Using Semantic Networks for Data Base Management

by

Nicholas Roussopoulos

John Mylopoulos

Department of Computer Science
University of Toronto

Abstract

This paper presents a semantic model of data bases. The model assumes the availability of a semantic network storing knowledge about a data base and a set of attributes for the data base. The use of the semantic net in generating a relational schema for the data base, in defining a set of semantic operators and in maintaining the data base consistent is then demonstrated.

This work was partially supported by the Department of Communications of Canada and by the National Research Council of Canada. Authors' address: Dept. of Computer Science, Artificial Intelligence Group, University of Toronto, Toronto, Ontario M5S 1A7, Canada.

1. INTRODUCTION

The usefulness of Data Base Management Systems (DBMSs) is severely restricted by their failure to take into account the semantics of data bases. Although all three models (Hierarchical, Network and Relational) provide a logical view of the data base in terms of data structures and a set of operators on them, they fail to incorporate the semantics of the data base into these data structures and operators.

Some of the problems that are not handled adequately by existing models are listed below. For reasons of economy, we will discuss the relational model only, although similar criticisms apply to the other models as well.

(a) What do attributes and relations mean? Each user must know what the attributes and relations of a relational schema mean, otherwise he cannot use them. The methods that are available for solving this problem (data dictionaries) are in their infancy and are restricted to primary relations only.

(b) How do we choose a relational schema for a particular data base? Some work has been done on this problem using the concept of functional dependency [1,3,7,14]. It has been argued elsewhere [9], and we concur, that this concept is not adequate for expressing the semantic relationships that may exist between items constituting a data base, and that a new, more semantic, approach may be needed.

(c) When do data base operations make sense? Apart from obvious syntactic considerations, the only constraints on the execution of a particular data base operation the current systems can account for are related to cost and security. On the other hand, there are many semantic pointers that could be used to determine whether an operation makes sense or not.

(d) How do we maintain the data base consistent? With the semantics of the data base excluded from the relational model the effect insertions, deletions and updates have on the data base is only understood by the user in terms his/hers subjective view of what the information in the data base means. Thus consistency becomes a subjective notion and this can easily lead to its violation.

Our approach to data base management is based on the availability and use of a semantic network which stores knowledge about the data base being considered. Given this semantic network, we proceed to tackle the problems mentioned above, and others, always refering back to the net whenever a question arises regarding the meaning of the data base.

It should be clear to the reader that any system which uses the semantic approach we are proposing here will be expensive, since it has to account for information about, as well as in the data base. It is our position, however, that many problems data base management faces today will not be solved until the semantics of the data base are included in the designer's as well in the user's viewpoint of the data base.

The semantic model we will develop is in several respects an extension of Codd's relational model [2]. Two first attempts to use the semantics of a data base in order to derive the relational schema in such a way that some consistency constraints can be posed on it are due to Deheneffe et al [6] and Schmid and Swenson [13]. Both papers use a simple-minded representation for the semantics of a data base and provide consistency rules for addition-deletion operations on the data base. Another work that must be mentioned because it is our starting point in this research is the TORUS project whose aim was to provide a natural language front end for a data base management system [9]. In the process of designing and implementing a prototype version of TORUS we have reached many of the conclusions that are presented in this paper.

The paper assumes that a data base is presented in terms of the set of attributes to be used and a semantic network representation of the knowledge defining the meaning of the data base. It then considers some of the problems mentioned earlier, namely the generation of the relational schema, the definition of semantic operators with data base counterparts, and the maintenance of consistency for the data base, demonstrating in each case how the availability of the semantic net can be of use.

Section 2 gives an introduction of the representation we will use for knowledge about a data base. Section 3 considers the generation of the relational schema from the semantic net. Section 4 provides semantic operators and their data base counterparts. Finally, section 5 discusses consistency of data bases and gives four examples to demonstrate the uses of the semantic net regarding this problem.

2. REPRESENTING KNOWLEDGE ABOUT A DATA BASE

In this section we discuss the representation of knowledge that will be used in the rest of the paper. This representation is based on semantic networks as developed by the TORUS project and more complete descriptions of its features and uses can be found elsewhere [9,10,11]. A major extension to the TORUS representation had to be introduced in order to allow it to handle quantification, which is rather important for expressing queries about the data base.

The section consists of two parts. In the first, we introduce the representation and discuss various aspects of its use, notably the generation of context and the integration of new information to the semantic net (graph-fitting). In the second, we describe the representation of quantification that we will use.

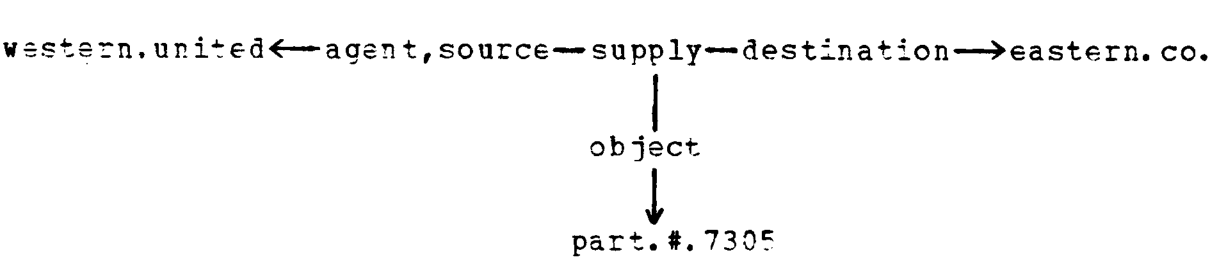
2.1. The Semantic Net and its Uses.

The semantic net is a labelled directed graph where both nodes and edges may be labelled. The labels of nodes will only be used for reference purposes and will usually be mnemonic names. The labels of edges, on the other hand, will have a number of associated semantic properties and inferences.

There are four types of nodes: *concepts*, *events*, *characteristics* and *value-nodes* which are used to represent ideas making up the knowledge related to a particular data base.

Concepts are the essential constants or parameters of the world we are modelling and specify physical or abstract objects.

Events are used to represent the actions which occur in the world. Their representation is based on a case-grammar model, (Fillmore [8]), and consists of an event node and several nodes that specify who plays the *roles* (or fills the *cases*) associated with this event. For example,



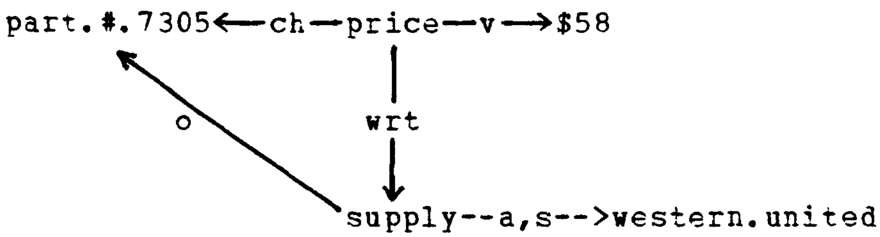
represents an instantiation of the event 'supply' with 'western.united' playing the role of "agent" and "source", 'eastern.co.' playing the role of "destination" and 'part.#.7305' being the supplied part.

The list of cases we will use and their abbreviations has as follows: agent (a), affected (aff), topic (t), instrument (i), result (r), source (s), destination (d) and object (o). The names of these cases are intended to be self-explanatory.

Characteristics are used to represent states (situations) or to modify concepts, events or other characteristics. A characteristic may be considered to be a binary relation mapping elements from its domain -those nodes to which the characteristic may apply -to its range -those values which the characteristic may take. For example, ADDRESS maps LEGAL.PERSON (the set of persons and institutions) into the set of possible address.values. Graphically, a characteristic is represented as a node labelled by the name of the characteristic, with a "ch" ("characterize") edge pointing to an element of the domain and a "v" ("value") edge pointing to the corresponding value:

john.smith←ch—address—v→65 st. george st.,toronto,canada

"True" characteristics are usually natural attributes of concepts but characteristics can also be used as abbreviations of more complicated situations where we wish to omit unnecessary detail. In some circumstances such abbreviations are mappings from a cross-product domain to a range and we use a "wrt" ("with-respect-to") edge to indicate the second argument. For example, PRICE characterizes PARTs with respect to SUPPLY, producing a DOLLAR.VALUE:



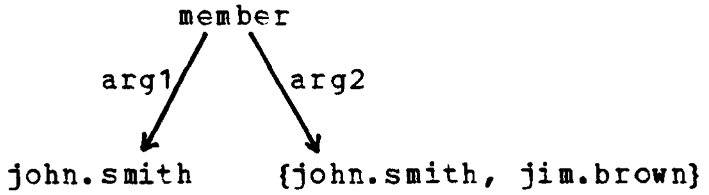
We will distinguish four types of characteristics, depending on the relation defined between the domain and the range of the characteristic: many-to-many, many-to-one, one-to-many and one-to-one. Below we give examples of the four different types, demonstrating the graphical notation we will use for each kind:

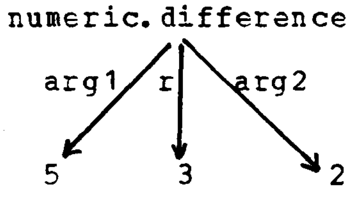
PERSON⇐ch═ADDRESS═v⇒ADDRESS.VALUE
(many-to-many)
PHYSICAL.OBJECT⇐ch═WEIGHT—v→WEIGHT.VALUE
(many-to-one)
PERSON←ch—POSSESSION═v⇒PHYSICAL.OBJECT
(one-to-many)
PART←ch—PART.#—v→PART.#.VALUE
(one-to-one)

Thus a person can have several addresses and at the same time several persons may have the same address, each physical object has a unique weight but a weight cannot be associated to a unique physical object; a physical object is possessed by a unique person but a person does not possess a unique object. Finally, a part has a unique part number and each part number is associated to a unique part.

Value-nodes represent values of characteristics such as an address ('65 st. george st., ontario, canada'), a weight ('65lbs'), a dollar value ('$53.7C'), a name ('john smith') etc.

In addition to these types of entities, we will sometimes use mathematical predicates and functions such as SET.MEMBER, SET.DIFFERENCE, NUMERIC.DIFFERENCE etc. Two examples of such nodes, and the types of edges we associate to them, are given below:





The "r"-labelled edge is the result edge that is also used for events.

The nodes that constitute the semantic net will be divided into two classes: one, relating to generic concepts, events, characteristics and value-nodes describes the possible or allowable states of affairs in our domain of discourse. This class we will informally call the "upstairs" of the semantic net, in contrast to the second class, its "downstairs", where we keep instantiations or particular occurances of ideas. Note that each generic node can be thought of as the possibly empty or possibly infinite set of its instantiations. Similarly, each instantiation can be thought of as a (conceptual) constant. "Upstairs" nodes will have their names given in capital letters whereas "downstairs" ones will have their names given in small letters. For example, in

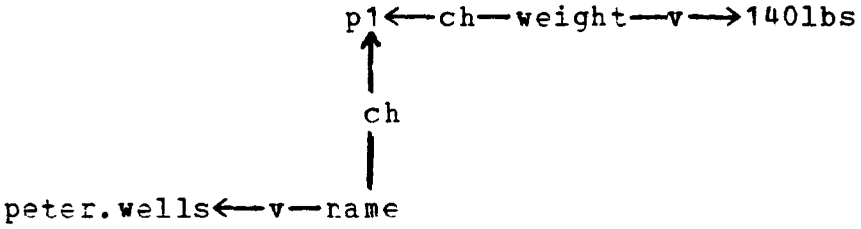
PHYSICAL.OBJECT⇐ch═WEIGHT—v→WEIGHT.VALUE

the nodes are generic and the fact described by this graph is "physical objects can have a weight whose value is an instantiation of the generic node WEIGHT.VALUE; moreover the relation between PHYSICAL.OBJECT and WEIGHT.VALUE is many-to-one. On the other hand,

peter.wells←ch—weight—v→140lbs

specifies that the instantiation 'peter.wells' has weight '140lbs'. This graph could be meaningless if the item 'peter.wells' is not recognized as an instantiation of PHYSICAL.OBJECT, and '140lbs' as an instantiation of WEIGHT.VALUE. Thus structures which include generic nodes serve in a certain sense as templates that must be matched by structures that consist of instantiations only, if the latter are to be meaningful to the semantic network.

In the representation of 'peter.wells weighs 140lbs' we have introduced a simplification that we intend to use throughout this paper: we have named the node that represents the person named 'peter.wells' with the name 'peter.wells'. A more complete representation of this would have been



where p1 is an arbitrary identifier. In general, when we have one-to-one characteristic for a certain class of concepts we will often omit this characteristic from the representation altogether and we will use the value-nodes associated to that characteristic as replacements for the characterized concepts. This way, assuming that NAME is an one-to-one characteristic, we replace the structure

p1←—ch—name—v—→peter.wells

by a single node labelled 'peter.wells'.

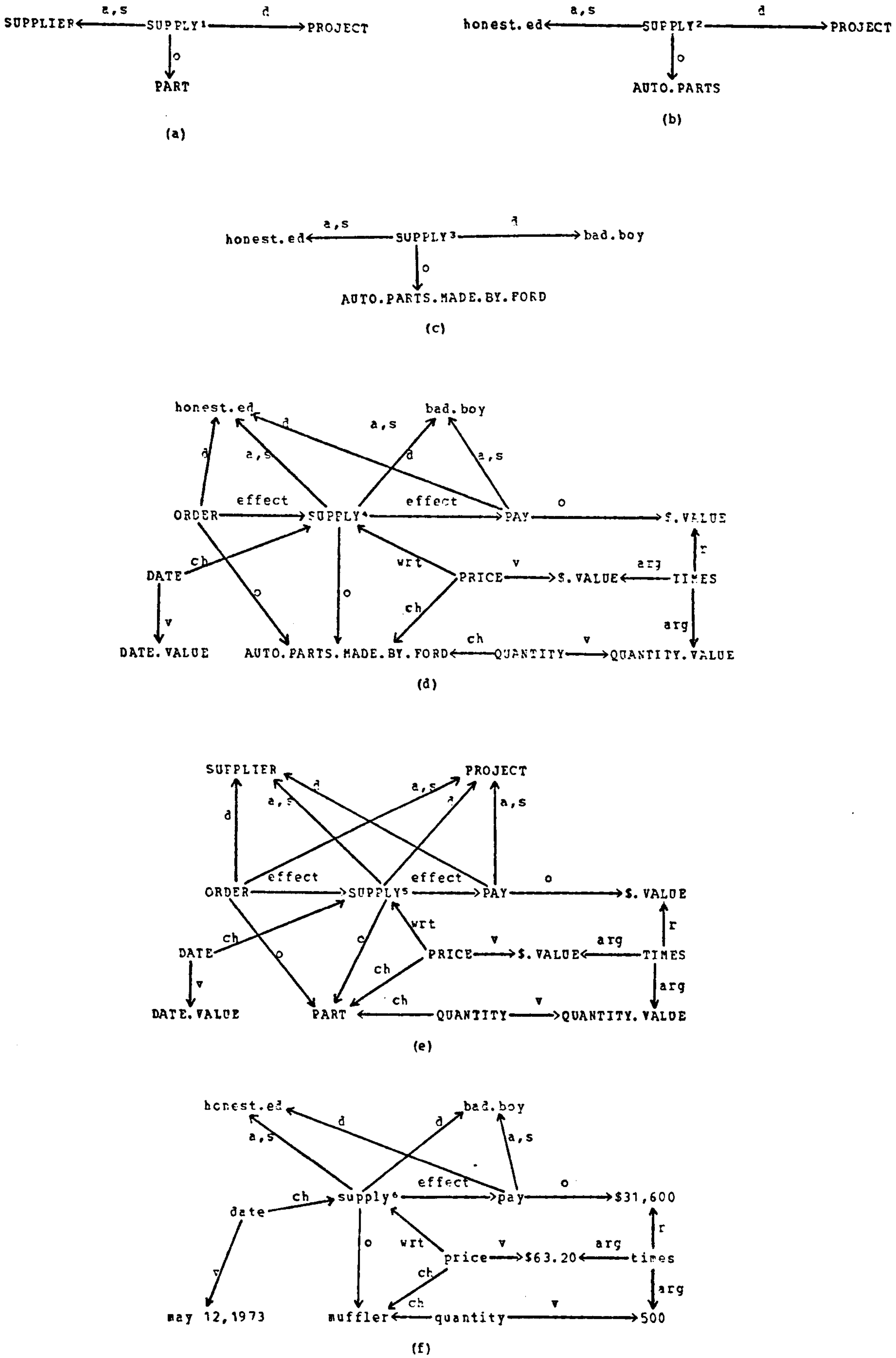
The apparatus we have described so far is sufficient for the representation of most isolated phenomena, but we need the ability to represent larger chunks of knowledge. We achieve this by introducing scenarics.

A "scenario" is a collection of events, characteristics and mathematical predicates related through causal connectives such as "prerequisite" ("prereq") and "effect".

One may regard a scenaric as a pattern or template which when matched by a structure, causes various kinds of inferences and predictions to be made. Moreover, only structures which are matched by some of the scenarios on the semantic net are meaningful to the system. Consider, for example, the notion of 'suppliers supply projects with parts', which we can represent as shown in fiq. 2.1(a). This is a general scenario that will be matched by any instantiation of supply if the latter is to make any sense at all. Another scenario that involves 'supply' is shown in fig. 2.1(b) and represents the meaning of 'honest.ed supplies auto.parts' which means that 'honest.ed' is willing/equipped /in contract to supply AUTO.PARTS Note that some project has to be assumed as the destination of such 'supply' actions as well. Another 'supply'-related scenario is given in fig. 2.1(c) and means 'honest.ed supplies bad.boy with auto.parts.made.by.ford'. Again the 'supplying' is supposed to be taking place on a regular basis, possibly after a mutual agreement. Yet another scenario related to 'supply' involves particular cases where 'honest.ed supplies bad.boy with a certain quantity of parts on a certain date'. Fig. 2.1(d) shows the scenario for this situation and notes the effects of any such 'supply' action has: the parts must have been ordered by 'bad.boy', and 'bad.boy' must pay 'honest.ed' because the latter supplied the parts. This is a partial instantiation of a more general scenario shown in fig. 2.1(e). Finally, fig. 2.1(f) shows a particular instantiation of the 'supply' event of fig. 2.1(d), which may correspond to a statement such as 'honest.ed supplied bad.boy on may 12, 1973 with (a quantity of) 500 mufflers at the price of $63.20 each and that he received a total of $31,600.00'.

In fig. 2.1 we presented six different scenarios or instantiations of scenarios that are obviously related semantically. We will now describe the overall organization of the semantic network, in other words how are all these scenarios put together to form the semantic network. This organization will be defined in terms of "axes" or "dimensions".

The first axis we will discuss is called "SUB" because it is based on the subset (set-theoretic containment) relation. We will say that node X is a SUBnode of node Y if the set of



(a)

(b)

(c)

(d)

(e)

(f)

fig. 2.1

instantiations of X is a subset of the set of instantiations of Y. The SUB relation between X and Y will be denoted by

$$Y \xrightarrow{\text{sub}} X$$

or simply

$$Y \longrightarrow X$$

If X is downstairs, the relation between Y and X is one of "instantiation" or "example-of". We will continue to use an unlabelled edge to denote such relations, since the fact that X is downstairs is already specified by its name (small letters). Fig. 2.2 shows a portion of the SUB axis for concepts that may be related to a Suppliers-Projects-Parts data base.

In general, we can organize (partially order) the concepts occuring in our domain of discourse into a hierarchy representable by its Hasse diagram. It is important to note that (semantic) properties of concepts are inherited along the SUB axis. For example, since SUPPLIERs are COMPANYs which are INSTITUTIONs, which are LEGAL.PERSONs, and since any LEGAL.PERSON can have an ADDRESS, a SUPPLIER can have an ADDRESS. This property of the SUB axis is a very important memory-saving device.

Scenarios are also organized on the SUB axis. Thus the six structures of SUPPLY given in fig. 2.1 can be organized as shown in fig. 2.3. It should be noted that cases or other characteristics of events which are not explicitly represented on the net are inherited from its lowest super-event that fills those cases or characteristics. The reader should satisfy him/herself that indeed the SUB relations do hold between the various SUPPLY nodes, as claimed on fig. 2.3. It must also be noted that for an event E with cases C1, C2,...,Cn to be placed below another event E' with cases C1', C2',...,Cn' on the SUB axis, it must be that E is a subset of E', but also Ci is a subset of Ci' for $1 \leq i \leq n$.

Another important axis is the "DEF(initional)" one. Let us go back to the scenario of fig. 2.1(e) and the SUPPLY^5 node present there. Here we are obviously talking about a sequence of events that starts when a SUPPLIER begins to make arrangements to SHIP PARTs to a PROJECT and ends when the latter receive them. Thus the scenario of fig. 2.1(e) is semantically ambiguous since it does not specify what does DATE refer to, the date the shipment is made or the date it is received. In order to define how does one SUPPLY^5 (something) and what does DATE refer to, we use the DEF axis. Fig. 2.4 shows the scenario that defines SUPPLY^5 in terms of the events SHIP and RECEIVE. The figure shows how are the cases of SUPPLY^5 related to cases in the scenario, but also how is DATE defined (here we define it as the date on which the shipment was made).

In general, the DEF axis enables us to give more details about events and characteristics.

Concepts can also be defined in terms of scenarios which specify the roles of those concepts. For example, PARTS.MADE.BY.FORD is defined as the concept filling the object case of the event MANUFACTURE whose agent case is filled by 'ford'. This definition of AUTO.PARTS.MADE.BY.FORD is indicated

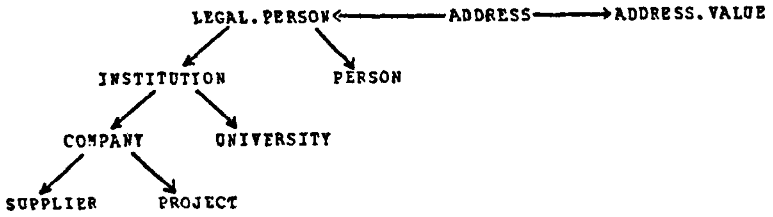


fig. 2.2

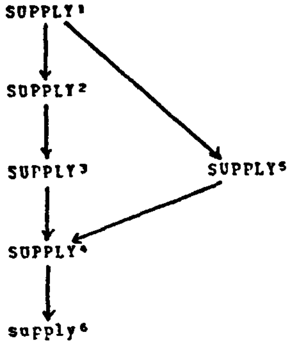


fig. 2.3

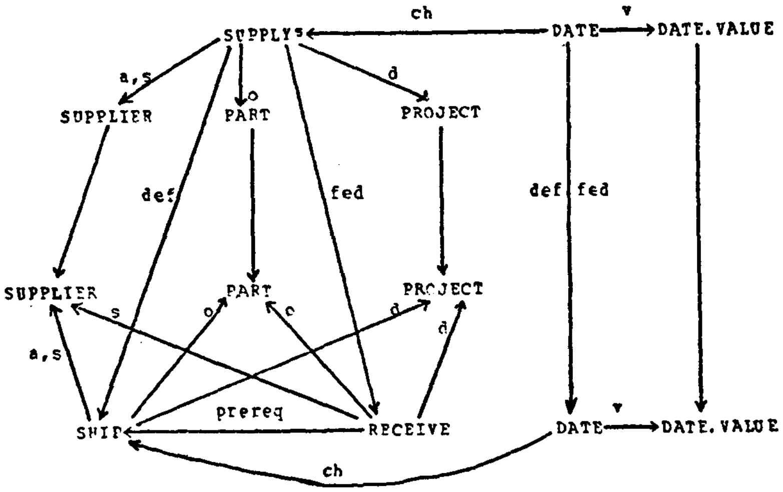


fig. 2.4

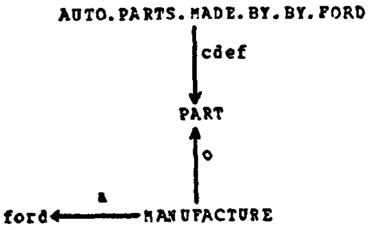
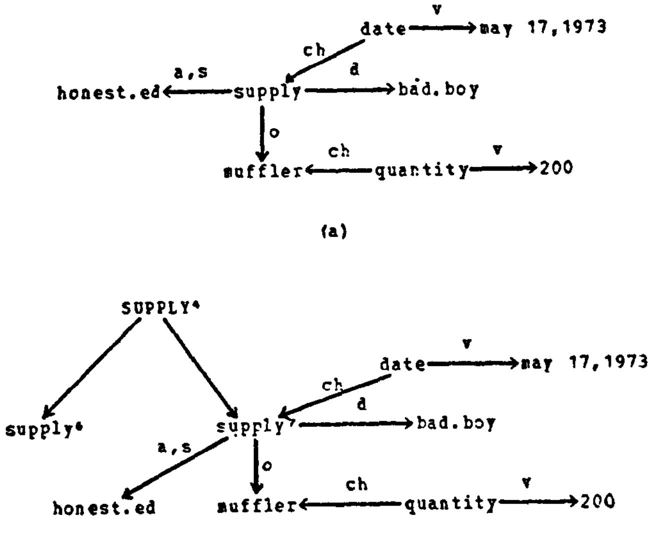


fig. 2.5



(a)

(b)

fig. 2.6

on the net by a "cdef" labelled edge, see fig. 2.5. The role of concept definitions is very important for the so called "membership problem". In other words "cdefs" denote the sufficient and necessary conditions for membership in a particular class. In the example of PARTS.MADE.BY.FORD, a part belongs into this class iff it has been manufactured by 'ford'.

Finally, another edge which defines an axis is the "part" edge (a DEPARTMENT is "part" of a COMPANY, a WHEEL is "part" of a CAR, etc.)

Representing knowledge on the semantic network has the advantage that this information can be examined and reasoned about provided that there is an appropriate interpreter. On the other hand, this representation is expensive and for any universe of discourse there will be "peripheral" knowledge for which general reasoning may not be necessary. We will represent such knowledge in terms of functions which we associate to corresponding nodes on the semantic net.

Some of these functions we will call "recognition functions" because their job is to recognize instances of a class by using syntactic or semantic information. For example, dates can be recognized by syntactic string matching rules while the "cdef" axis has to be used in order to determine whether or not a particular part belongs to the class of AUTO.PARTS.MADE.BY.FORD. Value-nodes in general do have associated recognition functions. "Mapping functions" are useful for mapping structures from one level of the representation to another. For example, mapping functions may be used to replace every instantiation of SUPPLY by instantiations of SHIP and RECEIVE so that there is no need for the explicit DEFinition of SUPPLY on the semantic net. "Definitional functions" are used to define procedures for performing particular actions (RETRIEVE all tuples that satisfy a given description, UPDATE something in the data base, MOVE a block, etc.). The nodes of the net that have associated definitional functions will have their names preceded and followed by *'s. For example,

system←a—*retrieve*—o→?←v—part.#—ch→muffler

the function "retrieve" will perform the retrieval of the part number of the part 'muffler' and it will replace the question mark by this value.

It is important to stress that knowledge can be represented in either procedural or declarative form and which form is used is strictly an issue of trading cost for "understanding power".

We turn our attention now to some uses of the semantic network in accomplishing "understanding". There are two uses we will discuss: the generation of "context" during a dialog and the "integration" of new information to the already existing semantic network (graph-fitting). We discuss these uses partly to give some justification for the representation we have described so far, and partly because some aspects of these uses are closely related to semantic problems of data bases (see sections 4.3 and 5).

The presence of a network entity in the context represents the system's expectation that this item is or will be relevant to the current dialogue. When new information, which has been predicted, enters the dialogue, its relevance can be explained by the "generation path" taken to create the expectation. Consider, for example, the statements:

'honest.ed send out a shipment yesterday.'

and

'there were 300 snow-tires and 50 mufflers.'

Here, we can generate part of the context of SHIP when the first sentence is "understood". Part of this context is the event of SUPPLY^5 according to the scenario of fig. 2.4. Once SUPPLY^5 with 'honest.ed' as agent-source is in, the object case of this SUPPLY^5 (i.e. AUTO.PARTs) also enters the context. When the second statement is presented, it can be "understood" in terms of the existing context, since both 'snow-tires' and 'mufflers' are AUTO.PARTs. By "understood" we mean here that an interpreter can infer what is the relationships of the sentence to what was said before.

In generating the context one has to take into account the semantics of the various edge labels. To give an example, whenever we have the configuration

A——effect——>B

every instantiation of A implies strongly an instantiation of B, while every instantiation of B implies weakly an instantiation of A. This means that when a node enters the context, it has a "strength" value attached, which specifies how reliably it can be inferred from the already existing context. More information on the context mechanism can be found in [10].

A part of the procedure for integrating new input to the semantic network will have to be done by an algorithm which we call "graph-fitting". Assume that the semantic network includes the scenarios of fig. 2.1 and that the the new sentence

'honest.ed supplied bad.boy with 200 mufflers on may 17,1973'

is presented to the system. The system's job is to construct the graph of fig. 2.6(a) representing the meaning of this sentence, and then to integrate this graph with the semantic network (fig. 2.6(b)). To accomplish that, the graph-fitting algorithm may start from the most generic SUPPLY^1 node, making sure that all the cases of the input 'supply' may be placed below the cases of the generic SUPPLY^1. Once this has been accomplished, it may try to see whether there are any SUPPLY events below the generic one which are matched by the input 'supply'. The scenario of fig. 2.3(b) is chosen and a test is again performed to make sure that the input 'supply' in fact matches the SUPPLY^4 already on the net. This process is repeated until it is no longer possible to move the input graph any further down along the SUB axis. A portion of the net resulting from the integration is shown in fig. 2.6(b).

Note that if there is a context at the time the above sentence is presented for integration, with a SUPPLY node on it, the graph-fitting algorithm will begin with that SUPPLY rather than the most generic one since the lower is the most relevant and an instantiation of it is expected.

2.2 A representation for quantification.

In this section we present an extension of the TORUS representation which allows the representation of simple quantified statements. TOPUS avoided this issue because of its complexity and because primitive types of quantification can be handled by other means, as we will see below. However, many queries to a data base management system such as

'Give me all suppliers who supply all auto-parts to all projects located in Houston'

obviously involve many nested (universal) quantifiers. The need to be able to represent the meaning of such queries has forced us to consider the problem of quantification.

The semantic network, as we described it so far, can handle some aspects of quantification. For example, statements such as

'Every supplier is a company'

and

'Every supplier supplies some parts to some projects'

can be handled through "sub" and case edges respectively.

Consider now the statements

s1: 'suppliers who supply all parts to some project'

and

s2: 'projects that are supplied all parts by some supplier'

Clearly, their meaning is different as they can be represented by the following statements in pseudo-Predicate Calculus notation:

s1': (s $\in$ SUPPLIEP)(all p $\in$ PART)(some pr $\in$ PROJECT)
SUPPLY(s,p,pr)

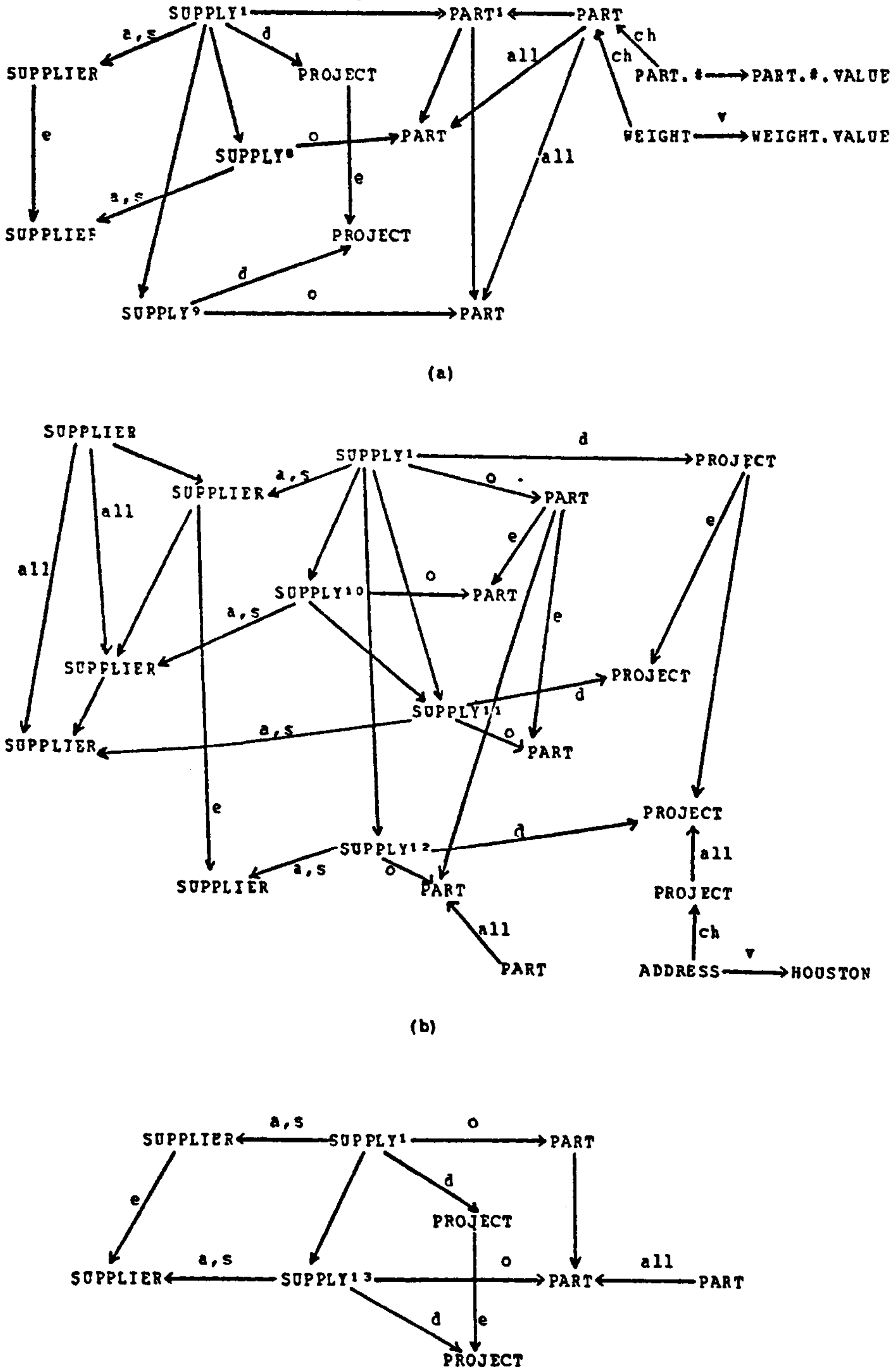
s2': (pr $\in$ PROJECT)(all p $\in$ PART)(some s $\in$ SUPPLIER)
SUPPLY(s,p,pr)

Thus the difference in meaning hinges on which argument of SUPPLY is being quantified. We will represent these statements as shown in fig. 2.7(a), (SUPPLY^{8} and SUPPLY^{9} respectively), where "all" and "e" are new edge labels that are used to specify universally and existentially quantified variables outside the scope of universal quantifiers. Note that the statements, as given here, make no claim about the existence of any suppliers for s1 and projects for s2 that satisfy s1 and s2 respectively.

We can proceed now to represent the meaning of

s3: 'parts that are supplied by all suppliers to some projects'

s4: 'projects supplied some parts by all suppliers'



(a)

(b)

(c)

fig. 2.7

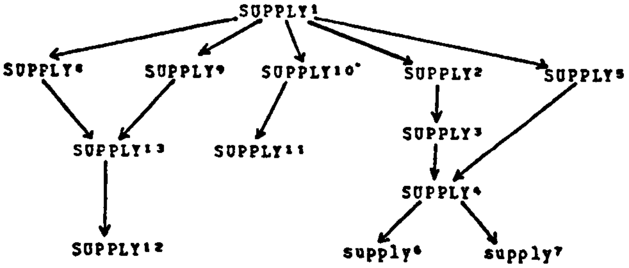


fig. 2.8

and

S5: 'suppliers supplying all parts to all projects in Houston'

with the structure of fig. 2.7(b) ($SUPPLY^{10}$, $SUPPLY^{11}$ and $SUPPLY^{12}$ respectively). Note that some of these sentences are ambiguous. For example, the meaning of s1 could have been represented by the statement

s1": (s ∈ SUPPLIER) (some pr ∈ PROJECT) (all p ∈ PART) SUPPLY(s,p,pr)

and the corresponding semantic representation would have been what is shown in fig. 2.7(c). As far as this discussion is concerned, we are only interested in making sure that both meanings can be represented and not in developing disambiguation algorithms.

The partial ordering of all SUPPLY events mentioned so far is shown in fig. 2.8.

Introducing quantification into our representation is not merely a problem of defining a graph-theoretic notation for it. One has to make sure that the semantic properties of quantification are also inherited by this new representation. We briefly describe some of these semantic properties of the representation we just presented in section 5. Here we wish to stress that we have only made a first step that will help us handle simple cases of quantification. No claims are made about a complete solution to the problem.

3. GENERATING THE RELATIONAL SCHEMA

The first attempts to generate algorithmically the relational schema for a data base are described in [7,14,1]. These papers start with functional dependency as the primitive in terms of which the semantics of a data base are to be described, and provide algorithms which generate from the set of functional dependencies among the attributes, a functional schema in 3rd normal form. In [13], on the other hand, the authors argue, convincingly, that the concept of functional dependency is not sufficient for the expression of all semantic information about a data base and they choose a different set of semantic primitives. These primitives are "independent objects", "characteristics" and "associations" and they have been inspired by the effect of insertions, deletions and modifications on a data base. This method of representing semantics runs into difficulties, however, when a situation arises where an item, such as TRAINING.PROGRAM, can be viewed simultaneously as an independent object and as a characteristic of another independent object, say EMPLOYEE. If TRAINING.PROGRAM is considered as an independent object, then deletion of an instance of it has the effect that the information that some employees had been trained by this program is lost. On the other hand, if it is considered as a characteristic of EMPLOYEE, then the model cannot express other properties of

TRAINING.PROGRAM which are not dependent on EMPLOYEE, i.e. DURATION, PROGRAM.DESRIPTION, etc.

We base the generation of the relational schema on the semantic net that stores the semantics of the data base so that there exists a natural correspondence between the relations of the schema and the nodes of the semantic net. We are assuming that data base attributes are associated with nodes of the net whose names are enclosed in slashes (e.g. /PART/) and that this association is given along with the semantic net. Note that nodes below data base attributes are also data base attributes even if their names are not enclosed in slashes.

A methodology for the generation of the relational schema from the semantic net is given below. Keys are not used in our model because the information conveyed in the keys is implied by the different types of relations that are available in our model.

The relations in the data base correspond to either concepts or semantic relationships between concepts, such as the "part" relationship, and relationships that involve an event or a characteristic. Thus, there are four basic types of data base relations, named "concept", "part", "event" and "characteristic" respectively. The relations in the data base are associated with a corresponding concept, event or characteristic node on the net and store either collections of instantiations of concepts, events and characteristics or collections of generic concepts, events and characteristics. The nodes which are associated with data base relations are called "realized".

Note that characteristic relations can be one-to-many, many-to-one and many-to-many, but not one-to-one. One-to-one characteristics are mapped onto attributes in the relation of the concept, event or other characteristic which they characterize.

The four types of relations used in our model are:

A. Concept-relations correspond to concept nodes of the net which are data base attributes. Their names are identical to the names of the concepts to which they are associated with and have as attributes the concept itself and the names of the value-nodes of their one-to-one characteristics which are data base attributes. For example, the concept /PART/ on the network, fig. 3.1, is mapped onto the relation

PART(PART, PART.#.VALUE, WEIGHT.VALUE)

in the data base. The PART concept on the net is underlined as an indication that this concept is realized. Note that the attribute PART in the above relation stands for PART.NAME.VALUE, while the relation named PART stands for the concept PART. As mentioned in section 2.1, the two nodes have been identified on the net.

B. Part-relations correspond to "part" relationships between data base attributes of the net. Their names are identical to the containing concept name and have as attributes the names of both containing and contained concepts. For

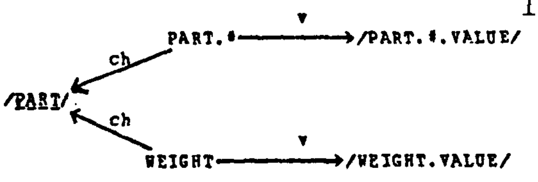


fig. 3.1

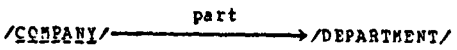


fig. 3.2

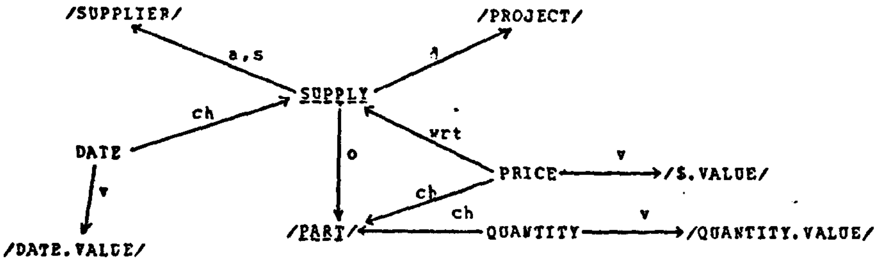


fig. 3.3

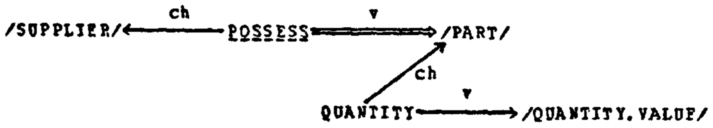


fig. 3.4

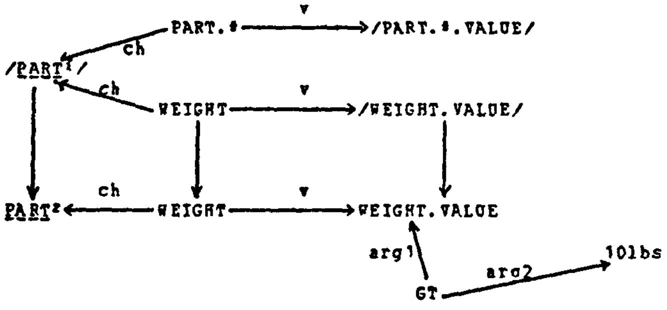


fig. 4.1

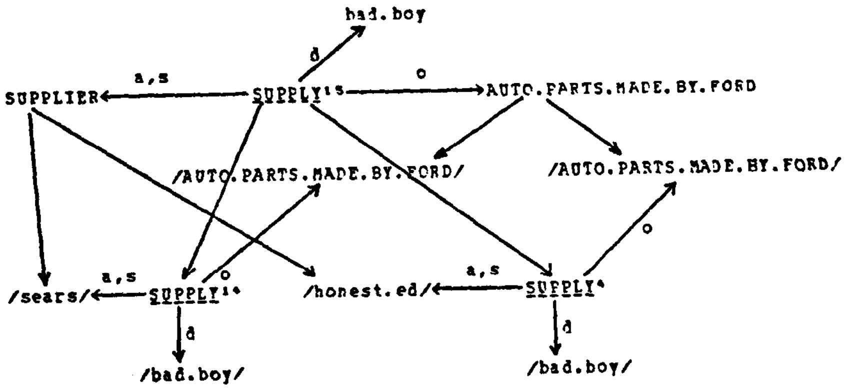


fig. 4.2

example, consider the concept /COMPANY/ in fig. 3.2. The "part" relationship is mapped onto the data base relation:

COMPANY(COMPANY, DEPARTMENT)

Again the attribute COMPANY stands for COMPANY.NAME.VALUE, while the relation named COMPANY stands for the concept node COMPANY. The concept COMPANY, (underlined), indicates that there is a data base relation where the instances of the relationship "part" are stored. Note that the containing concept also has a concept data base relation associated with it to store its one-to-one characteristics and that there is one part-relation for each "part" relationship of it.

C. Event-relations correspond to event relationships among data base attributes of the net. Their names are identical to the names of the events to which they are associated and have as attributes the names of their case-nodes, the names of the value-nodes of their one-to-one characteristics and the value-nodes of one-to-one characteristics of their cases which are not inherited from supernodes. For example, the event SUPPLY on the net, (fig 3.3), is mapped onto the relation:

SUPPLY(SUPPLIER, PROJECT, PART, DATE.VALUE,
QUANTITY.VALUE, $.VALUE)

in the data base. The SUPPLY event node on the net is underlined as an indication that this object is realized and that if a supplying action is requested, it can be retrieved from the SUPPLY relation in the data base.

D. Characteristic-relations. There are three different kinds of characteristic relations to account for the three different types of mappings, many-to-one, one-to-many and many-to-many. Their names are identical to the characteristic nodes and have as attributes the concepts they characterize, the names of value-nodes of one-to-one characteristics of their cases which are not inherited from supernodes, the names of their value-nodes, and the names of the value-nodes of other one-to-one characteristics characterizing the characteristics themselves. Consider the semantic net of figure 3.4. This is mapped onto the data base relation:

POSSESS(SUPPLIER, PART, QUANTITY.VALUE)

which is associated to the node POSSESS.

As one can see, concepts can be relations and/or attributes. Below we give an example where a concept is a relation and an attribute at the same time. Consider the network of fig. 3.3, where the concept /PART/, (at the bottom of the diagram), is one of the attributes in

SUPPLY(SUPPLIER, PROJECT, PART, DATE.VALUE,
QUANTITY.VALUE, $.VALUE)

and has a certain value domain. At the same time, the relation

PART(PART, PART.#.VALUE, WEIGHT.VALUE)

which corresponds to the same /PART/ node is a concept relation that stores the values of the domain of the attribute PART in SUPPLY along with its one-to-one characteristics inherited from node /PART/ in fig. 3.1.

The partial ordering of realized nodes by the SUB edge reflects a partial ordering onto the data base relations which correspond to those nodes. Given a relation r associated with a node n of the net, we will use the terms "superrelation", "subrelation" to specify relations r1 and r2 which are associated to nodes n1 and n2 respectively such that

$$n1 \longrightarrow n \longrightarrow n2$$

Contrary to Codd's view of the relational scema as a "flat" collection of independent relations, [5], the semantic network organizes relations in a hierarchy which explicitly states the semantic relationships among relations. This enables the model, as we will see later, to maintain consistency of primary and derived relations.

Our method for the generation of the relational schema is based on the primitive blocks for building the semantic net (concepts, events and characteristics). The justification for using it is that since those primitives are the smallest semantic entities accessable in our representation, they are also natural units for semantic operations that correspond to data base insertions, deletions and modifications. On the other hand, there may be other criteria that should be taken into account in the process of generating the schema. Thus, it may be that scenarios should also serve as semantic blocks in terms of which the relational schema is constructed.

4. OPERATIONS ON DATA BASE RELATIONS

As suggested in the introduction, the operations allowed by a model must be ones it can account for. In other words, the operations and their results must be explained (interpreted) in terms of the primitives provided by the model. It follows from this premise that for our semantic model we must provide "semantic operators", in contrast to the data base operators defined by the relational model [2]. By a "semantic operator" we mean here an operator which takes as arguments (operands) one or more nodes of the net and constructs a new node or nodes related semantically to those it was obtained from.

Since some nodes on the net have associated relations or attributes of the data base, a semantic operator may have a corresponding data base operation. It is important to stress, however, that in our model the starting point for the definition of operators is the semantic net not the data base. The data base operators are defined by studying the effect semantic operators must have on the data base.

All semantic operators we will define are set-theoretic in nature and can be directly related to manipulations of the SUB axis.

The definition of the semantic operators are given informally in section 4.1. As part of each definition, we give an English expression of the semantic operator. It must be noted that we do so for the reader's convenience in understanding the meaning of the operators. We do not assume the existence of a natural language analyzer for our model. The data base operators we will use can be defined algebraically, as in [4], but this will not be done in this paper.

Section 4.2 describes when and how is a data base operation executed as a result of the execution of a corresponding semantic operation. Section 4.3 considers when is a semantic operation "legal" and whether there is always a corresponding data base operation.

4.1. The semantic and their corresponding data base operators

a. Selection

The semantic operator of selection on a node n consists of creating a subnode below n which has more restricted semantic properties than node n. For example, the expression

'parts which have weight greater than 10lbs'

operates on node PART^1 and results in node PART^2 of fig. 4.1. The data base operator of selection is defined as the selection of tuples of a relation according to certain condition(s) on one or more attribute value(s) and results in a subrelation of the operand relation. Returning to our example, if selection is applied to relation PART^1 associated to node PART^1 it results in a relation PART^2 in the data base and it is associated to node PART^2 of fig. 4.1.

b. Union

Union operates on two nodes n1 and n2 and results in a new node nr which

i. is below every node n that is above n1 and n2
ii. is above n1 and n2
iii. inherits all common characteristics and/or cases of n1 and n2.

For example

'cases of supplying auto.parts.made.by.ford carried out by honest.ed or sears with bad.boy as destination'

operates on the two SUPPLY^4 and SUPPLY^{14} nodes on fig. 4.2 and results in node SUPPLY^{15}, also shown on the figure.

The corresponding data base operator of union takes as arguments two relations associated with nodes n1 and n2 respectively and creates a new relation which is associated with nr. Its attributes are those of the operand relations that correspond to the common characteristics and/or cases of n1 and n2. Thus the new $\underline{\text{SUPPLY}}^{15}$ relation obtained from the union of

$\underline{\text{SUPPLy}}^{4}$(honest.ed, bad.boy, AUTO.PARTS.MADE.BY.FORD)

and

$\underline{\text{SUPPLY}}^{14}$(sears, bad.boy, AUTO.PARTS.MADE.BY.FORD)

is

$\underline{\text{SUPPLY}}^{15}$(SUPPLIER, bad.boy, AUTO.PARTS.MADE.BY.FORD)

c. Intersection

Intersection operates on two nodes n1 and n2 and results in a new node nr which

i. is above every node that is below n1 and n2
ii. is below n1 and n2
iii. inherits all characteristics and/or cases of n1 and n2.

For example,

'parts that have been ordered by some project and possessed by some supplier'

operates on nodes PART^{3} and PART^{4} of fig. 4.3 and results in node PART^{5} also shown on the figure.

The corresponding data base operator of intersection takes as arguments two relations associated with nodes n1 and n2 respectively and creates a new relation which is associated with nr. Its attributes are those of the operand relations that correspond to the characteristics and/or cases of nr. In the above example, the new relation $\underline{\text{PART}}^{5}$, created from the intersection of $\underline{\text{PART}}^{3}$ and $\underline{\text{PART}}^{4}$, has the same form as $\underline{\text{PART}}^{3}$ and $\underline{\text{PART}}^{4}$ and is associated with node PART^{5}.

d. Difference

Difference operates on two nodes n1 and n2, (n1-n2), and results in a new node nr which

i. is below n1
ii. is connected with n2 by an edge pointing to it and labelled "none"
iii. inherits all characteristics and/or cases of n1.

For example,

'parts that no supplier possesses'

operates on PART^{1} and PART^{4} of fig. 4.4 and results in node PART^{6} also shown in the figure.

The corresponding data base operator takes as arguments two relations r1 and r2 associated with nodes n1 and n2 respectively, and creates a new one rr which is a subrelation of r1. The new relation is associated with nr and has as attributes those of r1. In the above example the difference of $\underline{\text{PART}}^{1}$ and $\underline{\text{PART}}^{4}$, ($\underline{\text{PART}}^{1}$-$\underline{\text{PART}}^{4}$), will result in a relation $\underline{\text{PART}}^{6}$ which has the same attributes as $\underline{\text{PAFT}}^{1}$ and is associated with the node PART^{6} of fig. 4.4.

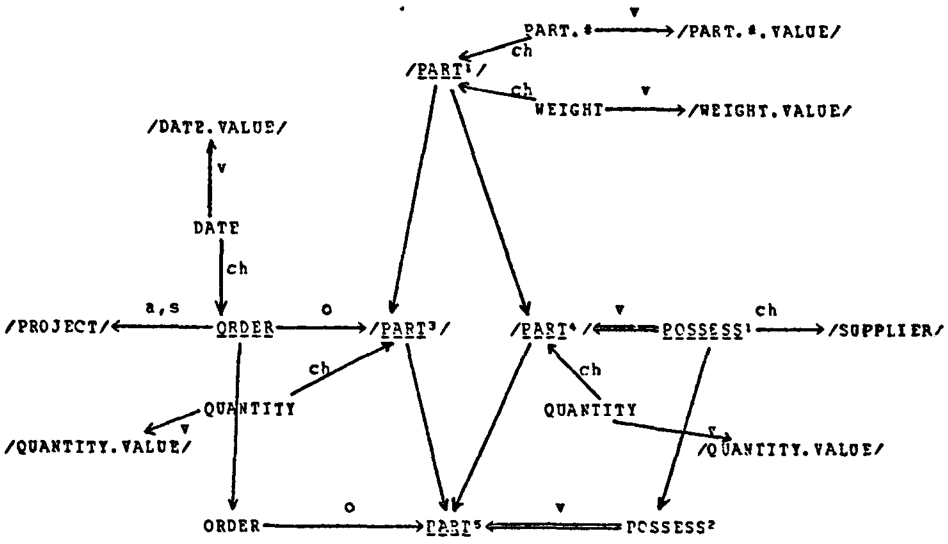

PART.#
v
/PART.#.VALUE/
ch
/PART¹/
ch
WEIGHT
v
/WEIGHT.VALUE/
/DATE.VALUE/
v
DATE
ch
a,s
/PROJECT/
ORDER
o
/PART³/
/PART⁴/
v
POSSESS¹
ch
/SUPPLIER/
ch
QUANTITY
v
/QUANTITY.VALUE/
ch
QUANTITY
v
/QUANTITY.VALUE/
ORDER
o
PART⁵
v
POSSESS²

fig. 4.3

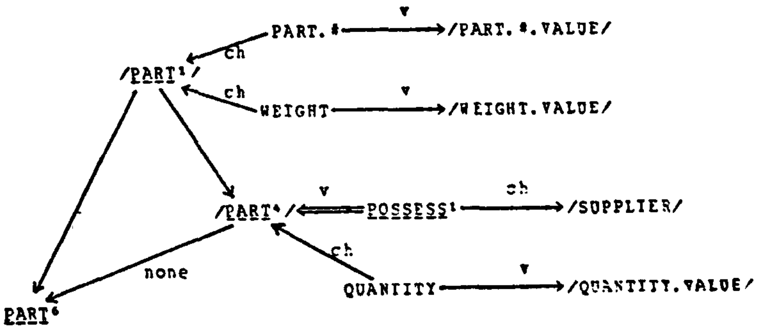

PART.#
v
/PART.#.VALUE/
ch
/PART¹/
ch
WEIGHT
v
/WEIGHT.VALUE/
v
/PART⁴/
POSSESS¹
ch
/SUPPLIER/
ch
none
QUANTITY
v
/QUANTITY.VALUE/
PART⁶

fig. 4.4

e. Division

Division is the semantic operator that is related to our representation of quantification. It takes as arguments

i. an event or a characteristic node n (the dividend)

ii. a node nd (the divisor), and a case-node n1 of n, over which division is to be applied

iii. one or more case-nodes n2, n3,.. of n with respect to which the division is to be applied

It results in

i. a new node nr below n

ii. new nodes nr1,nr2,.., case-nodes of nr corresponding one-to-one with the cases of n

iii. a new edge labelled "all" from nd to nr1 to indicate the node over which the division was applied

iv. one or more edges labelled "e" from n2,n3,.. to nr2,nr3,.. respectively to indicate the node(s) with respect to which the division was applied.

For example,

'suppliers possessing all parts ordered by project pj1'

operates on node POSSFSS^1 and PART^7 over node PAPT^4 with respect to node SUPPLIFR^1 on fig. 4.5 and results in nodes POSSESS^3, PART^8 and SUPPLIEP^2, as shown on fig. 4.5, along with the appropriate links created by the division.

The corresponding data base operator of division takes as arguments

i. an event or a characteristic relation (dividend) associated with node n

ii. a concept relation (divisor) associated with node nd

iii. an attribute of the dividend relation over which the division is to be applied (corresponding to node n1)

iv. one or more other attributes of the dividend relation with respect to which the division is to be applied (corresponding to nodes n2,n3,..)

It results in a subrelation of the dividend relation and is associated with node nr. Thus in our example,

POSSESS^1(PAFT, SUPPLIER, QUANTITY.VALUE)
(dividend)
PART^7(PART, PART.#.VALUE, WEIGHT.VALUE)
(divisor)

are divided and result in

POSSESS^3(PAPT, SUPPLIER, QUANTITY.VALUE)

which is associated with POSSFSS^3 in fig. 4.5.

Note that our data base division is slightly different from the one given in [4]. In our definition an extra argument is provided which specifies the attribute(s) with-respect-to which division is applied. Thus the dividend relation does not have to be binary.

4.2. Execution of data base operators

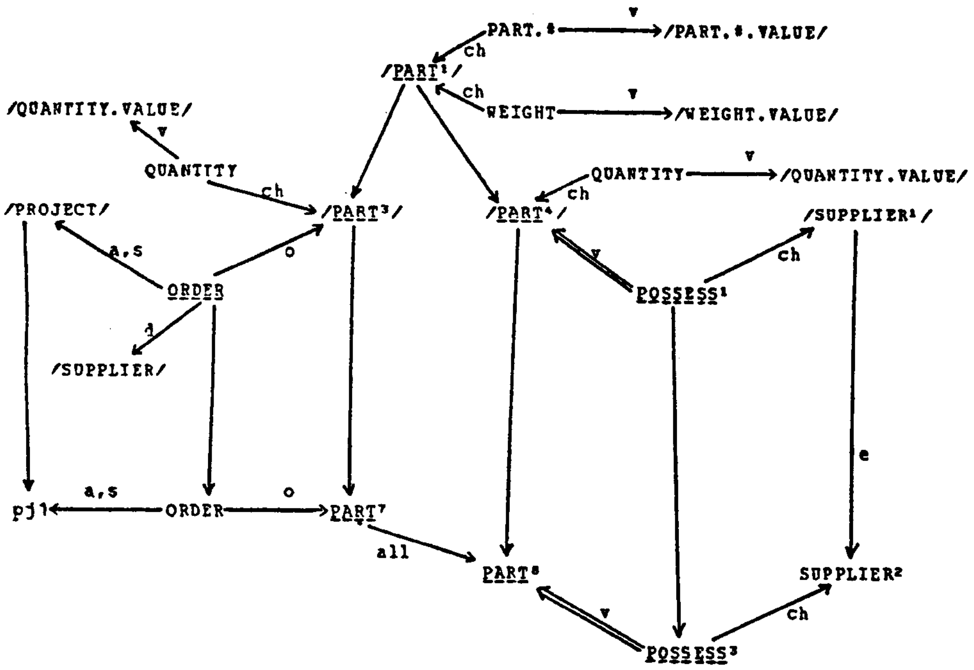

PART.#
v
/PART.#.VALUE/
ch
/PART^1/
ch
WEIGHT
v
/WEIGHT.VALUE/
/QUANTITY.VALUE/
v
QUANTITY
ch
QUANTITY
v
/QUANTITY.VALUE/
ch
/PROJECT/
/PART^3/
/PART^4/
/SUPPLIER^1/
a,s
o
v
ch
ORDER
POSSESS^1
d
/SUPPLIER/
e
a,s
o
pj1
ORDER
PART^7
all
PART^8
SUPPLIER^2
v
ch
POSSESS^3

fig. 4.5

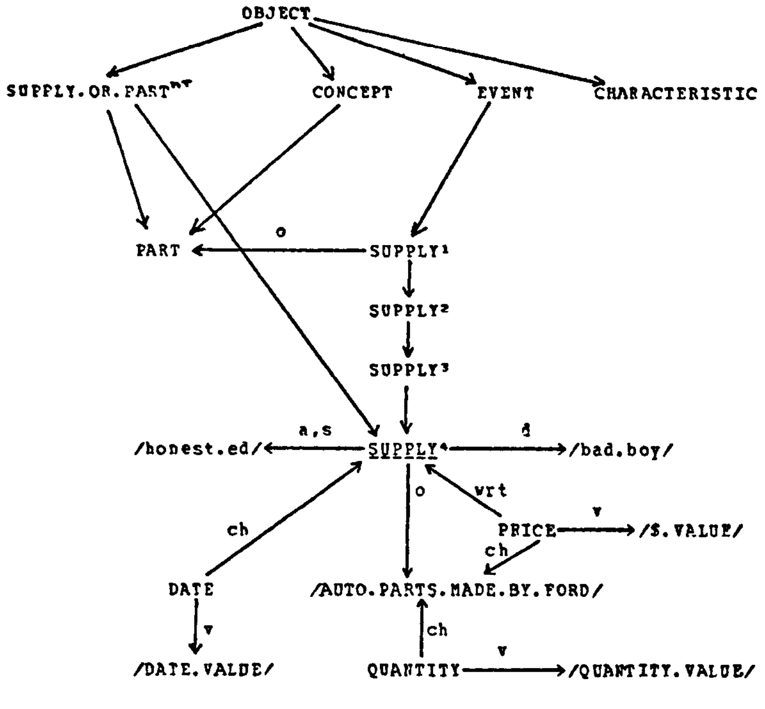

OBJECT
SUPPLY.OR.PART
CONCEPT
EVENT
CHARACTERISTIC
o
PART
SUPPLY^1
SUPPLY^2
SUPPLY^3
a,s
d
/honest.ed/
SUPPLY^4
/bad.boy/
o
vrt
ch
PRICE
v
/$.VALUE/
ch
DATE
/AUTO.PARTS.MADE.BY.FORD/
v
ch
/DATE.VALUE/
QUANTITY
v
/QUANTITY.VALUE/

fig. 4.6

Consider the statements

'find all parts which have weight greater than 10lbs'

and

'honest.ed increased the prices of parts which have weight greater than 10lbs'

Both statements involve the execution of the (semantic) selection operator, as shown in fig. 4.1. The question is whether the data base selection operator must be executed at the same time or whether its execution can be deferred. In the case of the first statement execution of the data base selection operator appears necessary, so that the FIND command can be carried out. For the second statement, however, creation of a new relation through the data base selection operator may be altogether unnecessary.

Our general position on this issue is that data base operations are not carried out when corresponding semantic ones are, but rather, when definitional functions (see section 2.1) corresponding to system commands -such as "find", "update", "insert", "delete", etc.- are executed.

4.3. "Legality" of semantic operations

The data base operations we have defined, like the original ones introduced by Codd, place certain restrictions on the relations that may serve as their operands. For example, it is not possible to take the union of the relations

SUPPLY[4](honest.ed, bad.boy, AUTO.PARTS.MADE.BY.FORD,
DATE.VALUE, $.VALUE, QUANTITY.VALUE)

and

PART[1](PART, PART.#.VALUE, WEIGHT.VALUE).

On the other hand, the expression

'cases where honest.ed supplied auto.parts.by.ford, or parts supplied to projects'

can cause the creation of the node marked nr on the net, and it can therefore be said to "make sense". The node OBJECT in fig. 4.6 is the highest node on the net with respect to the SUB axis. The conclusion to be drawn from this example is that semantic operators are more general than data base ones and that there will be situations where the data base operation associated to a semantic one cannot be carried out.

Given that there are no restrictions on the application of semantic operators similar to those that exist for data base ones, the reader may still wonder whether there is at least a measure of "strangeness" that could be introduced to make the model suspicious of expressions such as the above. Such measures of "strangeness" are in fact possible and depend directly on the semantic net representation. Thus, any semantic operation that causes the creation of a node so high on the SUB axis, and therefore so far removed from what would normally be expected to be of interest (e.g. through context), may raise questions on a system's part regarding the user's credibility, infallibility, sanity, or whatever. What action the system takes depends on the designer of the system. Our point is that as long as the semantic model is used, there are no clearcut "illegal" semantic operations, although some semantic operations are rendered more expected and less "strange" than others because of the structure of the semantic net and its associated mechanisms (e.g. context).

5. MAINTAINING THE DATA BASE CONSISTENT

Consistency is an important issue in data base management and some efforts have been made to account for it. For example, normalization [3] and insertion, deletion and modification rules [6,13] were introduced to avoid certain kinds of anomalies caused by the execution of such operations on the data base. These techniques are only applicable to primary relations, not to derived ones. They are not meant to maintain the data base consistent throughout insertion, deletion or modification operations but instead they describe what the user can or cannot do in order to avoid some incosistencies.

As was done in previous sections, we approach this issue by first defining what are the semantic implications of insertions, deletions and modifications on the net, and from those we derive the appropriate sequence of data base operations to be performed. Two basic features of our semantic model are essential in the process of maintaining data base consistency. The first one is the relative position on the net of the information to be inserted, deleted or modified and the second is the different axes and other edges available in the model which define the various relationships among attributes and relations of the data base. It should be pointed out that the methods we are describing here will keep the data base consistent with respect to the semantic net. Thus, if the net is inadequate, so will be the notion of consistency that will be derived from it.

This section includes four examples which will demonstrate how the semantic model maintains a data base consistent. Space considerations force us to use the tiny semantic net described so far which has very limited knowledge, as we will demonstrate in the fourth example.

Example 1. Consider the statement

'honest.ed supplied bad.boy with 100 tables on July 15, 1974'.

Although this statement is meaningful it has no place in the world of our data base. The semantic net (as it has been described so far) only knows 'honest.ed' as a source of parts and since 'tables' are not parts, the statement is immediately rejected and no change is made to the data base.

Example 2. Consider now the statement

'honest.ed supplies bad.boy with cadillac fenders'.

Since 'cadillac fenders' are parts there exists a position on the net where this information can be placed. This position is below SUPPLY^2 on fig. 2.1(b). Note that this information cannot be moved any further down the SUB axis because although SUPPLY^3 on fig. 2.1(c) has 'honest.ed' and 'bad.boy' as agent-source and destination respectively, its object case is PARTS.MADE.BY.FORD which does not match 'cadillac fenders' (made by GM). Accordingly, this instantiation of SUPPLY^2 is inserted as a data base tuple if SUPPLY^2 is realized. Similarly, the same tuple is inserted to all superrelations of SUPPLY^2.

In both examples given so far the recognition functions (see section 2.1) for PARTs and AUTO.PARTS.MADE.BY.FORD play an important role in maintaning the integrity of the data base while the SUB axis is used for maintaining consistency. Also note that the process of graph-fitting a query to the data base is instrumental in determining what should be done about the query.

Example 3. Our third example demonstrates how consistency is maintained for primary as well as derived relations. Consider the statement

'supplier dominion electric now possesses 83 generators'

The position of this statement on the net is below POSSESS^1 shown in figures 4.3-4.5. Node POSSESS^1 is realized (underlined) and thus the appropriate tuple conveying the new information is inserted to the data base. Similar insertions must be made for all superrelations of POSSESS^1, if any.

When this new information is inserted in POSSESS^1, it may cause inconsistencies to other relations namely PART^5, PART^6 and POSSESS^3 which store

'parts that have been ordered by some project and possessed by some suppliers'

'parts that no supplier possesses'

and

'suppliers possessing all parts ordered by pj1'

respectively (see section 4.1). The semantic model can detect what is affected by the new information by searching below POSSESS^1 along the SUB axis and by matching the new information against other scenarios. Partially matched scenarios, created by semantic operations, may be affected, in which case the data base operations which created their associated data base relations are executed again.

Example 4. Our last example concerns deletions. Consider the statement

'sears no longer supplies bad.boy with
auto.parts.made.by.ford'

Its position is exactly the same as the position of SUPPLY^{14} on fig. 4.2. Note that SUPPLY^{14} is a generic event which might have instantiations and/or generic subevents. Deletion of the relation SUPPLY^{14} must be followed by deletion of all its subrelations, if any, in order to maintain consistency. The reason for this is that SUPPLY^{14} is the connecting event between SUPPLY^{15} and its associated relation and the other subevents of SUPPLY^{14} which are not applicable any more. It should be pointed out that any information about 'auto.parts.made.by.ford supplied by sears to bad.boy in the past' will be lost once these changes to the net and the data base have been made. This is not a deficiency of our model but rather of the network we use. If one wants to extend the data base's world to include information about the past, then a "time" axis [11] has to be included in the net, which will specify the period of applicability for each scenario.

Returning to our example, the tuple that corresponds to the deletion of SUPPLY^{14} will be removed from the relation SUPPLY^{15}. In general, deletion of a tuple from a relation r must be followed by deletions of the same tuple from all the subrelations of r while deletions of the same tuple from superrelations of r corresponding to higher level scenarios may follow if those scenarios match partially the information to be deleted.

Modifications of the data base are handled using the same techniques as for insertions and deletions.

6. CONCLUSIONS.

We have presented a semantic model of data bases which assumes the availability of a semantic network storing knowledge about a data base and a set of attributes for the data base. The use of the semantic net in generating a relational schema for the data base, in defining a set of semantic operators and in maintaining the data base consistent is then demonstrated and it is shown that the model does not distinguish between primary and derived relations of a data base.

The description of the semantic model is by no means complete. More work has to be done to establish that the association of relations to basic building blocks of the semantic net (concepts, events and characteristics) is adequate, that the set of semantic operators we have proposed is in fact sufficient and that other aspects of consistency, integrity, cost and security can be handled by the semantic net representation we have proposed so far. We believe, however, that the results of this paper set the foundations of a semantic model for data bases, with respect to goals as well as methodology.

ACKNOWLEDGMENT

The authors would like to thank Phil Cohen and Hans Schmid for their helpful comments.

REFERENCES

1. Bernstein, P.A., Swenson, J.R., Tsichritzis, D., "A unified approach to functional dependencies and relations", Proc. of ACM SIGMOD Workshop, San Francisco, May 1975.
2. Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, vol. 13, no. 6, June 1970, 377-387.
3. Codd, E.F., "Further Normalization of the Data Base Relational Model", Courant Computer Science Symposia 6, Data Base Systems, New York City, May 24-25, 1971, Prentice-Hall.
4. Codd, E.F., "Relational Completeness of Data Base Sublanguages", Courant Computer Science Symbosia 6, Data Base Systems, New York City, May 24-25, 1971, Prentice-Hall.
5. Codd, E.F., "Recent investigations in Relational Data Base Systems", Proc. of IFIP 1974, North Holland Pub. Co., Amsterdam 1974, 1017-1021.
6. Deheneffe, C., Hennebert, H., Paulus, W., "Relational Model for Data Base", Proc. of IFIP 1974, North Holland Pub. Co., Amsterdam 1974, 1022-1025.
7. Delobel, C., Casey, R.G., "Decomposition of a Data Base and the Theory of Boolean Switching Functions", IBM Journal of Research and Developments, Vol. 17, No. 5, Sept. 1973, pp. 374-386.
8. Fillmore, C., "The case for case", In *Universals in Linguistic Theory*, Bach, E. and Harms, R., (eds.), Holt, Rinehart and Winston Inc., Chicago, Illinois, 1968.
9. Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N., Tsotsos, J., Wong, H., "TORUS: A Natural Language Understanding System for Data Management", Proceedings of the 4-th International Joint Conference on AI, Tbilisi, USSR, Sept. 1975.
10. Mylopoulos, J., Cohen, P., Borgida, A., Sugar, L., "Semantic Networks and the Generation of Context", Proceedings of the 4-th International Joint Conference on AI, Tbilisi, USSR, Sept. 1975.
11. Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N., Tsotsos, J., Wong, H., "The TORUS Project: Progress Report", In preparation, Dept. of Computer Science, University of Toronto.
12. Reason, C., Sugar, L., "Reference Determination and Context, as applied to TORUS", Unpublished report, Dept. of Computer Science, University of Toronto, Toronto 1975.
13. Schmid, H.A., Swenson, J.R., "On the Semantics of the Relational Data Model", Proceedings of SIGMOND Conference, San Jose, May 1975.
14. Wang, C.P., Wedekind, H.H., "Sequent synthesis in logical data base design", IBM Journal Research and Development, vol. 19, no. 1, January 1975, pp. 71-77.