additional logical characteristics may represent actions, feelings, objects worn by the person, appearance and posture of the person, and so on.

Consider the following examples where each quadruplet represents a descriptor followed by logical characteristic, semantic characteristic, and link in that order:

a) Hamlet is 28 years old

(Hamlet, 1, 28, _)

The logical characteristic 1 is assumed to represent a numerical "age" property.

*The examples which follow are "Satin-like" but do not purport to present the existing Satin implementation or to describe the scope of that system.

b) Hamlet loves Ophelia and hates Polonius

(Hamlet, 8, subject, 1) (Hamlet, 9, subject, 2)

(Ophelia, 8, object, 1)

(Polonius, 9, object, 2)

Here 8 and 9 represent the feelings of love and hatred respectively and the link indicators 1 and 2 relate corresponding quadruplets.

c) Hamlet carries one sword and Laertes carries two swords and a helmet

(Hamlet, 6, subject, 1)

(Laertes, 6, subject, 3)

(Laertes, 7, subject, 2)

(Sword, 6, object, 1)

(Sword, 6, object, 3)

(Sword, 1, 1, 1)

(Sword, 1, 2, 3)

(Helmet, 7, object 2)

The logical characteristics 6 and 7 represent respectively items carried, and worn by the corresponding subject. Thus the vector pair

(Laertes, 6, subject, 3)

(Sword, 6, object, 3)

implies that "Laertes carries a sword" in view of the common link indicator 3. The additional triplet

(Sword, 1, 2, 3)

says that the number of swords carried by Laertes is equal to 2 (the logical characteristic 1 specifies that the corresponding semantic characteristic is a number (2 in that case)).

It is clear that with the proper choice of predicates (logical characteristics), a wide variety of different characteristic information can be attached to each descriptor, including term weights, attribute-values, and hierarchical and other relationships. There is therefore reason to believe that the composite vector representation may serve to represent many types of records, including commercial files, document collections, paintings, sculptures, and other kinds of artifacts. [15] In each case, an unambiguous or unrelated description can be left to stand on its own, while other descriptor occurrences can be explicated by long lists of characterizing data.

A time coordinate can be added to the representation either by explicit time-stamping of the vector components, or by using chains of successive vector quadruplets to represent the time factor. Inference making capabilities appear also to be possible by adding intensional rules for some of the logical characteristics. [16]

Two important problems are posed by the composite vector representation. The first concerning the automatic generation of this type of content description is beyond the scope of this study. If it should prove impossible to devise procedures for an automatic content description in composite vector form, the usefulness of the representation would be severely impaired. There is hope, however, that language analysis methods may be available for producing at least simple forms of the composite vectors, covering those subject areas which are adequately represented by a small number of logical characteristics.

The second problem arises in query formulation and query-record comparison. When the descriptor characteristics are absent, or ignored, the query-record comparison reduces to the normal vector match operation of equations (1) and (2). When qualifying information is added to the descriptors in a query, the

quadruplet information must however be taken into account. Consider the following sample queries in increasing order of specificity (the notation used corresponds to that introduced in the "query-by-example" system, where the underlining denotes unspecified elements that can be filled arbitrarily [17]):

a) items dealing with Hamlet

(Hamlet, x, y, z)

b) items dealing with Hamlet holding an item

(Hamlet, 6, subject, z), (w, 6, object, z)

c) items dealing with Hamlet holding a pair of swords

(Hamlet, 6, subject, z), (Sword, 6, object, z),

(Sword, 1, 2, z).

Various possibilities suggest themselves for computing the similarity coefficients between queries and stored records of this type. The query components can be used in unweighted form; alternatively, each component can in principle be weighted according to its presumed importance, the weight being determined by the user, or by the system. Furthermore, two components can exhibit a perfect match when all its subfields agree, or the match can be partial when only some of the subfields agree.

Consider a k-component query

$$Q = (q_{11}, q_{12}, q_{13}, q_{14}), (q_{21}, q_{22}, q_{23}, q_{24}), \ldots, (q_{k1}, q_{k2}, q_{k3}, q_{k4})$$

and let w_i represent the weight of the ith quadruplet component. Since each query component has four subfields, the number of matching subfields between a given query component and a record component ranges from 0 to 4. Let n_i represent the number of subfield matches between the ith query component and some component of the record D. The query-record matching coefficient might

then be defined as

$$f(Q,D) = \sum_{i=1}^{k} w_i \cdot n_i \ , \qquad (3)$$

where the assumption is that each query component is successively matched with all record components. This allows a given query component to produce a match with several different record components. The final matching value for a given query descriptor could then depend on the number of quadruplet components included in the record vector for the given descriptor.

Other possibilities consist in allowing only one match between a query component and the set of record components for a given descriptor. The record descriptors could then be weighted according to the number of existing quadruplet components included in the record vectors for that descriptor. Additional possible matching strategies are readily devised.

The detailed implementation problems do not appear to be substantially more complicated in the composite vector system than in many other comparable retrieval environments.

REFERENCES

[1] D. Kroenke, Data Base Processing, Science Research Associates, Chicago, 1977.

[2] C. J. Date, An Introduction to Data Base Systems, 2nd edition, Addison Wesley Publishing Company, Reading, Mass., 1977.

[3] D. L. Waltz, An English Language Question Answering System for a Large Relational Data Base, Communications of the ACM, Vol. 21, No. 7, July 1978, pp. 526-539.

[4] L. R. Harris, User Oriented Data Base Query with the Robot Natural Language Query System, Int. Journal of Man-Machine Studies, Vol. 9, 1977, pp. 657-713.

[5] G. G. Hendrix, E. D. Sacerdoti, D. Sagalowicz and J. Slocum, Developing a Natural Language Interface to Complex Data, ACM Transactions on Data Base Systems, Vol. 3, No. 2, June 1978, pp. 105-147.

[6] N. Roussopoulos and J. Mylopoulos, Using Semantic Networks for Data Base Management, Proceedings of the Conference on Very Large Data Bases, New York, 1975

[7] J. R. Abrial, Data Semantics, in Data Base Management, J. W. Klimbie and K. I. Kofferman, editors, North Holland Publishing Co., Amsterdan, 1974.

[8] R. J. Bachman, What's in a Concept: Structural Foundations for Semantic Networks, Int. Journal of Man-Machine Studies, Vol. 9, 1977, pp. 207-222.

[9] J. W. Sowa, Conceptual Graphs for a Data Flow Interface, IBM Journal of Research and Development, July 1976, pp. 336-357.

[10] G. Silva and C. A. Montgomery, Knowledge Representation for Automated Understanding of Natural Language Discourse, Computers and the Humanities, Vol. 11, 1978, pp. 223-234.

[11] H. J. Schmid and J. R. Swenson, On the Semantics of the Relational Data Model, ACM SIGMOD Conference Proceedings, 1975, pp. 211-223.

[12] G. Salton, A Theory of Indexing, Regional Conference Series in Applied Mathematics No. 18, Society of Industrial and Applied Mathematics, Philadelphia, 1975.

[13] G. Salton and A. Wong, Generation and Search of Clustered Files, ACM Transactions on Data Base Systems, Vol. 3, No. 4, December 1978, pp. 321-3

[14] L. Bourrelly and E. Chouraqui, Le Systeme Documentaire Sat In 1, 2 Volumes, Centre National de La Recherche Scientifique, Paris 1974 and 1978.

[15] R. C. Chenhall, Museum Cataloging in the Computer Age, American Association for State and Local History, Nashville, Tennessee, 1975.

[16] J. Minker, Search Strategy and Selection Function for Inferential Information Retrieval, ACM Transactions of Data Base Systems, Vol. 3, No. 1, March 1978, pp. 1-31.

[17] M. M. Zloof, Query-by-Example: A Data Base Language, IBM Systems Journal, No. 4, 1977, pp. 324-343.

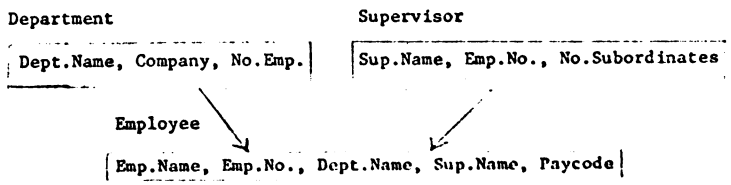
i) Department (Department Name, Company, Number of Employees)

ii) Supervisor (Supervisor Name, Employee Number, Number of Subordinates)

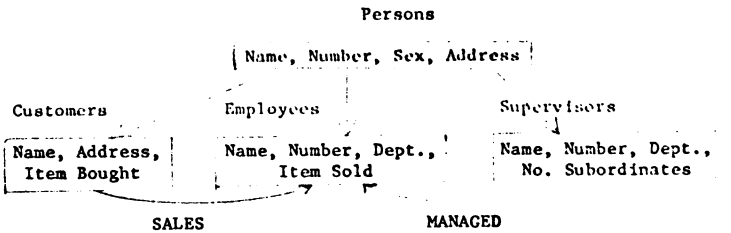
iii) Employee (Employee Name, Employee Number, Department Name, Supervisor Name, Paycode)

a) Entity Sets with Implicit Relationships

(through 'Dept.Name' between (i) and (iii) and 'Supervisor Name' between (ii) and (iii))



b) Entity Sets with Explicit Relations

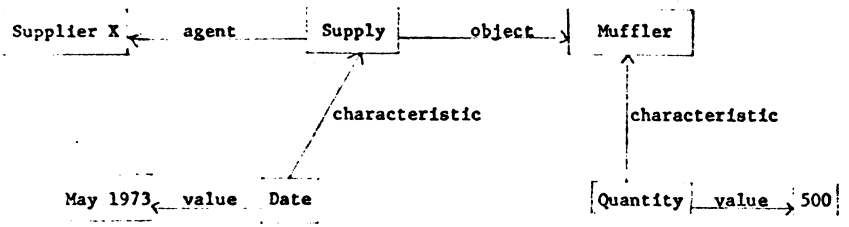


c) Entity Sets with Hierarchical and Contextual Relations

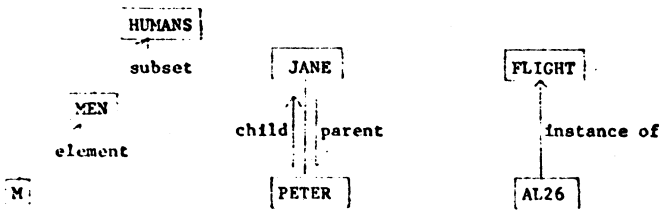
(contextual relation 'SALES' between Customers and Employees, and 'MANAGED' between Employees and Supervisors)

Various Representations of Entities, Attributes and Relationships

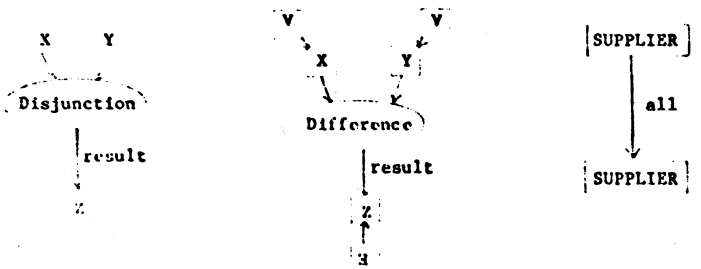
Figure 1



a) Links with "Case" Labels

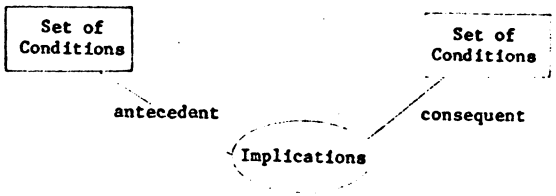


b) Hierarchical Inclusion Relations

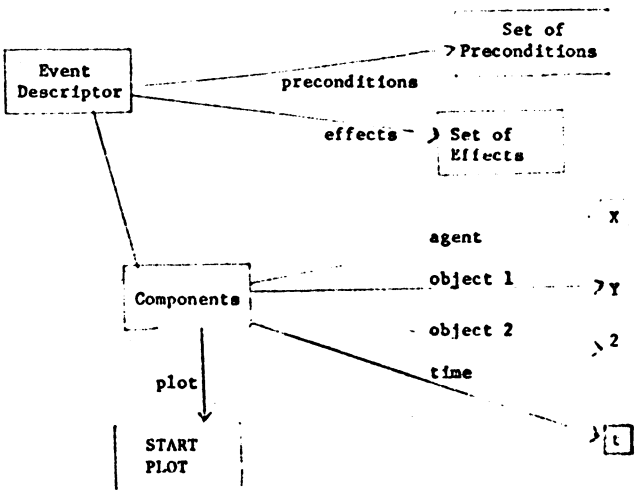


c) Operator Nodes and Quantification

Figure 2



d) Implication Operator



e) Event Description

Semantic Graphs Including Various Types

of Nodes and Links

Figure 2 (cont.)