COLING 82, J. Horecký (ed.)
North-Holland Publishing Company
© Academia, 1982

NATURAL LANGUAGE UPDATES*

Sharon C. Salveter
David Maier
Computer Science Department
State University of New York at Stony Brook
Stony Brook, New York 11794
USA

A great deal of research effort has been expended in support of natural language (NL) database querying. English and English-like query systems already exist, such as ROBOT[Ha77], TQA[Da78], LUNAR[Wo76] and those described by Kaplan[Ka79], Walker[Wa78] and Waltz[Wa75]. Little effort has gone to NL database update [KD81, Br81, Sk80, CH81]. We want to extend NL interaction to include data modification (insert, delete, modify) rather than simply data extraction. The desirability and unavailability of NL database modification has been noted by Wiederhold, et al[Wi81]. Database systems currently do not contain structures for explicit modelling of real world changes.

NL querying of a database requires that the correspondence between the semantic description of the real world and the database definition be explicitly stated. The NL query system must translate a question phrased in terms of the semantic description into a question phrased in terms of the database definition, that is, into a data retrieval command in the language of the database system. For NL database modification, a stative correspondence between database states and real world states is not adequate. To support NL update we need to represent an active correspondence, the connection between real world changes and database updates. We have a means to describe the actions that change the state of the real world: active verbs. We also have a means to describe a change in the database state: a data manipulation language (DML) command. We must capture the notion that an action in the real world causes changes in the real world that must be reflected in the database, as shown in Fig. 1. But given a real world action, how do we find a DML sequence that will accomplish the corresponding changes in the database? We need to connect verbs like "schedule," "hire" and "reserve" with some structures that dictate appropriate DML sequences that perform the corresponding updates to the database. In addition, a verb may denote various actions, that is, it may have different senses.

There is no explicit database object that represents all the changes in the database that correspond to the changes in the real world brought about by actions such as "schedule." The desired situation is shown in Fig. 2, where RWSi statively corresponds to DBSi. We have an active correspondence between "schedule" and a parameterized database transaction PT. Different instances of the schedule action, S1 and S2, cause different changes in the real world state, from RWS1 to RWS2 or RWS3. From the active correspondence of "schedule" and PT we want to select the proper transaction, T1 or T2, to effect the correct change in the database.

We propose a structure, a verbgraph, to represent action verbs on the database side. One verbgraph is associated with each sense of a verb; its structure will represent all variants of that sense. A verbgraph exploits commonalities among the variants of a verb sense and also distinguishes the variants. A verbgraph is used to select DML sequences appropriate to reflect the actions of each variant of a verb sense. The primitives in the structures are relations, attributes and values from the database, employed in DML-like expressions.

Verbgraphs are extensions of frame structures used to reoresent verb meaning in Moran[Sa78, Sa79]. A verbgraph is directed acyclic graph with five kinds of nodes: header, footer, information, AND and OR. An example of a verbgraph is shown in Fig. 3. The header is the source of the graph, the footer is the sink. Every information node (represented by a rectangle) has one incoming and outgoing edge. An AND or OR node (represented by ⊗ and ○ , respectively) can have any number of incoming and outgoing edges. A variant corresponds to a complete directed path in the graph. We define a complete path to be a connected subgraph such that (1) the header is included; (2) the footer is included; (3) if it contains an information node, it contains the incoming and outgoing edge; (4) if it contains an AND node, it contains <u>all</u> incoming and outgoing edges; and (5) if it contains an OR node, it contains exactly one incoming and outgoing edge. An example of a complete path in Fig. 3 is the header, the footer, information nodes A,B,D,J, and connector nodes a,b,c,d,g,k,l,n.

Expressions in information nodes can be of two basic types: assignment and restriction. An example of the assignment type (node D in Fig. 3) is RES.date $\leftarrow$ APPT.date. An example of the restriction type (node B in Fig. 3) is APPT.who <u>in</u> R1, where, in this case, R1 is the result of a query against the database.

A verbgraph supports NL update in the following manner. Assume we have only a single sense for each verb. When a user update command is entered, information is first extracted from the command, classified by domain and used to instantiate elements of the information nodes. We then examine the graph to see if a unique path has been determined. If not, we generate from the graph a question whose response further constrains the possibilities. Once a unique complete path is determined, the information in that path is used to instantiate the parameterized DML in the footer.

The verbgraph SCHEDULE-APPOINTMENT in Fig. 3 is based on the following database schema:

```
EMP(name, office, phone, supervisor)
APPOINTMENT(name, date, time, duration, who, topic, location)
MAILBOX(name, date, time, from, message)
ROOMRESERVE(room, date, time, duration, reserver)
```

with domains (permissible sets of values):

DOMAIN	ATTRIBUTES WITH THAT DOMAIN
personname	name, who, from, reserver, supervisor
roomnum	room, location, office
phonenum	phone
calendardate	date
clocktime	time
elapsedtime	duration
text	message, topic

The basic variations for this verbgraph are whether the person being scheduled is in the company, whether a room should be reserved and whether ones supervisor should be notified.

Suppose we have the update command "Schedule an appointment with James Parker on April 13," where James Parker is a company employee. Interaction with the verbgraph proceeds as follows. First, information is extracted from the command and classified by domain. For example, James Parker is in domain personname, which can only be used to instantiate APPT.name, APPT.who, APPT2.name and APPT2.who. However, since USER is a system variable, the only slots left are APPT.who and APPT2.name, which are necessarily the same. Thus we can instantiate APPT.who and APPT2.name with "James Parker." We classify "April 13" as a calendardate and instantiate APPT.date, APPT2.date and RES.date with it, because all these must be the same. No more useful information is in the update request.

Second, we examine the graph to see if a unique path has been determined. In this case it has not. However, other possibilities are constrained because we know the path must go through node B. This is because the path must go through either node B or node C and by analyzing the response to retrieval R1, we can determine it must be node B (i.e., James Parker is a company employee).

Now we must determine the rest of the path. One determination yet to be made is whether or not node D is in the path. Because no room was mentioned in the query, we generate from the graph a question such as "Where will the appointment take place?" Suppose the answer is "my office." Presume we can translate "my office" into the scheduler's office number. This response has two effects. First, we know that no room has to be reserved, so node D is not in the path. Second, we can fill APPT.where in node F.

Finally, all that remains to be decided is if node H is on the path. A question like "Should we notify your supervisor?" is generated. Supposing the answer is "no." Now the path is completely determined: it contains nodes A, B and F.

Now that we have determined a unique path in the graph, we discover that not all the information has been filled-in in every node on the path. We now ask questions to complete these nodes, such as What time?, For how long? and What is the topic? At this point we have a unique complete path, so the appropriate calls to INFORM can be made and the parameterized DML in the footer can be filled-in.

Note that the above interaction was quite rigidly structured. In particular,

1) After the user issues the original command, the verbgraph instantiation program chooses the order of the subsequent data entry.

2) There is no provision for default, or optional values.

3) Even if optional values were allowed, the program would have to ask questions for them anyway, since the user has no opportunity to specify them subsequent to the original command.

We want the interaction to be more user-directed. Our general principle is to allow the user to volunteer additional information during the course of the interaction, as long as the path has not been determined and values remain unspecified. We could use the following interaction protocol. The user enters the initial command and hits return. The program will accept additional lines of input. However, if the user just hits return, and the program needs more information the program will generate a question. The user then answers that question, followed by a return. As before, additional information may be entered on subsequent lines. If the user hits return on an empty line, another question is generated, if necessary.

The following advantages accrue from letting the user volunteer information. The user may choose the order of data entry. We can now have optional values, but not have to ask questions about them. Since the user has an opportunity to volunteer any values, if he or she does not volunteer the value, a default value will be used.

From our previous example, suppose topic is optional, with default null string. Consider the following interaction under our new paradigm.

```
1 > Schedule an appointment with James Parker on April 13
```

```
2 > At 3:00pm for 15 minutes
3 >
4 > Where will the appointment take place?
5 > My office
6 > Notify my supervisor
7 >
```

The user enters the initial command on line 1. In line 2 she volunteers supplemental information. Since line 3 is empty, the program generates the question in line 4. The user responds to the question in line 5 and volunteers information at line 6. At line 7, a value for topic still has not been specified, but the user has the option. Since an empty line is entered, and all non-obligatory slots are filled in, the command interaction terminates, and the program uses the default for the optional slot. DML can now be executed.

Verbgraphs are also a means for specifying non-database operations, such as sending a confirmation letter when an appointment is made. The verbgraph can also be used to express integrity constraints on the update operation, just as functional dependencies represent constraints on states of the database. We can also easily express integrity constraints on successive states of the database. Finally, there is the opportunity for computer aided design of the verbgraphs.

We are currently considering hierarchically structured transactions, as used in the TAXIS semantic model [MBW80], as an alternative to verbgraphs. Verbgraphs can be ambiguous, and do not lend themselves to top-down design. Hierarchical transactions would seem to overcome both problems. Hierarchical transactions in TAXIS are not quite as versatile as verbgraphs in representing variants. The hierarchy is induced by hierarchies on the entity classes involved. Variants based on the relationship among particular entities, as recorded in the database, cannot be represented. Also all variants in the hierarchy must involve the same entity classes, where we may want to involve some classes only in certain variants. However, these shortcomings do not seem insurmountable.

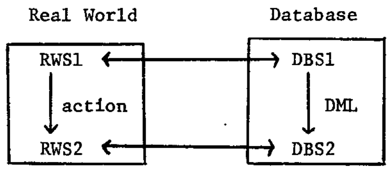


FIGURE 1

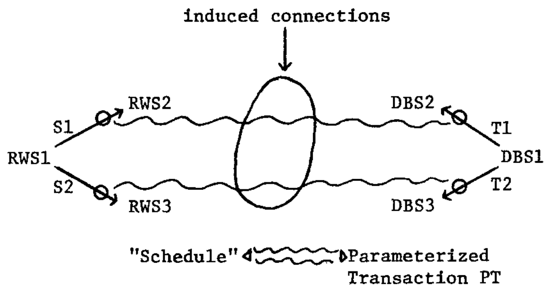
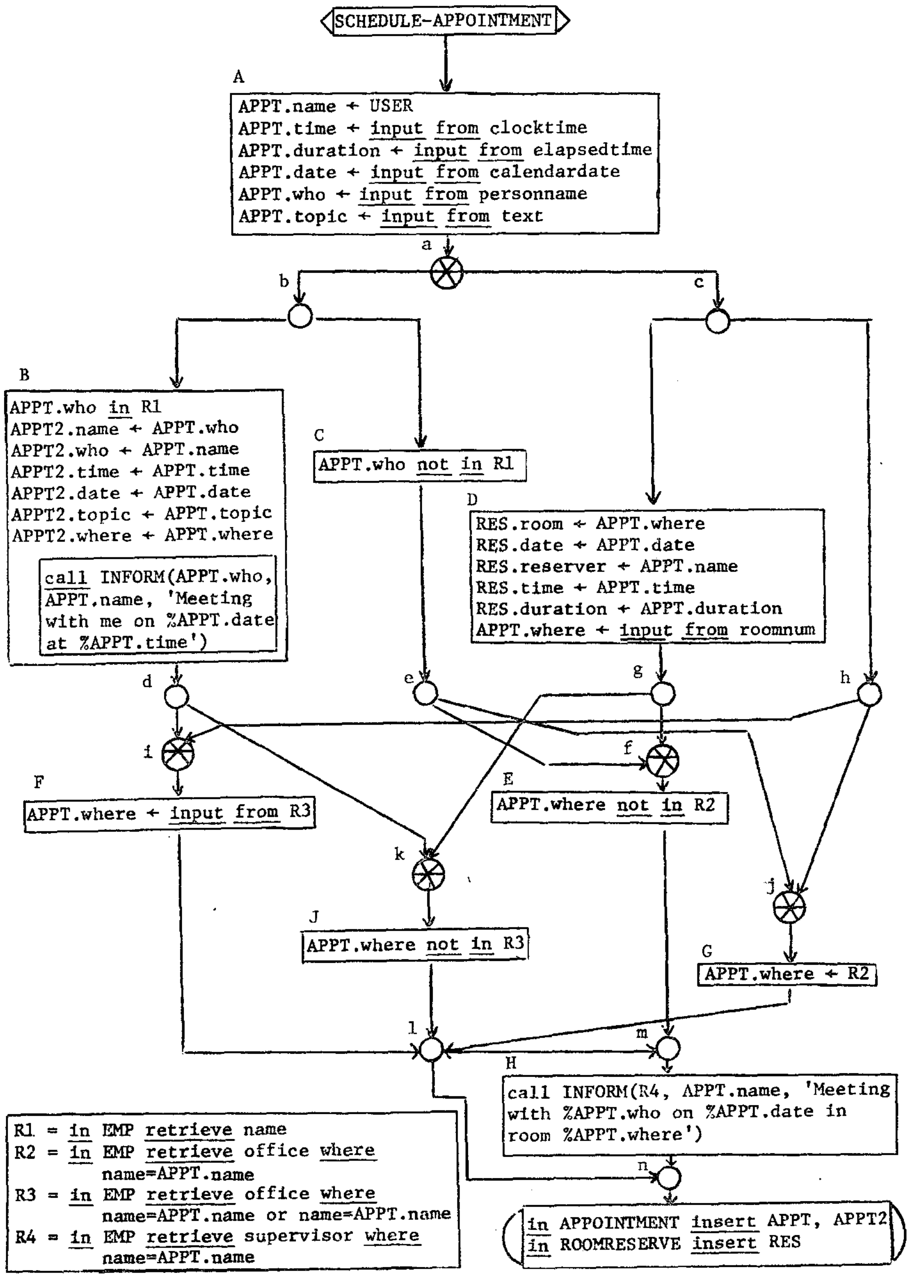


Figure 2



```
A
APPT.name ← USER
APPT.time ← input from clocktime
APPT.duration ← input from elapsedtime
APPT.date ← input from calendardate
APPT.who ← input from personname
APPT.topic ← input from text

B
APPT.who in R1
APPT2.name ← APPT.who
APPT2.who ← APPT.name
APPT2.time ← APPT.time
APPT2.date ← APPT.date
APPT2.topic ← APPT.topic
APPT2.where ← APPT.where
    call INFORM(APPT.who,
    APPT.name, 'Meeting
    with me on %APPT.date
    at %APPT.time')

C
APPT.who not in R1

D
RES.room ← APPT.where
RES.date ← APPT.date
RES.reserver ← APPT.name
RES.time ← APPT.time
RES.duration ← APPT.duration
APPT.where ← input from roomnum

F
APPT.where ← input from R3

E
APPT.where not in R2

J
APPT.where not in R3

G
APPT.where ← R2

H
call INFORM(R4, APPT.name, 'Meeting
with %APPT.who on %APPT.date in
room %APPT.where')

in APPOINTMENT insert APPT, APPT2
in ROOMRESERVE insert RES

R1 = in EMP retrieve name
R2 = in EMP retrieve office where
        name=APPT.name
R3 = in EMP retrieve office where
        name=APPT.name or name=APPT.name
R4 = in EMP retrieve supervisor where
        name=APPT.name
```

Figure 3

[Br81] Brodie, M.L., On modelling behavioral semantics of database. VLDB VII, Cannes, France, 1981.

[CH81] Carbonell, J. and Hayes, P., Multi-strategy construction-specific parsing for flexible database query and update. CMU Internal Report, July 1981.

[Da78] Damereau, F.J., The derivation of answers from logical forms in a question answering system. American Journal of Computational Linguistics, Microfiche 75, 1978, pp. 3-42.

[Ha77] Harris, L.R., Using the database itself as a semantic component to aid the parsing of natural language data base queries. Dartmouth College Mathematics Dept. TR 77-2, 1977.

[Ka79] Kaplan, S.J., Cooperative responses from a natural language data base query system. Stanford Univ. Heuristic Programming Project paper HPP-79-19.

[DK81] Kaplan, S.J., and Davidson, J., Interpreting natural language updates. Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, June 1981.

[MBW80] Mylopoulos, J., Bernstein, P., and Wong, K., A language facility for designing database - intensive applications. ACM TODS, 5, 2, 1980, pp. 185-207.

[Sa78] Salveter, S.C., Inferring conceptual structures from pictorial input data. University of Wisconsin, Computer Science Dept., TR 328, 1978.

[Sa79] Salveter, S.C., Inferring conceptual graphs. Cognitive Science, 3, pp. 141-166.

[Sk80] Skuce, D.R., Bridging the gap between natural and computer language, Proc. of Int'l Congress on Applied Systems, and Cybernetics, Acapulco, December 1980.

[Wa78] Walker, D.E., Understanding Spoken Language. American Elsevier, 1978.

[Wi81] Wiederhold, G., Kaplan, S.J. and Sagalowicz, D., Research in knowledge base management systems. SIGMOD Record, VII, #3, April 1981, pp. 26-54.

[Wo76] Woods, W., et al., Speech understanding systems: final technical report. BBN No. 3438, Cambridge, MA, 1976.