Natural Language Interaction Using a Scalable Reference Dictionary

Veera Boonjing* and Cheng Hsu**

January, 2003 for NLDB 2003

*Kbveera@kmitl.ac.th, (662) 3269982, Mathematics and Computer Science
King Mongkut's Institute of Technology Ladkrabang, Ladkrabang, Bangkok 10520
Thailand

**Hsuc@rpi.edu, (518) 276-6847, Decision Sciences and Engineering Systems,
Rensselaer Polytechnic Institute, Troy, NY 12180-3590
(**Please send correspondences to Cheng Hsu.**)

Abstract

A truly natural language interface to databases also needs to be practical for actual implementation. We developed a new feasible approach to solve the problem and tested it successfully in a laboratory environment. The new result is based on metadata search, where the metadata grow in largely linear manner and the search is linguistics-free (allowing for grammatically incorrect and incomplete input). A new class of reference dictionary integrates four types of enterprise metadata: enterprise information models, database values, user-words, and query cases. The layered and scalable information models allow user-words to stay in original forms as users articulated them, as opposed to relying on permutations of individual words contained in the original query. A graphical representation method turns the dictionary into searchable graphs representing all possible interpretations of the input. A branch-and-bound algorithm then identifies optimal interpretations, which lead to SQL implementation of the original queries. Query cases enhance both the metadata and the search of metadata, as well as providing case-based reasoning to directly answer the queries. This design assures *feasible solutions* at the termination of the search, even when the search is incomplete (i.e., the results contain the correct answer to the original query). The necessary condition is that the text input contains at least one entry in the reference dictionary. The sufficient condition is that the text input contains a set of entries corresponding to a complete, correct single SQL query. Laboratory testing shows that the system obtained accurate results for most cases that satisfied only the necessary condition.

Keywords: database query, natural language interaction, metadata search, Metadatabase, reference dictionary

1. The Problem of Natural Language Database Query

A truly natural language interface for database query has many important promises; but they all predicate on its being sufficiently ***feasible*** (either in the naturalness or in the scalability) for actual applications. This feasibility is still a hard problem in the field. To illustrate natural language database query, we list below a number of actual queries from mocked end users against a laboratory Computer-Integrated Manufacturing (CIM) database that we tested in the research. Many of them were incorrect or incomplete for the English language, but could happen easily in actuality.

- Get billing address of John Smith.
- Get orders of John Smith
- Just give me everything we have about orders of John Smith and Jim Kowalski.
- I'd like to know customer names who order PZ1.
- PZ1 customers
- How about PZ1 orders?
- Look at John Smith's orders, now give me order_id, part_id, work order quantity, and num_completed of his orders.
- Give all order_id and models that John Smith's orders.
- Do we have records about John Smith, Jim Kowalski, and William Batt?; they are our clients.
- Not done orders
- Now I want to know cost of PZ1 and PZ10
- What models did John Smith order?
- Go to Tommy Lee's records of orders; list his order_id and order status.
- Make a list of orders that are not done yet. Just give me order no. and customer names, ok?
- Now give me the details of William Batt's orders that are not done.
- Which parts of orders are milling and assembly.
- Show the details of order no. 00008 and which customers placed this order.
- Order 00010: what is its due date?
- Is Jim Kowalski's billing address and shipping address the same?
- Find customer names whose orders are not done; please include each customer's billing address
- For all orders: what are the parts associated with order no. 00006?
- Part pz1, pz2, ba, and bb;
- William Batt's orders, please.
- Find all 'PZ3' order information made by Tommy Lee.
- I have asked you many times and I still have not heard a clear answer. Listen, I would like to know the status of my order. As I know, my order no is 00009. Please make sure to include part id and part status for each part in the order. Could you please help? Thanks.

- Step on John Smith's orders. I want to know the exact status of all parts he ordered including their descriptions, costs, and quantities completed to-date. I understand that you're supposed to deliver it on 12/31/99. I know I might not have given you all the information you need, but please do your best, anyway. Okay? I'm waiting...
- How about part PZ1?
- All customers.

In these truly natural queries, not only multiple different interpretations exist for the same expressions, the interpretations might not even contain sufficient data to complete the query for processing on the database. Thus, the two basic problems facing natural language database query are ***ambiguities in intent and specification***, and both require the system to possess deep and broad metadata about the underlying database to resolve.

A significant portion of previous results in the field develops specific concept-form-syntax models either for the users to use explicitly (imposed templates) or for the applications to target implicitly (presumed patterns). These models provide precise interpretation and mapping to traditional database query languages when the user input matches exactly. Thus, they control the ambiguities at the expense of naturalness. We might classify these results into five categories according to the technical nature of their linguistic models. They include (1) the template-based approach [e.g., Weizenbaum, 1966; Wilks, 1978; Shankar and Yung, 2001], (2) the syntax-based approach [e.g., Wood, 1970, 1978; 1980; Adam and Gangopadhyay, 1997], (3) the semantics-grammar-based approach [e.g., Hendrix et. al., 1978; Waltz, 1978; Codd, 1978; Codd et. al., 1978; Templeton and Burger, 1986; Owei, 2000], (4) the intermediate-representation-language-based approach [e.g., Warren and Pereira, 1982; Brajnik, et.al., 1986; Bates et.al., 1986; Gross et.al., 1987; Wu and Dilts, 1992], and (5) the concept-based models such as MindNet [Richardson et. al., 1998], WordNet [Fellbaum, 1998], and conceptual dependency [Schank, 1973, 1975]. Some recent works combine these results [Metais 2002] and have also tried to offer more naturalness by narrowing the application domain and engaging more natural language forms and rules [e.g., Zue, 1999; Rosenfeld et.al., 2000; Shankar and Yung, 2001].

Another fundamental approach developed previously is to use a general linguistics-string model to allow for interpretation of free-text queries. A prevailing idea of such models is the n-gram-based dictionary, containing all possible uses (permutations) of words found in all past queries to interpret new queries. It then maps the interpretation to a semantic representation of the database structure and thereby determines the executable queries for the original natural language input. Examples include many approaches using relation, object, tree, space vector, probabilistic measures, semantic parsing, and other designs to relate concepts to strings [Maier, 1983; Maier and Ullman, 1983; Wald and Sorenson, 1984; Johnson, 1995; Meng and Chu, 1999; Zhang et. al., 1999; Kim 2000; Jarvelin et.al. 2001; and Mittendorfer and Winiwarter 2002]. Some other results limit the design to particular applications in order to control the size of the models [e.g., Janus 1986; Motro 1986, 1990; Shimazu et. al.,

1992; Guida and Tasso 1982; Meng and Chu 1999; Zhang et. al. 1999; ask.com 2002 – more in the References]. A concern for these results is their scalability. When the linguistics used are simplistic, such as containing only synonyms to the database objects, the question of multiple interpretations - i.e., the ambiguity in intent - looms large. When, on the other hand, the models are theoretically comprehensive enough to deal with ambiguities, they may become too costly to implement for non-trivial applications. The size of an n-gram dictionary could increase exponentially as the number of queries and users increases.

In all these approaches, incomplete input could still cause ambiguities in specification and/or intent, if the systems do not possess sufficient metadata to "complete the picture" for the users. Furthermore, a truly natural language interface can never guarantee complete success for all uses at all times - even humans routinely misunderstand other humans. Thus, it has to be able to always yield a feasible solution that contains the correct answer plus some additional but relevant information, and to continuously improve its performance through experience: obtaining (online) users' responses and/or the cases on the users. It follows that interaction with users is always critical for the system to close the loop and assure its validity. However, to make the interaction meaningful, the system has to possess sufficient metadata in order to provide a useful reference scheme on which to base the interaction (e.g., dialogue). Previous results have shown the need for sufficient interaction capability to achieve the above goal [Mylopoulos et.al. 1976; Codd 1978; Codd et.al. 1978; Waltz 1978; Kaplan 1984; Carbonell 1983; Rissland 1984; Norcio and Stanley 1989; Miller 1975; Soderland 1997; Mooney 1997-2001; and Ruber 2001].

Therefore, we submit that ***open***, ***deep and scalable metadata*** about both the structure and application of the databases is a key to solving the problem of natural language query. With sufficient metadata, the system could resolve ambiguity in specification and support a reference dictionary capable of resolving ambiguity in intent that grows linearly. Moreover, with them, the system could develop an efficient interaction with the user to assure a feasible closure for query processing as well as to effect continuous improvement. We envision the following metadata: query cases as in case-based reasoning, an enterprise information model expandable to include any number of semantic layers, database objects and values, and open user-words recognized for the information model and database objects and values. We then develop a ***feasible region*** interpretation approach using metadata to achieve natural language database query, as described in Section 2 below. We present the core method of the new approach next in Section 3, the laboratory testing in Section 4, and discuss some proposed new research afterwards in the concluding section.

2. The Metadata Search Approach

The four types of ***metadata***: ***query cases, enterprise information model, database objects and values, and user-words***, mutually support each other in an integrative manner. Database objects (structure, operators, etc.) and values are what the database is; but users have to refer to them in one way or another in their queries. Enterprise

information models represent well-defined applications and the basic concepts of these applications for the underlying databases. An information model could be flat, representing only the immediate database structure and hence becomes fundamentally not scalable; but it could also include scalable layers of semantics on top of the database structure. The more scalable and layered, the more completely this subset of metadata captures the concepts users use to make queries. Users would, however, use their own words to refer to these concepts and database objects and values when they articulate a request against the database. The individual phrases and words user use in their original forms (not permutations of the original words that the system derives and generates) are the user-words. Query cases are past examples of user queries and their correct results. They should include minimally the exceptions - i.e., the cases where the system would have to engage an interaction with the users to obtain the correct answers. Thus, a case could include user, application and other pertinent information to assist its future use in interpreting the queries from perhaps the same user, the same application and the like. In this sense, cases close the gap between the metadata collected and the ones required at any given point of time. The system could use cases to interpret natural queries directly, if necessary, in a case-based reasoning manner. Clearly, database values, user-words and cases would grow as the use of the database continues; but the growth is proportional linearly to the actual queries processed. Other types of metadata could grow, too, to a much less extent. The scalable enterprise information model is at the core of the metadata. It allows the system to infer the missing information to complete the incomplete queries; and it minimizes the need for user-words. Together with cases, they make it possible for a sufficient reference dictionary to avoid using n-grams. All together, they promise a feasible solution to resolving the ambiguities in intent and specificity.

A natural query is reducible to the set of user-words, concepts, database objects and values it contains. Conversely, a particular combination of these keywords could represent a particular natural query. However, the set of keywords may not always correspond to a unique and complete database query that the system can process correctly. The reasons include possible multiple mappings of the keywords to database objects/values, and incomplete keywords that the systems can complete in multiple ways. This is where the ***search*** approach comes into play. We first store the metadata into a reference dictionary and manage it as a Metadatabase (see the next section), and then use networked graphs to represent their ***logical structure***. An interpretation of a set of keywords is an image of the set on the logical structure. The approach identifies implicitly all permissible interpretations (including the ones derived from the reference dictionary for incomplete input) for the set from the dictionary, and evaluates their relative merits to optimize the interpretation for the query using the branch-and-bound method. The optimal interpretation then leads to a query using the underlying database query language (such as SQL). The above approach is free of the need to understand linguistically the human meaning of the words and phrases used. It needs only one keyword to start the search process, since it could infer possible (multiple) interpretations from the graphs of the logical structure that contain it. When the original user query contains a complete set of metadata free of ambiguity, such as the ones that satisfies an SQL programmer, no derivation is necessary and a solution is readily available. Among all possible interpretations for a set of keywords, the one that involves least addition of

derived keywords (including zero) should be the one most accurate to the user's intent. The argument here is simple: users would not be deliberately wordy with respect to their own standards, nor misleading, when querying a database. Therefore, minimal sufficiency is a good search criterion while the logical structure is the constraint of the metadata search algorithms, for a constrained optimization paradigm.

The core of the reference dictionary (enterprise information model and initial user-words) will be a design-time product; while cases, additional user-words, changes to the information model, and database values will be added as the database system evolves. The reference dictionary-based logical structure holds four fundamental promises: it generates search-ready graphics; it supports case-based reasoning; it assures complete interpretations of natural queries; and it simplifies user-words. The last point is worth further elaboration. Consider the theoretical complexity of processing only one natural query. Suppose the text consists of a stream of **n** words of which **m** are recognized metadata. There would then be **n/m** words associated with each known term; these words become the candidate user-words for the known terms. The expected number of permutations of these new words would be **m*(n/m)!** for the query. As **n** grows over time with new queries, the number of user-words would grow in this rate - and this is the general complexity of a linguistic dictionary. Obviously, the system wants to increase the number **m** (hits) since the bigger **m** is, the fewer (exponentially) the possible groupings of words would be, resulting in fewer new user-words considered or added to the dictionary. Information models with rich (layered) semantics provide a large **m** for the initial design of user-words, and consequently lead to less ambiguity, fewer possible interpretations, and fewer new user-words. When **m** reaches a reasonable scale, permutations of user-words become less necessary. Cases do not directly change **m**, but do help resolve ambiguity due to insufficient user-words, and hence help to reduce the need for new user-words. Information models and cases represent a tightly structured, efficient kernel of meaning with which the users will be familiar.

The core logic of the metadata search approach proceeds this way. Given a graph **R = <V, E>** of the reference dictionary where **V** is a set of vertices consisting of cases (**C**), user-words (**U**), information models (**M**), and database values (**D**); and **E** is a set of edges connecting them:

Step 1: Identify an ordered k-tuple $\mathbf{I}_{keyword} = (t_1, t_2, \ldots, t_k)$ where t_i is a word/phrase (keyword) in the input that also belongs to **U, M or D** of **R**; i = 1, 2, …,k; and k is a number of keywords found in the input. Associated with each element t_i is the set of referenced **M or D** (each element of $\mathbf{I}_{keyword}$ may refer to many elements in **M or D**). Denote this set as V_i. Therefore, we have an ordered k-tuple $\mathbf{V}_{keyword} = (V_1, V_2, \ldots, V_k)$ where V_i is a set of referenced **M or D** for t_i.
Step 2: Determine the most similar past case by matching keywords of the query with problem definitions of past cases. If a perfectly matched case is found, then apply the case solution and go to Step 5. Otherwise, if the similarity measure of the most similar case is sufficiently significant (say at least 60%), then modify the case to obtain a solution and go to Step 5.
Step 3: Determine a minimal combination set of elements of $V_1, V_2, \ldots, V_k$.

Step 4: Search for the best interpretation by using the branch and bound method.
Step 5: Map the result to the database query language. Obtain the results of the query and confirm them with users.

Note that the case-based learning mechanism engages user in dialogue as needed. Its outcome becomes new cases and user-words added to **C** and **U**, respectively. The above logic consists of a few postulates.

Postulate 1: Necessary Condition
The natural query input contains at least one keyword found in **R**.
Postulate 2: Sufficient Condition
The natural query input contains an unambiguous set of keywords necessary for forming an accurate SQL statement for the query.
Postulate 3: Optimality
When all other things are equal, the best interpretation of the natural query input is the one requiring minimum traversal on the logical structure of **R**.
Although the sufficient condition guarantees the logic to succeed, the approach actually works under the necessary condition in most cases in our testing. Postulate 3 also allows for adding new criteria such as operating rules for the database to the search.

3. The Core Methods and Algorithms

Reference Dictionary of Metadata

We use the Metadatabase model [Hsu et. al., 1991] to create the reference dictionary and integrate all four types of metadata. Previous works have shown the model to be scalable (i.e., metadata independence) and established its feasibility and appropriateness for meeting the requirements of this research [Hsu, et.al., 1991 and Hsu 1996]. The other benefits of using this model include its capability to incorporate rules [Bouziane and Hsu, 1993, 1997; Babin and Hsu 1996] and to support global query processing across multiple databases [Cheung and Hsu, 1996]. A modeling system helps the development and creation of the Metadatabase [Hsu, et.al. 1993; Hsu 1996] using the Two-Stage Entity-Relationship (TSER) modeling Method [Hsu et.al., 1991]. This research expands the original metadata representation to include database values, user-words, and cases in an integrated (connected) structure for the entire reference dictionary.

The icons of the reference dictionary, shown in Figure 1, represent either a table of metadata or a particular type of integrity control rules. The metadata include subjects and views, entity-relationship models, contextual knowledge in the form of rules, application and user definitions, database definitions and database values. User-words are defined as ordered pairs (class, object). Classes include Applications, Subjects, EntRels (entity-relationship), Items, Values, and Operators; all of which are metadata tables as shown in Figure 1. Objects are instances (contents) of these classes. An object is uniquely identified by an ordered quadruple (Item name, EntRel name, Subject name, Application name) as well as an identifier. A case in the case-based reasoning paradigm typically consists of three components: a problem definition, a solution, and its outcome

[Kolodner, 1993]. In this research, the reference dictionary contains the complete domain knowledge needed, so the problem definition is expanded but the outcome is dropped. New problems would use the problem definition to find the (best) matching cases and apply the associated solutions to them. A set of keywords and their recognized vertices for a query describes the problem definition, and its interpretation defines the solution.

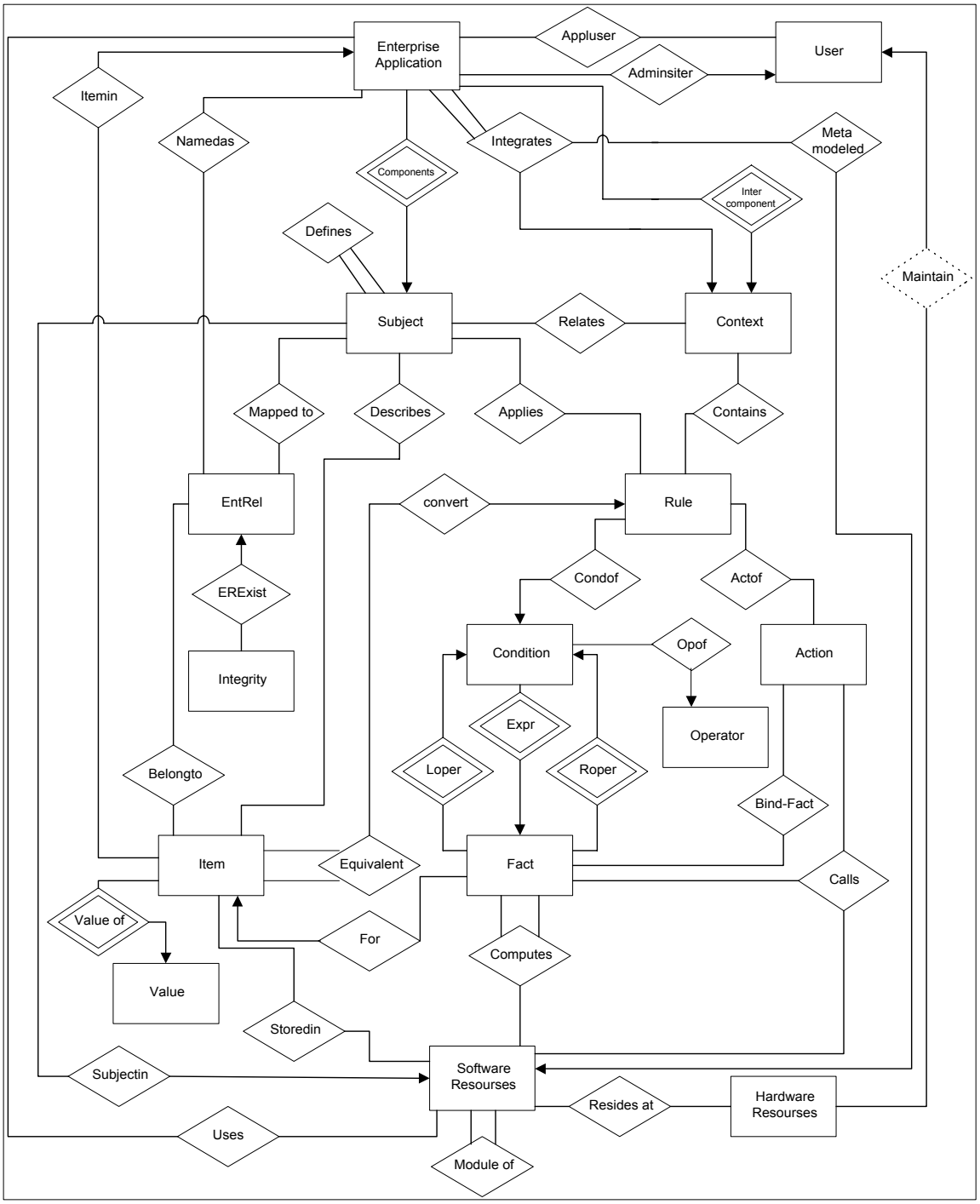

Figure 1: The structure of reference dictionary

Graphical Representation of Natural Language Queries

Interpretations of a natural language query are defined on a graph G (Definition 1), a sub-graph of the reference dictionary graph. Given a *natural language query* Q (Definition 2), the natural language interface performs an interpretation in several steps. It first scans Q to recognize the keywords (entries in the reference dictionary) in the natural query, i.e., the *recognized keywords* (Definition 3). It then determines their corresponding *recognized vertex sets* (Definition 4) in G and identifies all *query images* (Definition 5) of Q on G based on these vertex sets. Since Q may be ambiguous (e.g., incomplete and multi-valued mapping) to G, each of its recognized keywords could correspond to multiple recognized vertices, resulting in multiple query images. Further, a recognized vertex may not always connect to other recognized vertices in a way covering a complete range of data semantics (database value, attribute, entity, and relationship) with a unique path. Therefore, it could have multiple *semantic paths* (Definition 6). Taking these data semantics into account, we obtain all possible semantic paths for a query image, called the *feasible graphs* (Definition 7). The refinement of feasible graphs leads to *connected feasible graphs* (Definition 8) and *complete query graphs* (Definition 9). *A complete query graph represents an executable interpretation of the natural language query Q according to the logical structure G*. The optimization algorithm implicitly searches all possible interpretations to determine the final query graph for execution.

Definition 1: A graph G is a graph <V, E>, where sets V and E are defined on the reference dictionary. In particular, V is a set of vertices of five types: subjects, entities, relationships, attributes, and values; and E is a set of their connection constraints (owner-member and peer-peer associations). Owner-member constraints belong to two types: subject-(sub)subject-entity/relationship-attribute-value and subject-attribute. Peer-peer constraints belong to three types: entity-entity, entity-relationship, and relationship-relationship.

Definition 2: A natural language query Q is a string of characters segmented by spaces.

Definition 3: A recognized keyword t_i of Q is a segment of Q matching some entries in the reference dictionary or some vertices in graph G.

Definition 4: A recognized vertex set of a recognized keyword t_i , V_{t_i} , is a set of vertices of G that matches t_i. A member of V_{t_i} is a recognized vertex.

Definition 5: Given an n-recognized-keyword query where $n \neq 0$, a query image in graph G is a set of recognized vertices v_i where $i = 1, \ldots, n$ and $v_i \in V_{t_i}$, respectively.

Definition 6: A semantic path of a recognized vertex v_i is a minimum set of vertices in G containing v_i that satisfies the following conditions: it contains an entity/relationship vertex and its vertices are connected. The vertices it contains, other than v_i, are all implied vertices by v_i to provide an interpretation (semantics) of v_i.

Definition 7: A feasible graph of an n-recognized-vertex query image, where $n \neq 0$, is a collection of semantic paths sp_i where $i = 1, \ldots, n$ and sp_i is a semantic path implied by a recognized vertex v_i of the query image.

Definition 8: A connected feasible graph is a connected sub-graph of G containing the feasible graph and a collection of entity/relationship vertices and edges in this graph. This collection is minimally sufficient to connect entity/relationship vertices of the feasible graph. The path connecting these entity/relationship vertices is called entity/relationship solution path.
Definition 9: A query graph is a subgraph of connected feasible graph. It consists of <V, E> where V is a set of entity/relationship vertices, attribute vertices, and value vertices and E is a set of edges connecting them.

The graphical representation method yields a feasible region poised for search. As long as the search method employed is bound within the region, then any results obtained are guaranteed to be a feasible solution relevant to the original query. The solution may not offer pinpoint precision, i.e., may contain additional information not requested in the original query, but will nonetheless include the correct answer and the additional information will be meaningful. A case would be a query to get the cost of part ID 111222 that returned an answer of all information for the part including the cost. The answer is clearly feasible and useful to the user.

Query Interpretation Algorithm

The search algorithm itself follows the Branch-and-Bound logic. Given an input I = (w_1, w_2, …, w_n), where n is the number of words in a natural language query, the search proceeds as follows:

Step 1: Determine recognized keywords and their recognized vertex sets.

Step 1.1: Determine a set of recognized keywords in the input (we denote this set as $I_{keyword}$) such that these keywords are elements of or indirect references to elements of the graph G. Therefore, we have $I_{keyword}=(t_1, t_2, \ldots, t_k)$ where k is the number of keywords formed from n words of I and $k \leq n$.

Step 1.2: Determine an ordered k-tuple $V_{keyword}=(V_1, V_2, \ldots, V_k)$ such that V_i is a set of recognized vertices of its correspondent $t_i \in I_{keyword}$ where i=1, 2, ..,k.

Step 2: Determine a minimal set of query images.

Step 2.1: For each $V_i \in V_{keyword}$ where i=1, 2, ..,k and $|V_i| \geq 2$, determine $minV_i$ such that there does not exist an entity/relationship or subject vertex $\in minV_i$ belonging to a subject vertex $\in minV_i$. Therefore, we have an ordered k-tuple $V_{min}=(minV_1, minV_2, \ldots, minV_k)$.

Step 2.2: Determine $V_{minVertexSets}$ of V_{min} such that it contains no $minV_i$ that $|minV_i|=1$ and its element is an entity/relationship or subject vertex belonging to $minV_j$ where $|minV_j| = 1$, $i \neq j$, and j=1,2, …, k. Therefore, we have $V_{minVertexSets}=(minV_1, minV_2, \ldots, minV_{k^*})$ where $k^* \leq k$. Corresponding to $V_{minVertexSets}$ is $I_{minKeywords}=(t_1, t_2, \ldots, t_{k^*})$.

Step 2.3: Determine a query image set QI such that QI = {QI_i | QI_i is a set of k* recognized vertices, $\{v_{1,i} \in minV_1, v_{2,i} \in minV_2, \ldots, v_{k^*,i} \in minV_{k^*}\}$; i = 1, …, n ; n = $|minV_1||minV_2|\ldots|minV_{k^*}|$ where $|minV_j|$, j = 1, 2, …,k*}.

Step 2.4: Determine a minimal set $minQI_i$ for each $QI_i \in QI$ such that $minQI_i$ = { v_j | for all entity/relationship or subject vertex v_j , there does not exist subject vertex v_l such that $v_j \in v_l$ where $j \neq l$; j, l = 1, …, k**; $k^{**} \leq k^*$}.

Step 2.5: Determine a minimal QI set, QI_{min} = { $minQI_i$ | for all $minQI_i$, there does not exist $minQI_j$ such that $minQI_i$=$minQI_j$ where $i \neq j$; i, j = 1, …, n*; $n^* \leq n$}.

Step 3: Search for the best interpretation (query graph).

Search for the query graph QG_k such that $cost(QG_k) = min(cost(QG_1), cost(QG_2), …, cost(QG_m))$ where QG_1, QG_2, …, QG_m are all possible query graphs enumerated from QI_{min} and $cost(QG_i)$ = (number of vertices of QG_i) – 1 where i = 1, 2, …, m. Since there could be multiple query graphs satisfying this condition, we have QG_{opt} = { QG_k | $cost(QG_k) = min(cost(QG_1), cost(QG_2), …, cost(QG_m))$ and $k \in \{1, 2, …, m\}$}.

Step 4: Remove equivalent query graphs in QG_{opt}.

If $|QG_{opt}| > 1$, then determine QG_{case} from QG_{opt} such that for all $QG_i \in QG_{case}$ there does not exist $QG_j \in QG_{case}$ that (1) entity/relationship solution path of QG_i is equal to entity/relationship solution path of QG_j, (2) set of attribute vertices of QG_i is equivalent to set of attribute vertices of QG_j, and (3) set of value vertices of QG_i is equivalent to set of value vertices of QG_j.

Step 1 aims at text processing and is self-evident. Step 2.1 removes redundant recognized vertices for each keyword using owner-member relationship subject-(sub)subject-entity/relationship. This step maximizes semantic domains and minimizes possible query images. Step 2.2 employs a similar idea to remove redundancy between unambiguous keywords. When all recognized vertices for a keyword are removed, the keyword is removed too. Step 2.3 enumerates query images from sets of minimal recognized vertices obtained from Step 2.2. Step 2.4 removes redundant elements in each query images based on owner-member constraints (i.e., subject-(sub)subject-entity/relationship. Step 2.5 removes identical query images to minimize the number of enumerated query graphs. Overall, Step 2 is important for metadata search when the enterprise information models include layers of semantics (subject hierarchy in this model or object hierarchy in general) on top of database structures. Layered semantics helps resolve ambiguity (e.g., completing the query), but also leads to redundant interpretations. The search problem in Step 3 is an optimization problem with objective function z(t) where t is a terminal vertex (a query graph). The problem is to minimize z(t) with respect to graph G. An evaluation function LB(v) finds the lower bound for an intermediate vertex v (a query image or a feasible graph), so that the search could either fathom all paths starting with v or pick the most promising v to explore further. This lower bound of a vertex indicates the best minimum-cost query graphs that could be obtained if the vertex is picked to explore further.

Database Query Language Generation

The next step is to map the final query graph into the target database query processing language. This research has developed the query generator for the Structured Query Language (SQL). The query generator infers from the query graph the entities and relationships and their join conditions that the query involves. It then determines the database objects and values required, and puts these operators and parameters into an SQL statement.

The SELECT list is determined by rules based on appearance of attribute vertices in a query graph as follows.

- If the query graph contains attribute vertices, the select list contains those attribute vertices and all attributes of value vertices.
- If the query graph does not contain any attribute vertices, the select list contains all attributes that belong to entity/relationship vertices and all attributes of value vertices.

The selection condition is determined by rules based on appearance of value vertices in a query graph as follows.

- If v_1 belong to attribute A_1 of entity/relationship R_1; v_2 belong to attribute A_2 of entity/relationship R_2; …; and v_n belong to attribute A_n of entity/relationship R_n, then the selection condition is "($R_1.A_1 = v_1$ AND $R_2.A_2 = v_2$ AND … AND $R_n.A_n = v_n$)."
- If v_1, v_2, …, v_n belong to attribute A of entity/relationship R , then the selection condition is "($R.A = v_1$ OR $R.A = v_2$ OR … OR $R.A = v_n$)."

Case-Based Reasoning and Interaction

In theory, the system could accumulate all natural queries it has processed (using the above method) into a case-base. When the case-base has grown to a sufficient size, the designer would have a choice to shift to relying mainly on case-based reasoning to answer new queries and uses the interpretation algorithm only as a backup. This design would be reasonable since the growth of cases is linear. However, in this research, we still regard initially cases as a secondary tool to support the primary means of metadata search. As such, the role of case-based reasoning is to assure a closure to the user and in a sense a learning capability to the system. Therefore, we consider in this paper only the need to build new cases for natural queries that the system failed to provide a single (query graph) answer or the user rejected the answer. An interaction with the user is the means to achieve this end. The interaction capability is more a last defense than a regular way of solution; however, it could improve the performance in the worse than average cases when the repository of the cases grows. When an interaction becomes necessary, the system asks the user for additional information to solve the query correctly. There are three possible causes of failure: (1) incorrect recognized vertex for a recognized keyword, (2) no recognizable keywords in a query, and (3) insufficient keywords. A metadata-based interaction identifies the cause and guides the user to define new user-words to fix it. For the first cause, it provides the pertinent metadata (information model) describing the possible recognized vertices for the keywords found so that the user could either designate the intended vertex for a keyword or redefine the keywords. For the second cause, it walks the user through the enterprise information model from high level down to associate new user-words to vertices of information models. For the last cause, it similarly guides the user to provide additional keywords for the query. In all these cases, the interaction adds new user-words as well as building cases to the reference dictionary. The system will repeat this cycle until it yields a correct result. Another design to resolve the case of multiple query graphs for a query is to present all possible solutions to users in terms of natural language for them to choose the intended solution. If the number of

alternative solutions is relatively large, it presents all possible scopes of solutions for users to choose to narrow down the possible solutions before asking them to choose the intended solution. When the system obtains the correct result, it completes the case with problem definition, solution definition, and other information according to the structure of cases in the reference dictionary. The method of case-based reasoning uses the cases built and accumulated either to assist the metadata search or to answer directly the queries. The method includes (1) a matching mechanism to match a query with a case, (2) criteria to decide whether to reuse the most similar case, and (3) a mechanism to reuse the case. We adopt standard results in the information retrieval field, with modifications to accommodate equivalent keywords, to match a query with a case, viz., the vector space model [Salton and Mcgill, 1983] and its COSINE measure of similarity. The actual design will be a major task of the proposed research.

4. Verification at a Laboratory

We implemented the core algorithms in a software system running on a UNIX-Oracle Server. The reference dictionary and the application database - a Computer-Integrated Manufacturing (CIM) system - both reside in Oracle. The software itself has two components: the user interface, coded in HTML and Java Script to be compatible with major Web browsers, and the core algorithms, using Java Applets embedded in the user interface in a browser-based computing manner. The implementation developed so far solves primarily natural language queries that have a basic SQL solution. That is, the basic SQL uses search-conditions in the WHERE clause that either (1) are an expression of the form "$attribute_1$ = $attribute_2$" or "attribute = value," or (2) a combination of such expressions with the logical operators "AND" and "OR." These results are sufficient to show the feasibility of the metadata search approach but are insufficient for implementing the approach in practice.

We tested the prototype system with 10 users to substantiate the feasibility of the metadata search approach. The participants included professionals not associated with Rensselaer as well as undergraduate students and graduate students at Rensselaer. Their background in databases ranges from strong to virtually none. We provided the participants with the descriptions, information models, and contents of the CIM database; but were otherwise completely detached from them so as to allow them to ask as many questions as they wished against the database in any text forms. An answer to a question is correct when it includes all information the user required. In this implementation, the reference dictionary of the testing case included only information models, database values, and less than 200 user-words that we created by ourselves in advance.

The system solved every query listed in the first section, The Problem. It used a limited number of new user-words to solve every additional query listed below. It failed originally to produce correct answers to these queries; however, it solved every one when the boldface words/phrases that the users articulated were added to the reference dictionary as user-words (only one entry for each word or phrase). Without them, the system produced either multiple interpretations for the user to select or a single

incomplete interpretation. Either cases contained the correct interpretation. There were no other queries users generated that the system failed to solve correctly.

- William Batt's **contact**
- Michael Goin **address**
- Not done yet **transaction**
- **Buyers** of PZ10.
- How many **people** order pz1?
- Is Jim Kowlaski billing address and **mailing address** the same?
- What is Craig Norman billing address and **home address**? Please also list all of his orders.
- What are the **products** associated with order no 00001?
- Show the detail Craig Norman's order, part description, quantity, and **where to ship** it?
- Find the information [order_id, customer_id, desire_date] of order, which is **not completed**.
- List all **in-process** orders and their details.

The important fact about these new user-words is their consistency with the basic logic of metadata search: they do not require n-gram and their number grows linearly. The result shows, the concept of metadata search is sound and the approach performs as designed. It also supports the claim that enterprise information models with layered semantics reduce the number of user-words required and enhance the metadata search.

All queries tested satisfy the necessary condition and the sufficient condition postulated in Section II. The response times of all queries were between 1 and 10 seconds, running the software from a Web client site against the Oracle server.

5. Extending the Current Results

Many scholars consider natural language interaction a major pillar of Information Technology in the new century. However, the field has not been as active as it should be, because in part of past frustrations. The research reported here contributes to a particular field of the general area - enterprise database systems - and justifies the significance of this effort as such, as discussed in the Problem section. However, the work also promises to contribute to the quest for solution of the general problem of a natural language interaction. It might show that natural language processing is actually feasible, in the sense that it almost always produces a useful answer (feasible region-bound) to almost any relevant query, and that companies can use it at reasonable cost. The preliminary results we have obtained from laboratory tests suggest that natural language processing could apply to a significant domain such as enterprise databases, where many prospect users need completely intuitive user interfaces (allowing for ambiguous, incomplete, and incorrect articulation) fully reap the benefits of online enterprise information.. The continued research on this line promises to show the viability of having end users querying databases in their own languages. The medium of query could either be free text typed in the system or be natural sentences spoken to it. We believe the probability to be reasonably good for achieving this objective, based on our preliminary testing results.

The continuing research will cover the following areas. Foremost, the core method needs to extend. In particular, the algorithms should extend to include the full range of SQL capability, especially the other standard operators not included yet, derived-attributes and expressions, and date-related and other temporal selection criteria. We need to develop the full capability for case-based reasoning and metadata-based interaction, especially the possibility of developing a large-scale case-base to provide a parallel approach for query interpretation. Many alternative methods are also possible to achieve case matching. The interpretation algorithm and the case-based reasoning algorithms should support each other in an integrated manner, and hence require an overall design and engineering of the software system. Second, research on implementation tools and methodology to help acquire the metadata and maintain the reference dictionary is required. These tools should support information modeling, user-word acquisition, database value acquisition, and user dialog; and would use the progress in the field such as semantics acquisition and ontology to facilitate their development. Third, opportunities exist in connecting the system to voice interaction using even commercially available voice technologies to allow for voice query of databases. They also exist in the exploration for applications, especially in the Web-based settings. Finally, the possibilities of creating sufficient metadata to represent the application domain of information retrieval outside of databases deserve exploration. Research in this area would engage directly the progress on the general problem of human computer interaction.

REFERENCES

Adam, N.R. and A. Gangopadhyay, "A Form-Based Natural Language Front-End to a CIM database," *IEEE Trans. On Knowledge and Data Engineering,* 9:2, 1997, pp. 238-250.

Babin, G. and C. Hsu, "Decomposition of Knowledge for Concurrent Processing," *IEEE Trans. Knowledge and Data Engineering*, 8:5, October 1996, pp. 758-772.

Bates, M., M.G.Moser, and D. Stallard, "The IRUS Transportation Natural Language Interface," in *Expert Database Systems,* 1986, pp. 617-630.

Bergman, L., J. Shoudt, V. Castelli, and C. Li, "An Interface Framework for Digital Library," *International Journal on Digital Libraries,* 2:2-3, 1999, pp. 178-189.

Boonjing, V., *A Metadata Search Approach to Natural Language Database Query,* Ph.D. Dissertation, 2002, Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY, 2002.

Boonjing, V. and C. Hsu, "Metadata Search: A New Approach to Natural Language Database Interfaces," *Proceedings IASTED International Conference on Information Systems and Databases*, October, 2002, Tsukuta, Japan.

Boonjing, V. and C. Hsu, "A Feasible Approach to Natural Language Database Query," submitted.

Bouziane, M. and C. Hsu, "A Rulebase Management System Using Conceptual Modeling," *J. Artificial Intelligence Tools*, 6:1, March 1997, pp. 37-61.

Bouziane, M. and C. Hsu, "A Rulebase Model for Data and Knowledge Integration in Multiple Systems Environments," *J. Artificial Intelligence Tools*, 2:4, December 1993, pp. 485-509.

Brajnik, G., G. Guida, and C. Tasso, "An Expert Interface for Effective Man-Machine Interaction.," in *Cooperative Interfaces to Information Systems*, 1986, pp. 259-308.

Carbonell, J.G., "Learning by Analogy: Formulating and Generalizing Plans from Past Experience," in Machine Learning. 1983, pp. 137-161.

Cheung, W. and C. Hsu, "The Model-Assisted Global Query System for Multiple Databases in Distributed Enterprises," *ACM Trans. Information Systems*, 14:4, October 1996, pp. 421-470.

Codd, E.F, R.S.Arnold, J-M. Cadiou, C.L. Chang, and N. Roussopoulos, "RENDEZVOUS Vesion 1: An Experimental English Language Query Formulation System for Casual Users of Relational Data Bases," in IBM Reaearch Report RJ2144, 1978.

Codd, E.F., "How about Recently? (English Dialog with Relational Databases Using Rendezvous Vesion 1). In B. Shneiderman (Eds). Databases: Improving Usability and Responsiveness, 1978, pp. 3-28.

Fellbaum, C. (Eds), "WordNet: an Electronic Lexical Database," MIT Press, Boston, 1998.

Gross, B.J., D.E. Appelt, P.A. Martin and F.C.N. Pereira, "TEAM: an Experiment in Design of Transportable Natural-Language Interfaces," *ACM Trans*actions, 32, 1987, pp. 173-243.

Guida, G. and Tasso C., "NLI: A Robust Interface for Natural Language Person-Machine Communication," *Int. J. Man-Machine Studies*, 17, 1982, pp.417-433.

Hendrix, G.G. Sacerdoti, E.D. Sagalowicz, C. and Slocum, J., "Development a Natural Language Interface to Complex Data," *ACM Trans. on Database Systems*, 3:2, 1978, pp. 105-147.

Hsu, C., M. Bouziane, L. Ratter, and L. Yee, "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach," *IEEE Trans. on Software Engineering*, 17:6, 1991, pp. 604-625.

Hsu, C., Y. Tao, M. Bouziane, and G. Babin, "Paradigm Translations in Integrating Manufacturing Information Using a Meta-Model: the TSER Approach," *J. Information Systems Engineering*, 1:1, 1993, pp. 325-352.

Hsu, C., *Enterprise Integration and Modeling: the Metadatabase Approach*, Kluwer Academic Publishers, Boston, 1996.

Janus, J.M., "The Semantics-based Natural Language Interface to Relational Databases," in L. Bolc and M. Jarke (Eds), *Cooperative Interfaces to Information Systems*, pp. 143-187, Springer-Verlag, New York, 1986.

Jarvelin, K., J. Kekalainen and T. Niemi, "Concept-Based Query Expansion and Construction," *Information Retrieval,* 4:3/4, 2001, pp. 231-255.

Johnson, J.A.,"Semantic Relatedness," *Computers Math. Applic,* 29:5, 1995, pp. 51-63.

Kaplan, S.J., "Designing a Portable Natural Language Database Query System," *ACM Trans. on Database Systems*, 9:1, 1984, pp. 1-19.

Kim, W., "Corpus-Based Statistical Screening for Phrase Identification," *Journal of the American Medical Information Association,* 7:5, 2000, pp. 499-511.

Kolodner, J.L, "Case-Based Reasoning," Morgan Kaufmann Publishers, California, 1993.

Maier, D., "The Theory of Relational Databases," Computer Science Press, Maryland, 1983.

Maier, D. and Ullman, J.D., "Maximal Objects and the Semantics of Universal Relation Databases," *ACM Trans. on Database Systems*, 8:1, 1984, pp. 1-14.

Meng, F. and W. Chu, "Database Query Formulation from Natural Language using Semantic Modeling and Statistical Keyword Meaning Disambiguation," Technical Report CSD-TR 99003, 1999, Computer Science Department, University of California, Los Angeles, California.

Metais, E., "Enhancing Information Systems Management with Natural Language Processing Techniques," *Data and Knowledge Engineering,* 41:2-3, 2002, pp. 247-272.

Miller, P.L., "An Adaptive Natural Language System that Listens, Asks, and Learns," in *Advance Papers of Forth International Joint Conference on Artificial Intelligence*. pp. 406-413, Morgan Kaufmann, California, 1975.

Mittendorfer, M. and W. Winiwarter, "Exploiting Syntactic Abalisys of Queries for Information Retrieval," *Data and Knowledge Engineering,* 42:3, 2002, pp. 315-325.

Mooney, R., *Symbolic Learning for Natural language Processing: Integrating Information Extraction and Querying,* NSF Award Abstract #9704943, 1997-2001.

Motro, A., "Constructing Queries from Tokens," in *Proc. ACM-SIGMOD Int. Conf. Management Data*, pp. 120-131, ACM, Washington D.C., 1986.

Motro, A., "FLEX: A Tolerant and Cooperative User Interface to Databases," *IEEE Transactions on Knowledge and Data Engineering*, 2:2, 1990, pp. 231-246.

Mylopoulos, J. Borgida, A. Cohen, P. Roussopoulos, N. Tsotsos, Jr. and H. Wong, "TORUS: A Step Towards Bridging the Gap between Data Bases and the Casual User," *Information Systems*, 2:1, 1976, pp.49-64.

Norcio, A.F. and J. Stanley, "Adaptive Human-Computer Interfaces: A literature Survey and Perspective," *IEEE Trans. On Systems, Man, and Cybernetics*, 19:2, 1989, pp. 399-408.

Owei, V., "Natural Language Querying of Databases: an Information Extraction Approach in the Conceptual Query Language," *Int. J. Man-Machine Studies*, 53, 2000, pp.439-492.

Richardson, S.D., W.B. Dolan and L. Vanderwende, "MindNet: Acquiring and Structuring Semantic Information from Text," In *ACL'98 CONF 17*, 2, 1998, pp. 1098-1102.

Rissland, E.L. 1984, "Ingredients of Intelligent User Interfaces," *Int. J. Man-Machine Studies*, 21, 1984, pp. 377-388.

Rosenfeld, R., X. Zhu, S. Shriver, A. Toth, K. Lenzo and A. W. Black, "Towards a Universal Speech Interface," In *Proc. ICSLP 2000,* 2000.

Ruber, P., "Asking Naturally," *Knowledge Management,* 4:8, 2001, pp. 62-63.

Salton, G. and Mcgill, M., "Introduction to Modern Information Retrieval," Mcgraw-Hill, New York, 1983.

Schank, R.C., "Conceptualization Underlying Natural Language," In R.C. Schank and K.M. Colby (Eds), *Computer Models of Thought and Language*, pp. 187-247, W.H. Freeman and Co., San Francisco, 1973.

Schank, R.C., "Conceptual Dependency Theory," In R.C. Schank, *Conceptual Information Processing*, pp. 22-82, American Elsivier, North-Holland, 1975.

Shankar, AJ. and Yung, Wing, "gNarLi: A Practical Natural Language Interface," http://www.people.fas.havard.edu/~shankar2/stuff/gnari.html, 2001.

Shimazu, H., S. Arita, and Y. Takashima, "CAPIT: Natural Language Interface Design Tool with Keyword Analyzer and Case-based Parser," *NEC Res. & Develop*, 33:4, 1992, pp.679-688.

Soderland, S.G., *Learning Text Analysis Rules for Domain Specific Natural Language Processing,* Ph.D. Dissertation, 1997, Department of Computer Science, University of Massachusetts Amherst, Amherst, MA.

Templeton, M. and J. Burger, "Considerations for the Development of Natural-Language Interfaces to Database Management Systems," in L. Bolc and M. Jarke (Eds), *Cooperative Interfaces to Information Systems*, pp. 67-99, Springer-Verlag, New York, 1986.

Wald, J.A. and Sorenson, "Resolving the Query Inference Problem using Steiner Trees," *ACM Transactions on Database Systems*, 9:3, 1984, pp. 348-368.

Waltz, D.L., "An English Language Question Answering System for a Large Relational Database," *Communications of the ACM*, 21:7, 1978, pp.526-539.

Warren, D. and F. Pereira, "An Efficient Easily Adaptable System for Interpreting Natural Language Queries," *Computational Linguistics*, 8:3-4, 1982, pp. 110-122.

Weizenbaum, J., "ELIZA-A Computer Program for the Study of Natural Language Communication between Man and Machine," *Communications of the ACM*, 9:1, 1966, pp. 36-44.

Wilks, Y., "An Intelligent Analyzer and Understander of English," *Communications of the ACM*, 18:5, 1978, pp. 264-274.

Wood, W.A., "Transition Network Grammars for Natural Language Analysis," *Communications of the ACM*, 13:10, 1970, pp. 591-606.

Wood, W.A., “Semantics and Quantification in Natural Language Question Answering,” In M. Yovits (ed) *Advanced in Computers*: pp. 1-87, Academic Press, New York, 1978.
Wood, W.A., “Cascaded ATN Grammars,” *American Journal of Computational Linguistics*, 6:1, 1980, pp. 1-15.
Wu, W. and D.M. Dilts, “Integrating Diverse CIM Data Bases: The Role of Natural Language Interface,” *IEEE Trans. Systems, Man, and Cybernetics*, 22:6, 1992, pp.1331-1346.
Zhang, G. Chu, W.W. Meng, F. and Kong G., “Query Formulation from High-level Concepts for Relational Databases,” In N. W. Paton and T. Griffiths (eds), *Prodeedings in User Interfaces to Data Intensive Systems,* pp. 64-74, The IEEE Computer Society, 1999.
Zue, V., “Talking with your Computer,” *Scientific American*, 281:2, 1999, pp. 56-57.