TORUS--
A Natural Language Understanding System for Data Management

by

John Mylopoulos, Alex Borgida, Philip Cohen,
Nicholas Roussopoulos, John Tsotaos and Harry Wong

Department of Computer Science
University of Toronto
Toronto, Canada

Abstract

This paper describes TORUS, a natural language understanding system which serves as a front end to a data base management system in order to facilitate communication with a casual user. The system uses a semantic network for "understanding" each input statement and for deciding what information to output in response. The semantic network stores general knowledge about the problem domain, in this case "student files" and the educational process at the University of Toronto, along with specific information obtained during the dialogue with the user. A number of associated functions make it possible to integrate the meaning of an input statement to the semantic network, and to select a portion of the semantic network which stores information that must be output. A first version of TORUS has been implemented and is currently being tested.

1. Introduction

This paper describes the scope and current status of the TORUS (TORONTO Understanding System) project.

The main goal of the project is to develop a methodology for designing and implementing "Understanding" Data Base Management Systems (UDBMSs), i.e., systems which have available

(a) A data base stored in and maintained by a data base management system (DBMS),

(b) A natural language understanding system (NLUS), serving as a front end to the DBMS, which is given knowledge about the data, can understand and respond to simple English sentences and can engage the user in a dialogue.

Such systems can obviously play a very important role in the future in making data bases available to casual users who have neither the time nor the patience to learn a programming or query language so that they can communicate with the computer.

As a first step towards achieving the basic goal of TORUS, we have designed and implemented a prototype UDBMS, using a data base of graduate student files. Two of the most important methodological decisions we made are given below:

(a) We have fixed the type of DBMS available to the UDBMS to be a relational system along the lines proposed by Codd [1]. This decision enables us to be more explicit about the NLUS-DBMS interface and also makes our work easier since relational DBMSs offer a more powerful set of "higher level" commands than other types of DBMSs.

(b) We have decided to base the "understanding" capabilities of the NLUS on semantic nets, i.e., directed labelled graphs whose nodes represent conceptual objects and whose edges represent semantic relations. More specifically, we have based our semantic net representation of knowledge on a case structured representation and the use of a form of the subset and ISA relation found in several other representations [3, 4]. More information about the TORUS representation is given elsewhere [S].

Our emphasis has been on modelling an institutional world rather than a physical world. This fact has influenced much of the design of TORUS.

There already exist today several question-answering systems (e.g., CONVERSE [6], REL [7], LUNAR [8] etc.), whose main difference from TORUS is that they lack facilities for "understanding" the on-going dialogue (such as a semantic net or PIANNER-like theorems etc.). There also exist a few NLUSs (e.g., SHRDLU [9], ELINOR [3], MARGIE [10] etc.) with which we share our methodology to varying degrees. It is our belief, however, that none of these systems, including ours, has succeeded yet in formulating an integrated methodology for constructing NLUSs and that more of them will have to be built to test new proposals and to integrate solutions to specific problems in natural language understanding.

It is assumed that the reader is familiar with basic relational calculus [I], and with case-grammar terminology [2].

Throughout the paper we will be using as examples the queries

"What is John Smith's address?"

and

"How many recommendation letters did we receive for him?"

2. The Representation

Information is divided into two general categories and is accordingly stored in a DBMS available to TORUS or on the semantic net.

2.1 Information that is part of a student's file

Such information is stored in a relational DBMS in terms of n-tuples that are elements of relations. For example, one of the relations used for the representation of information regarding student files is the BIODATA relation:

BIODATA(NAME, ADDRESS, TELEPHONE-*, STATUS-IN-CANADA, FIELD-OF-STUDY, FULL-OR-PART-TIME)

Underlined attributes indicate the keys of the relation. One instance of thi& relation, regarding

the person named "John Smith" is given below:

```
BIODATA('john smith', '65 st. george st.,
    toronto, canada', 9286324, 'citizen',
    'computer science', 'full-time').
```

Another relation is

RECOMMENDATION-LETTER (NAME, DATE-RECEIVED, RECOMMENDING-PERSON, ADDRESS, LETTER)

and some instances of it, regarding "John Smith", are

RECOMMENDATION-LETTERC john smith', 740517, 'jim brown', 30 willowdale ave., toronto, canada', A)

RECOMMENDATION-LETTER('john smith', 740621, 'joe doe', '367 Charles St., toronto, canada', B)

where 740517 and 740621 specify dates {year-month-day) and A and B stand for the contents of the two letters.

We will often refer to the collection of student files stored in the DBMS as the data base.

2.2 Information about student files

The system also stores, in semantic net form, general knowledge about students, the educational process, and student files, along with the system's understanding of the on-going dialogue it may be having with a user. It should be noted that sometimes a particular item of knowledge will be stored on the semantic net and the data base at the same time, just as it may be remembered (temporarily) by a secretary and also appear in some student's file.

The semantic net representation is similar in several ways to those proposed and used by Rumelhart et al. [3] and Martin [11], and is described in more detail elsewhere [5]. Our goal here is to give enough information about it so that the reader will understand the examples to be given in the following sections.

Nodes on the semantic net can be concepts, events and characteristics. Thus the generic ideas PERSOK, UNIVERSITY, as well as instantiations "John Smith" and "University of Toronto" are all concepts. Similarly, the generic idea of RECEIVE (such that every particular "receiving" action is its instantiation), as well as specific instantiations of the idea, are all events. Finally, ideas which express states or properties of objects, actions or other properties are characteristics. For example, STUDENT-NUMBER characterizes students and can have as values 9-digit integers.

As noted above, an important classification of nodes is whether they represent generic ideas or instantiations of them. Generic nodes are placed "upstairs" and are denoted by names in capital letters, while specific instances are put "downstairs" in small letters.

Objects on the semantic net are organized in a superset-subset hierarchy, defined in terms of SUB(set)-labelled edges and E(xample-of)-labelled edges. Thus STUDENT is a subconcept of PERSON while "John smith" is an example (instantiation) of PERSON and will therefore be represented on the semantic net as shown in FIG. 2.1(a).

SUB-labelled edges can only link objects located upstairs, while E-labelled edges always

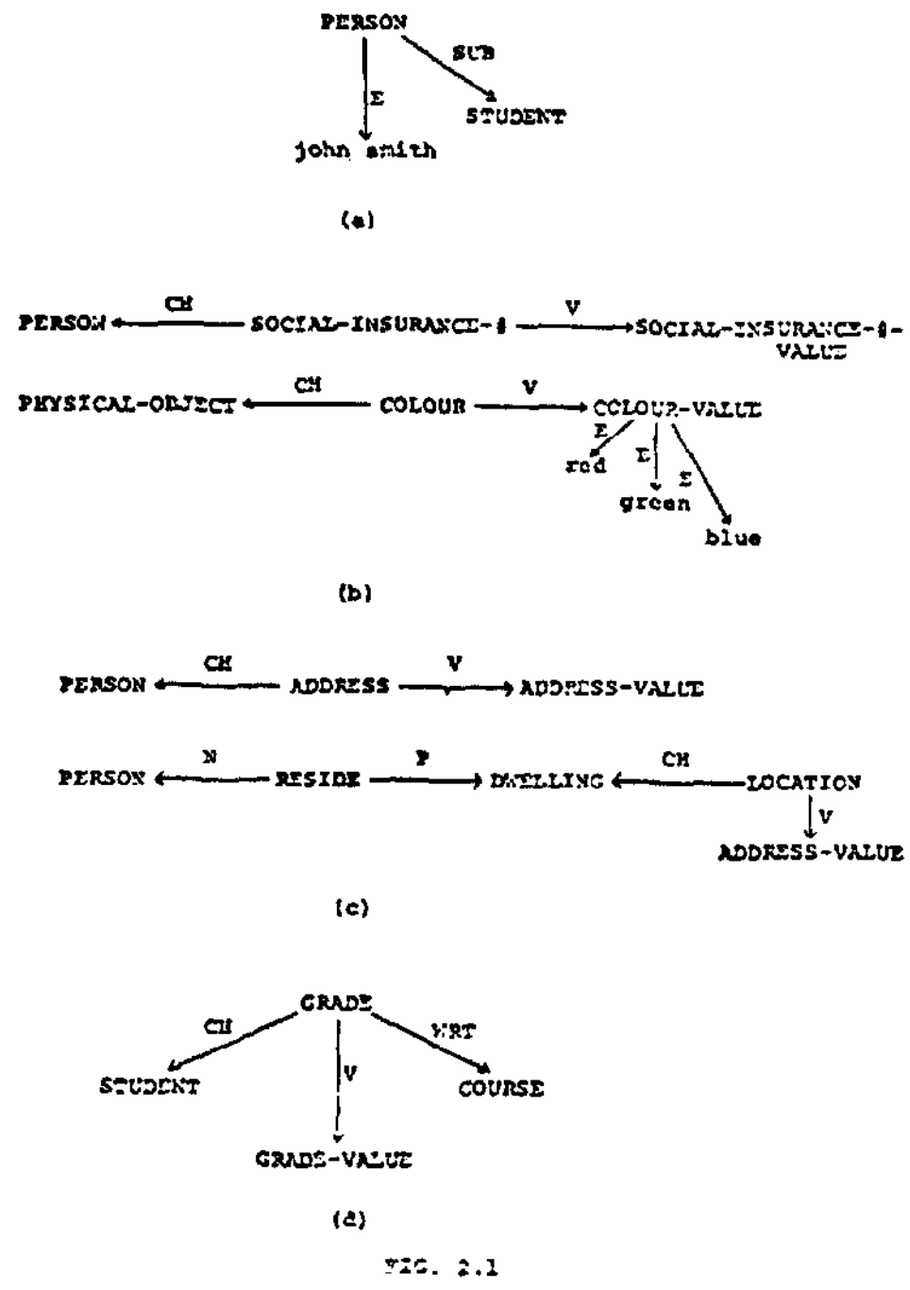


FIG. 2.1

link a node located upstairs to one located downstairs. The properties of any object on the hierarchy are inherited by all objects below it, unless otherwise specified for any of these objects.

Another important relation we will allow, usually between concepts, is the PART relation. Thus FINGER is a PART of ARM which in turn is PART of BODY. Note that, in general, properties of objects are not inherited along PART-labelled edges.

An important class of edges are the cases used to specify semantic relations between an event and other objects of the semantic net. The cases we will use are AGENT (A), OBJECTIVE (0), AFFECTED (AFF), TOPIC <T), INSTRUMENT (I), RESULT (R), SOURCE (S), DESTINATION (D) and NEUTRAL (N). The semantics of most of these cases are implied by their name.

Characteristics have their own edges labelled CHARACTERISTIC {CH> and VALUE (V) which specify respectively the object being characterized and the value of the characteristic. FIG. 2.1(b) shows two examples.

Note that characteristics are meant to define basic properties of objects, such as the mass, charge and velocity of an elementary particle, the colour of an object or the social insurance number of a person. In many situations, however, we will use them to specify that a semantic relation exists

between two objects when we do not want to explicate that relation. For example, ADDRESS will be treated as a characteristic of PERSON, although a deeper representation would have made explicit the semantic relation between a person and what we call his address (FIG. 2.1(c)).

Sometimes we will want to attach a third edge labelled WRT (with-respect-to) a characteristic. For example, the characteristic GRADE needs a WRT-labelled edge to specify the course with respect to which a student is characterized by a particular grade (FIG. 2.1(d)).

LOCATION, ABSOLUTE and RELATIVE TIME and DURATION will all be represented in terms of characteristics.

Apart from the above-mentioned edges, the representation allows connectives of various kinds such as

BEFORE, AFTER, WHILE (temporal)
PREREQUISITE, EFFECT (causal) etc.

Figures 2.2 and 2.3 show portions of the semantic net which will be used in the examples that follow.

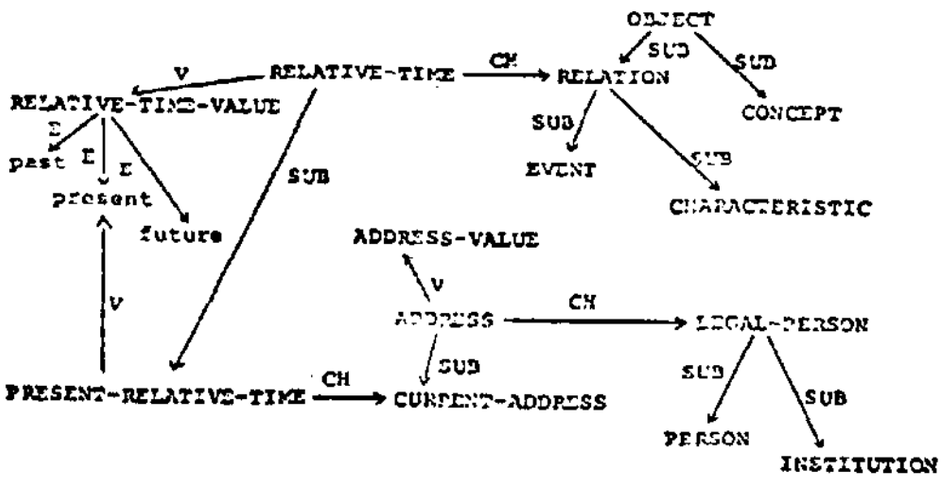


FIG. 2.2

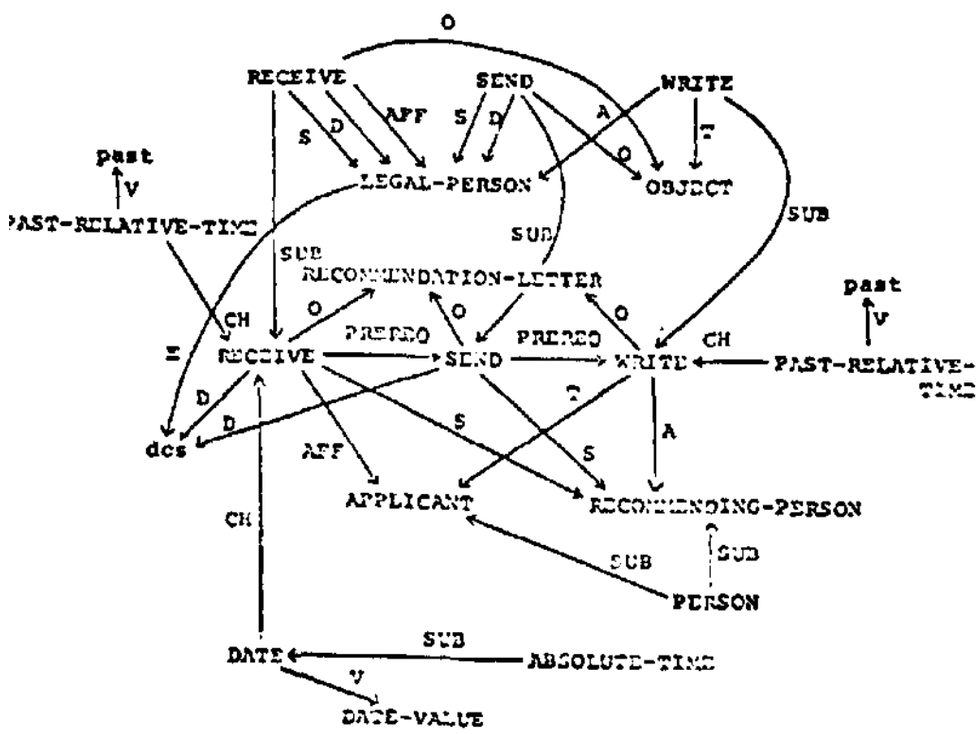


FIG. 2.3

For convenience we have omitted SUB-labelled edges which link EVENT, CHARACTERISTIC or CONCEPT (which are generic objects on the semantic net) to other events, characteristics and concepts. The types of the nodes can be determined by examining the edges that leave them.

The important things to note in FIG. 2.2 are

- an address can be associated to any person or institution (the union of these generic concepts we have named LEGAL-PERSON)
- CURRENT-ADDRESS is a sub-characteristic of the characteristic ADDRESS and its distinguishing feature is the fact that when characterized by RELATIVE-TIME, the value of the latter is always "present".

In FIG. 2.3 we have defined the scenario of WRITING-A-RECOMMENDATION-LETTER-TO-DCS.

2.3 Relating the attributes of data base relations to objects on the semantic net

Most data base attributes have nodes of the semantic net associated to them. These nodes define the semantics of the corresponding attributes and they can therefore be used in question-answering and in determining whether a given relational operation on one or more attributes makes sense.

For example, to the node of FIG. 2.2 named CURRENT-ADDRESS we will associate the attribute named ADDRESS of the BIODATA relation. This means that

- if someone asks a question involving the value of a person's current address, the system could retrieve the entry of the ADDRESS attribute for a BIODATA relation instance whose NAME attribute has the person's name as value.
- the semantics of the attribute ADDRESS are defined by the semantic relations associated with the node CURRENT-ADDRESS.

Similarly, we can associate the data base attributes NAME, RECOMMENDING-PERSON, DATE-RECEIVED and LETTER to the nodes named PERSON, RECOMMENDING-PERSON, DATE and RECOMMENDATION-LETTER of FIG. 2.3 respectively.

3. Understanding Input Sentences

In this section we describe the analysis carried out by TORUS when a sentence is presented to it by the user.

For convenience, the analysis will be partitioned into a syntactic and semantic component. The syntactic analysis is responsible for producing a parse tree (modified deep structure) of the input sentence, while the semantic analysis is responsible for constructing a semantic graph and for integrating it to the semantic net. It must be stressed at this point that the modularization and linearization of the understanding process was adopted only in order to by-pass the main memory constraints of present-day computers and to enable us to see more clearly the various problems involved. The sole purpose of the syntactic analysis is to identify the syntactic constituents corresponding to most of the semantic units making up the representation of the sentence's meaning.

3.1 Syntactic analysis

The first step consists of a complete word-by-word morphological analysis of the input sentence during which we deal with problems of word recognition, inflectional and derivational affix analysis and contractions. This section of

the system is also responsible for recognizing contiguous sets of words such as idioms and compound words, and for replacing them with other words or symbols which the rest of the system can understand (e.g., "Department of Computer Science" will be replaced by "dcs").

The output of this section of the algorithm is a table of dictionary entries, including multiple entries for ambiguous words, corresponding to the words in the original sentence.

The table is passed to an augmented transition network (hereafter ATN) grammar for syntactic analysis. The analysis is non-deterministic and several parses are possible for the same sentence. However, only one parse is produced at any one time. An interesting feature of the parser is the way in which it handles adverbials. Usually, adverbials can modify one of several predicates or noun phrases of the input and the modification which is actually correct can only be determined on semantic grounds. The parsing strategy we have adopted ensures that we can retrieve from the parse tree all and only the possible constituents which the adverbial can modify.

3.2 Constructing a semantic graph from a parse tree

By a semantic graph we mean here a graph which involves concepts, characteristics and events connected through semantic relations (cases, connectives etc.), but not yet integrated to the semantic net.

The preliminary processing of the parse tree finds possible referents for noun phrases (NPs) in the input. Each NP may be referring to an already existing object on the semantic net or a new one.

In the former case, either the referent is uniquely identified or a BVAR (Bound VARiable) node is created, with referent candidates connected to it by CAND edges. Question words replacing noun groups are represented by nodes labelled "?". See FIG. 3.1(a), (b) for examples.

Once a head node has been constructed for every NP, the system retrieves case frames stored with the predicate of each sentence and attempts to find the appropriate case fillers.

The case fitting algorithm for the second example will construct the (partial) semantic graph of FIG. 3.1(a) by using the case frame for RECEIVE.

The verbs BE and HAVE are handled in a special way by using functions rather than case frames, since they have many meanings and uses. Thus for the first example, the BE function notes that the parse tree fits the characteristic-value configuration and checks the selectional restrictions specified by the ADDRESS node. The result is the semantic structure shown in FIG. 3.1(b).

The next step to be carried out is the construction of a graph for each PP or NP. Each PP or NP consists of a head noun and zero or more modifiers. So far we have constructed a node for the head noun and we must now look at the modifiers Modifiers can be possessives, relative clauses.

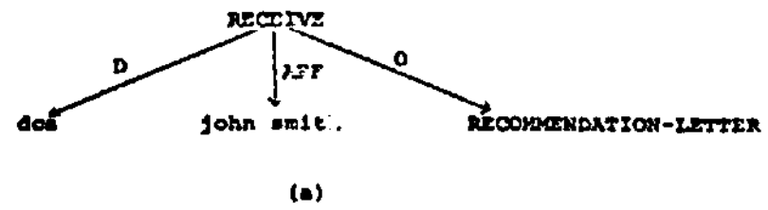


FIG. 3.1

participles, adjectives or unused PPs dominated by the current PP on the parse tree.

For example, possessive modifiers are handled by using a nine-point check list for the various possible semantic relations between two words involved in a construction of the form "P's N". Among them are:

(a) N is a characteristic of P ("John's height)
(b) P possesses or owns N ("Jim's ball")
(c) N is part of P ("Jim's leg")
(d) P plays a role (fills a case) in some event associated with N ("Mary's telegram" means "Mary sent/received a telegram").

Note that these cases, and especially (a) and (d), may not be mutually exclusive. Thus, as pointed out in section 2.2, "John's address" may be represented either in terms of a characteristic or an implied event, or both.

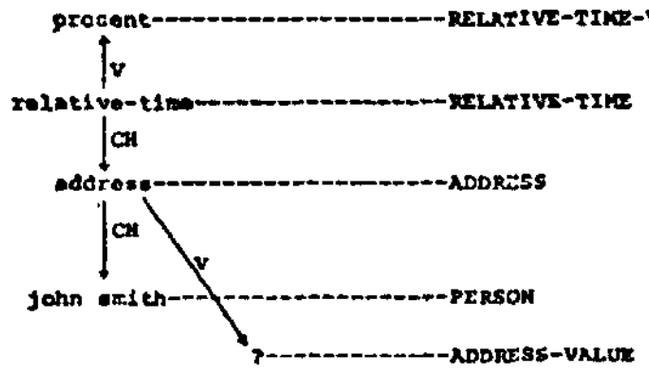
The other types of modifiers are also handled through heuristic rules, which will not be discussed here.

Once the construction of graphs for the PPs and NPs of the parse tree is complete, we attempt to determine the system commands to be used for this input sentence, if any. For example, one simple heuristic used to determine a command is: If the determiner is "how-many", then the command is COUNT (returning a number).

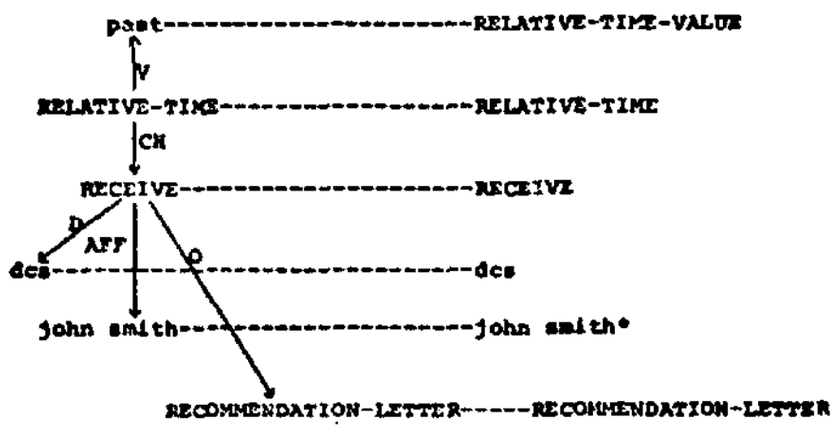
Not all verbs in the dictionary appear as events on the semantic net. For example, the graph for a sentence like

"John Smith took CSC 2505 from Jim Brown" which is shown in FIG. 3.1(c), will have to be mapped into the "deeper" semantic structure shown in FIG. 3.1(d) before it can be integrated with the semantic net. The task of mapping more surface semantic structures to deeper ones is carried out by mapping functions associated with some events or characteristics.

The final result of this part of the algorithm for our sample sentences is shown in FIG. 3.2. Note that each node of the semantic graph constructed is associated with a node of the semantic net shown in FIGs 2.2 and 2.3. More details about this part of the system can be found in [12].



(a)



(b)

FIG. 3.2

* Note that the node for "john smith" in the second sentence has been associated to the node added to the semantic net after the integration of the first sentence with the semantic net.

3.3 Integrating a semantic graph to the semantic net

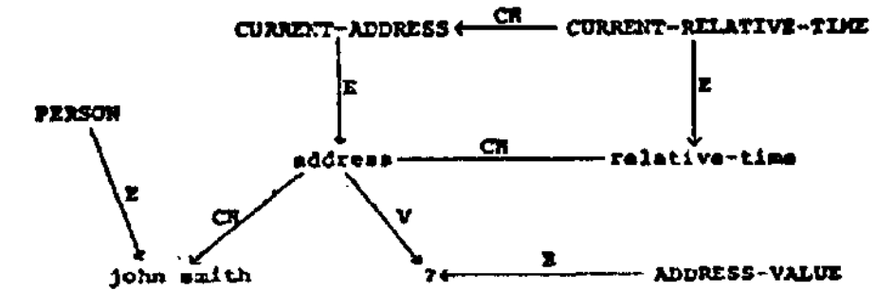
Due to the properties of the superset-subset hierarchy, there is a unique position on the semantic net for each semantic graph we wish to integrate. In this section we sketch briefly this integration process.

Many generic objects on the semantic net have recognition functions associated to them, which recognize instantiations of these generic objects.

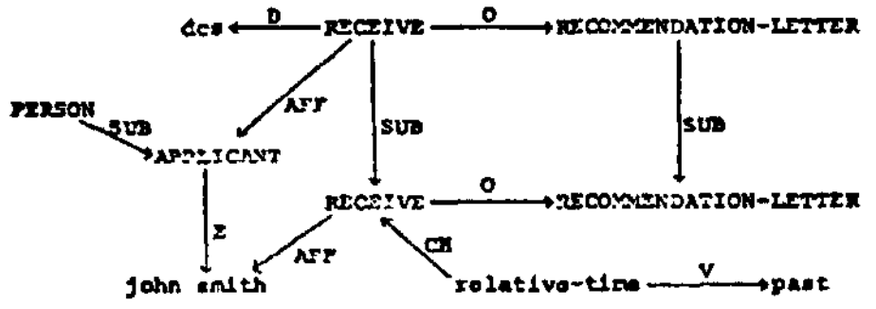
Consider, for example, the generic concept ADDRESS-VALUE and its associated recognition function IS-ADDRESS. When the semantic graph for, say, "John Smith's address is 65 St. George St., Toronto" is being integrated, IS-ADDRESS will be asked to determine whether "65 st. george st., toronto" is an instantiation of ADDRESS-VALUE. This will be done by using syntactic criteria and heuristics. These criteria and heuristics could have been expressed in terms of semantic relations, but we feel that in every universe of discourse there will be peripheral knowledge, such as recognizing addresses, which will best be integrated into the system through recognition functions rather than an expanded semantic network.

The integration of the semantic graph to the semantic net is carried out by moving the associations constructed during the previous step, shown in FIG. 3.2 as broken lines, as far "down" along SUB- and E-labelled edges as possible.

Portions of the resulting semantic net are shown in FIG. 3.3 for the two sample sentences.



(a)



(b)

FIG. 3.3

One of the interesting features of the integrating procedure is that in certain situations it will attempt to create missing cases, which are marked as obligatory, by calling the FIND-INSTANCES function or by using default values represented explicitly on the semantic net. Thus in integrating the semantic graph for

"What is his grade?"

FIND-INSTANCE will be called to find the relevant course associated with "his grade". To the question, however,

"What is. his status?"

the system will take "status" to mean "academic-status" and will also assume that it is to be considered WRT "dcs" (the default value in this case).

Another important feature of the integrating procedure is that it attempts to perform referent determination at a deeper semantic level than that tried earlier. For example, in determining a referent for "his" in

"What is his grade?"

one may consider first as candidate any male person. During the integration process, however, and as the semantic graph is moved down, it may be determined that the candidates should be

restricted to students (or graduate-students, or graduate-student-in-dcs). The integration process notes such restrictions and eliminates candidates which do not satisfy them.

4. Using the DBMS During a Dialogue

Once TORUS has found the "meaning" of an input sentence by constructing a semantic graph and integrating it to the semantic net, it must execute the generated system commands, if any. This section describes the execution of these commands and the interpretation of their outcome.

The section is divided into three subsections which deal with the commands increasingly closer to the DBMS:

(a) The semantic level: commands here operate on the data base attributes with no knowledge about how attributes are divided into relations.
(b) The interface level: commands here operate on one or more data base relations.
(c) The DBMS level: commands operate on individual data base relations.

4.1 The semantic level

The input to this level is a command-list, along with a list of pointers to objects on the semantic net on which these commands will operate. This level constructs a request for information from the interface level and sends this request to the interface. When this information is returned, the semantic net is updated accordingly.

For the first example, the result of this computation is the replacement of the '?'-labelled node (FIG. 3.3(a)) by John Smith's address, provided this can be found in the data base.

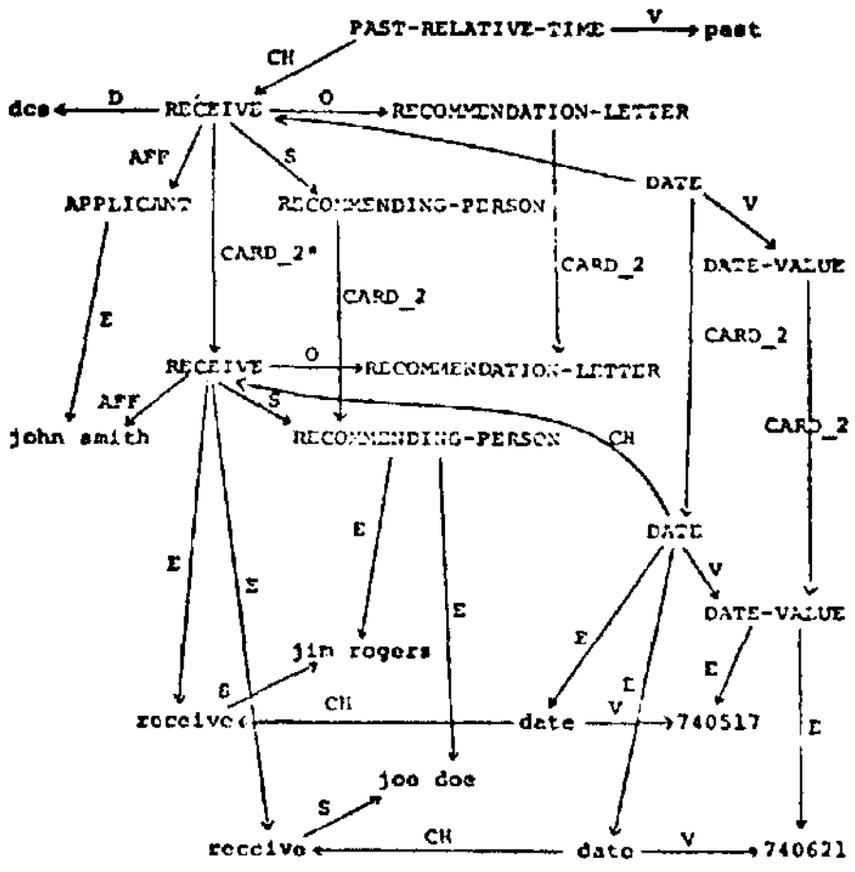


FIG. 4.1

* Edges labelled CARD_2 indicate that the object that is the target of the edge is a sub-object of the object that is the source, with precisely two instantiations. Thus, the next time a question is asked about the number of RECOMMENDATION-LETTERS for John Smith, we will be able to answer on the basis of the CARDinality edge we constructed at this point.

For the second example two letters of recommendation are retrieved from the data base and the resulting semantic net is shown in FIG. 4.1. Note that the RECEIVE event of the input sentence has been instantiated twice to account for the two instantiations of RECEIVE that were just retrieved from the data base.

4.2 The interface level

The interface level decides which relations should be accessed for the commands passed to it by the semantic level and what data base commands should be used. Security checking and cost estimation is also carried out at this level.

4.3 The DBMS level

This level offers primitives for maintaining and manipulating a relational data base through a system called MINIZ. Details about the system and a larger DBMS project called ZETA can be found elsewhere [13, 14].

5. Generating English Sentences

The problem of generating a response will be broken down into two parts:

(a) Selecting the information to be output
(b) Generating a sentence, given the information selected in (a).

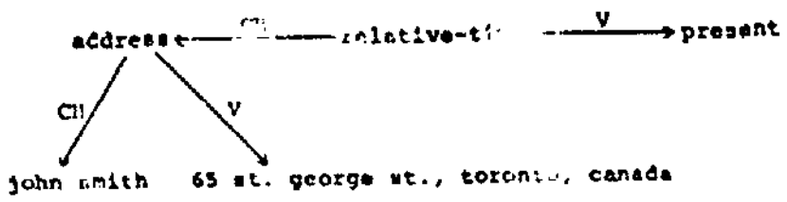
5.1 The selector

The selector's main job is to construct a graph which contains information relevant to the input statement. In constructing this graph the selector must:

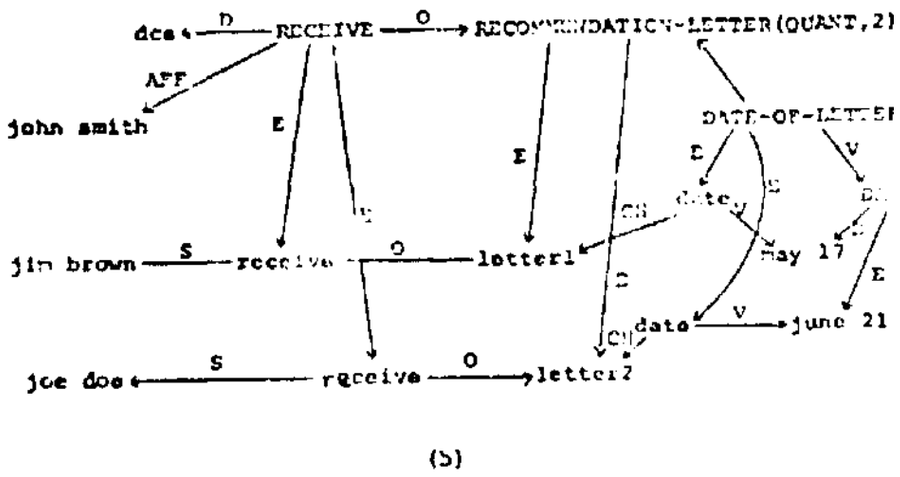
(a) Recover information stored implicitly on the semantic net. For example, it must infer the DESTINATION case of the RECEIVE event of the second sentence by taking into account the "transitivity" property of the superset-subset hierarchy.
(b) Consider stylistic problems such as pronominalization, nominalization, modalities, relative clauses and amount of detail that will be provided in the system's response.
(c) Decide whether it will output a sentence or a NP.

The input to the selector is a list of pointers to objects on the semantic net that constitute the main event or characteristic of the input sentence ("address" and RECEIVE respectively for our two examples) or stored information relevant to the input sentence (e.g., the address-value found for the first example, and the recommending persons along with the dates when the recommendation letters were sent for the second example).

The selector first copies the portion of the semantic net which is to be output (FIGs. 5.1(a) and 5.1(b) respectively for the two examples). It then uses inverse mapping functions to produce a more surface, but still case-grammar-based, representation of the information to be output. While this mapping is being carried out, NPs are constructed for the concepts of the answer graph



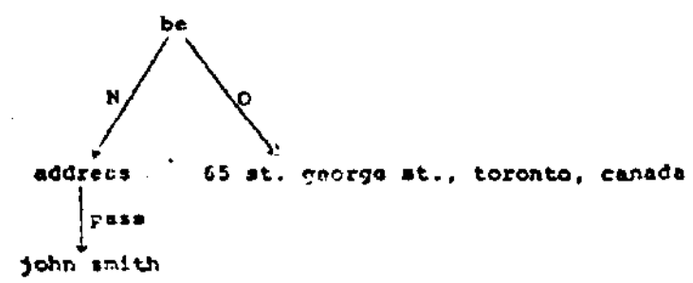
(a)



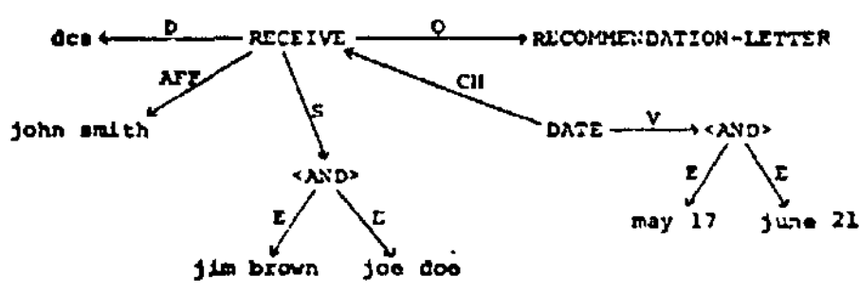
(b)

FIG. 5.1

and it is determined where to use pronominalization The resulting structures are shown in FIG. 5.2. Note that inverse mapping functions have mapped the numeric representation for dates (e.g., 750521) to a more surface one (e.g., 'may 21, 1975').



(a)



(b)

FIG. 5.2

Modality lists are constructed next and a surface ordering rule (SOR) is chosen for each verb of the resulting structure. SORs, as used here and elsewhere, [15, 16], basically specify the order in which the system should output the syntactic cases associated to a particular verb.

5.2 The generator

Once constructed, the answer graph is passed to the generator which must construct a sentence from it. This is accomplished by using a version of the Simmons-Slocum algorithm [16]. Some of the extensions we have incorporated into our generation algorithm are:

(a) The grammar we use, defined in terms of an ATN, is more complete than that described in [16] and allows complementation, reflexivization and complex NPs.
(b) Morphological functions are used to find regular inflections of verbs and nouns.
(c) Question-generation has been generalized to allow multiple queries.

More details about the selector and the generator can be found in [17].

The final output for the two examples is

"His address is 65 St. George St., Toronto, Canada"

and

"We have received two letters of recommendation from Jim Brown and Joe Doe on May 17 and June 21".

6. Implementation

A first version of TORUS has been implemented on an IBM 370/165 II under OS/MVT and is currently being tested. The languages used for the implementation are SPITBOL [18] and I.PAK, an extension of SPITBOL offering graphs and graph patterns [19]. Since MINI", has been implemented in PL/I, one part of the interface level was written in that language. The input dictionary contains at this time approximately 400 words, the semantic net only 100 nodes and we have written very few mapping, recognition and inverse mapping functions.

7. Concluding Remarks

The system we implemented is in many ways incomplete. Apart from expanding the vocabulary, the semantic net and the various tables used, and testing the system to determine its limitations, there are several important features which we are currently researching and propose to add to TORUS in the near future. Among them we note

(a) An inferential capability based on the semantics of connectives and cases.
(b) A question-generation facility which will enable the system to resolve difficulties it is encountering by asking questions to be answered either by the user, the semantic net or the DBMS.
(c) A more general referent determination routine based on inference and contextual expectations .
(d) More general mechanisms for handling conjunction, disjunction and quantification than those currently available in the system.

We believe at this time that these features can be added without having to modify or even extend the methodology we have been using. We currently are engaged in research on man-machine conversations and expect this research to have a major influence on future versions of TORUS.

We believe that TORUS has made important contributions in two areas of AI:

(a) Language Understanding

The main contribution here is in the emphasis we have placed on the design and implementation of an integrated language system for a particular

problem domain, in this case student files and the educational process. Designing a system with such goals is and will remain a difficult problem that will require expertise in combining and possibly modifying solutions that have been proposed to specific language problems (input, output, representation, referent determination etc.). Moreover, a methodology must be developed for considering a problem domain and representing the relevant knowledge. We know of only two other works (Winograd [9], Scragg [21]) which have tackled this problem. The student files world is different from Winograd's Blocks world and Scragg's kitchen world primarily in that the relevant knowledge is mostly institutional rather than physical in nature and this has naturally led us into a different set of representational primitives.

(b) Applications of AI

As we have already mentioned in the introduction, the use of language understanding systems in data management is very desirable and will soon become quite necessary in order to make data bases available to the casual user ("Date Bases to the People" is our slogan). In order to achieve this we will have to study the problem of interfacing NLUSs with DBMSs, a problem that is as much within AI as those of interfacing an understanding system to an ear (speech understanding), an eye (visual understanding) or a body (robotics). Related to the interface problem, are the DBMSs' problems of selecting a relational schema, given a semantic network description of a data base, maintaining the consistency of a (relational) data base by using the associated semantic network, and deciding on issues of cost and security regarding the data base by, again, using the associated semantic network. We tackled some of these problems and there are already more recent results (Schmid and Swenson [20], Roussopoulos [22]) that have been influenced by TORUS.

Traditional approaches to data base management emphasize the syntax of data using such phrases as "repeating group" etc. The relational approach, though still syntactic, is moving away from syntax ("relations", "functional dependencies", etc.). However, problems still exist due to the lack of semantics of the real world. TORUS is intended to be a bridge to this realm.

ACKNOWLEDGEMENTS

We would like to acknowledge the contributions of Walter Berndl, Lindsay Watt and Hector Levesque in the design and implementation of the TORUS system. Moreover, we are thankful to Stewart Schuster, Denis Tsichritzis and the data management group at the University of Toronto for their cooperation on the joint TORUS/ZETA project. Finally, we are grateful to Angie Blonski and Vicky Shum for their help in typing and preparing the figures for this paper.

This research was supported in part by the Department of Communications of Canada and the National Research Council of Canada.

REFERENCES

1. Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", CACM, June 1970.
2. Fillmore, C.J., "The Case for Case", Universals in Linguistic Theory, Bach and Harms (Eds.), Holt, Rinehart and Winston, 1968.
3. Norman, D., Rumelhart, D., Experiments in Cognition, Freeman, 1975.
4. Winston, P., "Learning Structural Descriptions", MIT Project MAC TR-76, September 1970.
5. Cohen, P., Mylopoulos, J., Borgida, A., Sugar, L., "Semantic Networks and the Generation of Context", 4IJCAI.
6. Kellogg, C., Burger, J., Diller, T., Fogt, K., "The CONVERSE Natural Language Data Management System: Current Status and Plans", Proceedings of the Symposium on Information Storage and Retrieval, University of Maryland, 1971.
7. Thompson, F.B., Lockemann, P.C., Dostert, B., Deverill, R.S., "REL: A Rapidly Extensible Language System", Proceedings 24th ACM National Conference, New York, 1969.
8. Woods, W.A., "Progress in Natural Language Understanding - An Application to Lunar Geology", Proceedings NCC, November 1973, pp. 441-449.
9. Winograd, T., "Understanding Natural Language", Academic Press, 1972.
10. Schank, R., Goldman, N., Rieger, C., Riesbeck, C., "MARGIE: Memory, Analysis, Response, Generation and Inference on English", Proceedings Third International Joint Conference on AI, September 1973.
11. Martin, W.A., Automatic Programming Group Memos 6, 8, 11, 12, 13, MIT. Also OWL Memos 1, 2, 3, 4.
12. Borgida, A., "Topics in the Understanding of English Sentences by Computer", Technical Report No. 78, Dept. of Computer Science, University of Toronto, December 1974.
13. Czarnik, B., Schuster, S., Tsichritzis, D., "ZETA: A Relational Data Base Management System", to be presented at the Pacific ACM Conference, San Francisco, April 1975.
14. Mylopoulos, J., Schuster, S., Tsichritzis, D-, "A Multi-level Relational System", to be presented at NCC '75, May 1975.
15. Simmons, R.F., "Semantic Networks: Their Computation and Use in Understanding English Sentences", Computer Models of Thought and Language, Schank and Colby (Eds.), Freeman and Co., 1973.
16. Simmons, R.F. and Slocum, J., "Generating English Discourse from Semantic Networks", CACM, October 1972.
17. Wong, H., "Generating English Sentences from Semantic Structures", M.Sc. Thesis, Department of Computer Science, University of Toronto, January 1975.
18. Dewar, R., "SPITBOL", Illinois Institute of Technology, February 1971.
19. Mylopoulos, J., Badler, N., Melli, L., Roussopoulos, N., "I.PAK: A SNOBOL-Based Programming Language for AI Applications", Proceedings Third International Joint Conference on AI, August 1973.
20. Schmid, H.A., Swenson, J.R., "On the Semantics of the Relational Data Model", Proceedings of SIGMOD Conference, San Jose, May 1975.
21. Scragg, G.W., "Answering Questions about Processes" in [3].
22. Roussopoulos, N., "Semantically-Driven Data Management Systems", Unpublished memo, Dept. of Computer Science, University of Toronto, April 1975.