THE ETA INTERFACE [*]

James Bradford
and Tomasz Pietrzykowski

Research Report CS-80-27

Department of Computer Science
University of Waterloo
Waterloo, Ontario
May, 1980

[*] This research was supported by NSER grant A 5267

The Eta Interface

Abstract

Natural language interfaces that mediate between data base management systems and naive users must be robust and easy to use. This paper discusses the Eta interface, an experimental system designed around a powerful error handling strategy. Eta's spelling and grammar correction facilities are illustrated. Error correction based solely on syntax may occasionally yield meaningless interpretations of the input. The use of semantic constraints to dispose of nonsensical interpretations is discussed.

Resume

Interfaces en langage naturel, doit etre robuste et simple a utiliser, afin de reconcilier l'operateur du systeme de donnees de base et les utilisateurs naifs. Cette communication traite du "Eta Interface," un systeme experimental designe autour d'une forte erreur traitant de strategie. Les facilites du correcteur d'epelation et de correction grammaticale sont illustrees. Le correcteur d'erreur qui base seulement sur la syntaxe, peut occasionellement mener a des interpretations sans sens de l'information donnee. L'utilisation des contraintes semantiques afin de dispose des interpretations sans sens est discutee.

The Eta Interface [*]

James Bradford and Tomasz Pietrzykowski
Department of Computer Science
University of Waterloo
Waterloo, Ontario

Introduction

A natural language interface must cope with ambiguous, incomplete and grammatically incorrect inputs. The error handling strategy used by an interface's parser can contribute substantially to the satisfaction or frustration felt by a data base user. The Eta interface (Eta stands for Error Tolerant Analysis) is designed around a robust error handling parser.

A number of papers describing natural language data base interfaces have been published recently. Some of the best known systems are: LADDAR [Hendrix78], The Linguistic String Project Medical Question Answerer [Grishman78, Grishman73], EUFID [Kameny78], REQUEST [Plath76], RENDEZVOUS [Codd78], PLANES [Waltz78] and TORUS [Mylopoulos76]. Of these, RENDEZVOUS and PLANES have the best error handling capabilities.

The RENDEZVOUS parser attempts to transform the user's query into a formal query by applying a number of transformation rules. Spelling mistakes are corrected during lexical analysis. Word sense ambiguity is handled during the transformation phase. Whenever a part of the input can be transformed to two or more formal equivalents, RENDEZVOUS

---

[*] This research was supported by NSER grant A 5267

engages the user in a clarification dialogue. For example it might ask whether "Houston" was the name of a city or the name of a customer. If a complete formal query is not produced by the first application of the transformation rules then "nonessential" words are dropped and the rules are reapplied. Unknown words are defined by menue selection. If all else fails, queries may be made entirely by menue selection.

The PLANES parser uses an Augmented Transition Network (ATN) and "concept case frames" to convert its input into an unordered set of "semantic constituents" (the query generator subsequently assembles these constituents into a formal query). Spelling correction is done during lexical analysis. The case frames can be thought of as abstract query patterns. If more than one case frame can apply to a particular input, then the system could[*] initiate a clarification dialogue.

Eta has been designed to be (1) error tolerant, (2) compact (it will be a microcomputer based system) and (3) easy to learn. One of the applications under consideration is the TELIDON system [Brown79]. In this application Eta would reside in the microcomputer which serves as an intelligent remote terminal for TELIDON. The intended user is the average householder. In this kind of application, error tolerance becomes crucial to user satisfaction.

---

[*] In his paper Waltz indicates that this feature has not yet been implemented.

For example, suppose the grammar for our interface accepts only simple interrogatives (eg. "Who led the liberals in 1974?"). Not knowing this, a user enters:

"The liberals has been ledd by who in 74?"

The Eta parser would encounter and surmount five problems in the preceding question. First, the entire sentence is in the passive voice, the form for which is not specified by the grammar. Second, the auxiliary verb, "has" is in the wrong person (it should be "have"). Third, the main verb, "led" contains a typographical error (an extra 'd'). Fourth, the pronoun, "who" is in the wrong case (it should be "whom"). And fifth, the date, "74" is a contraction which we will suppose is not in the lexicon.

Parsers of formal languages have been coping successfully with input errors for many years. The natural language parsing problem is sufficiently different however, to require a new approach to error handling. In particular a typical program submitted to a compiler may consist of hundreds or even thousands of statements. In contrast a typical query submitted to a natural language interface consists of only one or two statements. Thus natural language error handling techniques can afford to be more thorough [Rohrich78].

Formal language parsers usually repair syntax errors so that parsing may continue and any further errors may be caught during a single compile. The object of natural

language correction is to find a correct interpretation of the input. Thus for formal language parsers, any repair that does not produce an error avalanche is acceptable, while natural language parsers seek to discover the user's intent.

It is this latter objective that makes dialogue with the user an important part of the interface's error handling strategy. An advice taking parser represents a man/machine partnership where the user supplies intelligence and world knowledge. The system, for its part, supplies detailed knowledge of the data base query language and undertakes the mechanical aspects of query translation.

System Overview

As shown in figure 1 the Eta architecture consists of two sections, the parser and the translator. The error handling innovations are embodied in the parsing section.

Consider the problem of translating English queries about suppliers and parts into their equivalents in relational calculus [Date77]. The parsing section would take as its input queries of the form: "How many bolts do we buy from Smith?". It is the function of the parser to fit the words of the query into various syntactic categories and to identify the role of each. In this example the parser would have as its output:

```
Subject:         "we"              The Unknown: <Object Modifier>
Verb:            "buy"             The Unknown Type: <Quantity>
Verb Modifier:   "from Smith"
Object:          "bolts"
Object Modifier: "how many"
```

The Eta Interface

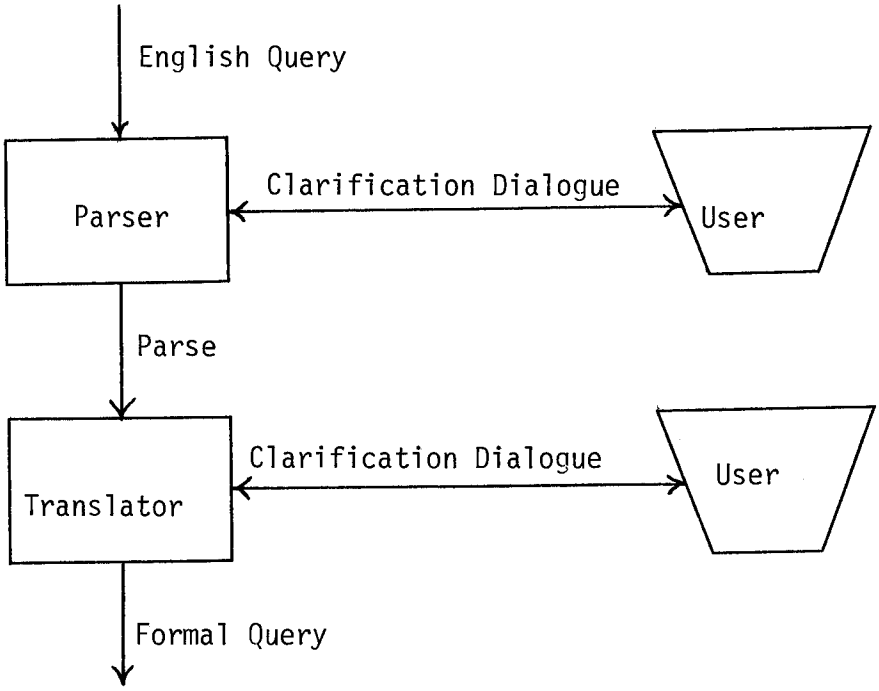

Figure 1
Eta - Architecture

It is the function of the translator to take this analysis and fit it to one or more formal query templates. The slots in the templates are filled directly from the analysed input, from the contextual store (in the case of ellipsis), from the data stored about the application area and from the user. In this example the translator would produce the following output:

"GET W(Total(SP.QTY): SP.#=S.# & SP.#=P.# & S.Sname='Smith' & P.Pname='bolt')" [*]

---

[*] In this case the attribute names, 'Smith' and 'bolt' are obtained directly from the input. The join is managed by reference to the translator's data about the application area. (This example is similar to one used by Date [Date77] page 100).

The above query is expressed in relational calculus. It means:

> Total the quantity field of the supplier/parts relation for which the following conditions hold:
> (a) The supplier number is the same as "Smith's" number and
> (b) The part number is the same as "bolt's" part number.

The parsing section will initiate a clarification dialogue if it cannot parse a query. If we reconsider the example:

"The liberals has been ledd by who in 74?"

then the user would participate in the following dialogue:

```
Eta: By 'ledd' do you mean...
     <cr> led
      1.  lead
      2.  leer
      3.  none of the above
User: <cr>
```

(Remark: Notice that the spelling corrector ranks its choices in decreasing order of likelihood. The choice, 'led' [a single deletion of a 'd'] can be selected by pressing carriage return. The other choices involve one and two substitutions respectively and are considered less likely to be correct.)

```
Eta: By '74' do you mean 1974?
     <cr> Yes
      1.  No
User: <cr>
```

(Remark: Again the choice that the system ranks as the most likely may be selected by pressing carriage return.)

```
Eta: Do you mean...
     <cr> "The liberals led who in 1974?"
      1.  "Who led the liberals in 1974?"
      2.  Something else?
User: 1
Eta:  Pierre Trudeau
```

(Remark: The syntax corrector chooses and ranks the alternatives given above as the most likely interpretations of the input. Notice that errors in person (ie 'has' instead of 'have') and errors in case ('who' instead of 'whom') were ignored. Such agreements, if they exist, may assist the parser but their presence is not required.)

One of the advantages of having a powerful syntactic corrector is that a smaller grammar can be used. An input will be accepted so long as it bears some structural resemblance to an equivalent form in the grammar. Since Eta is to be a microcomputer based system, this space saving feature is a considerable asset.

It is expected that even naive users will require very little formal instruction (beyond a definition of acceptable topics) in order to use Eta. The clarification dialogues serve as a reasonably painless training program.

The Error Handling Strategy

Figure 2 is a conceptual view of the Eta Parser.

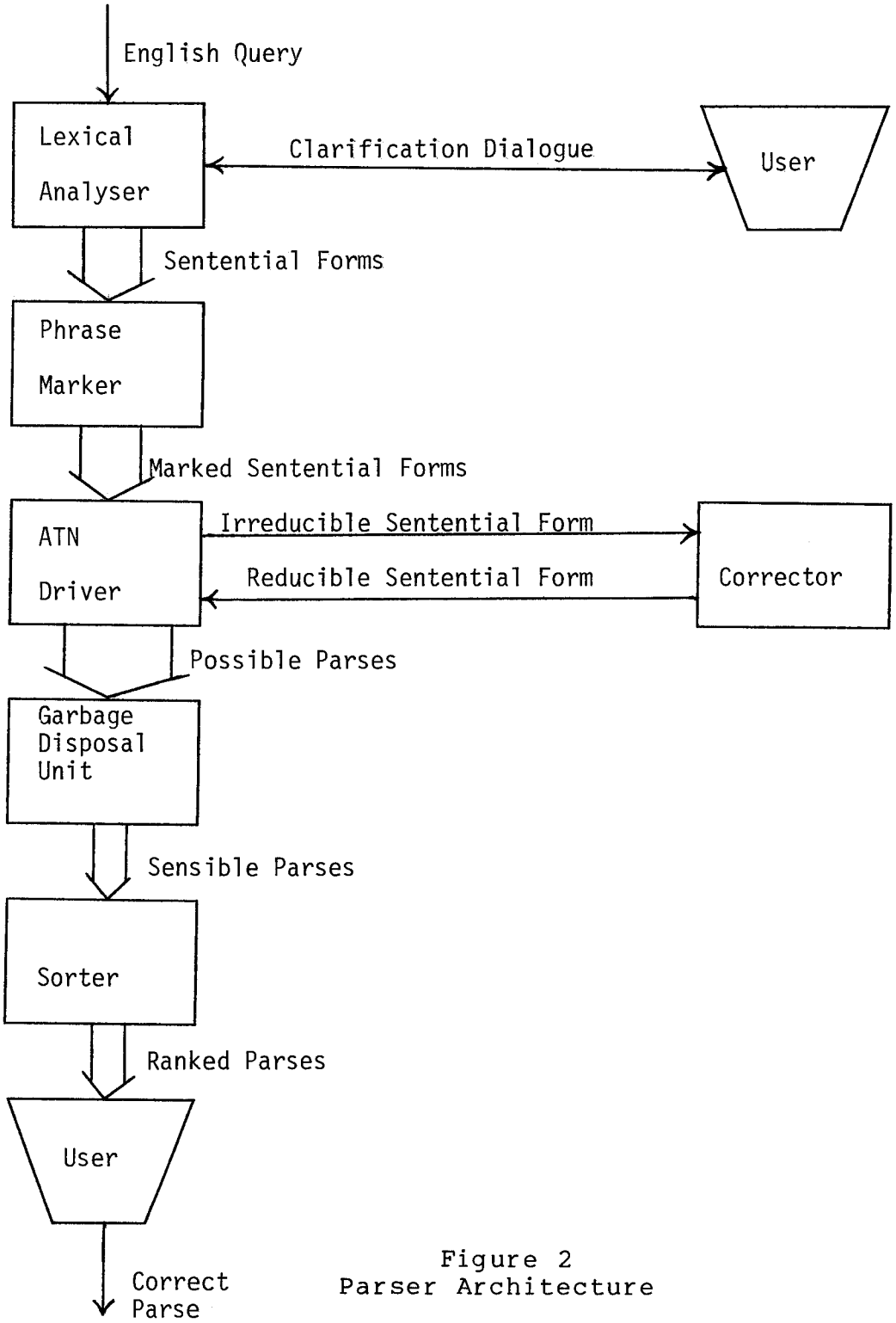

Figure 2
Parser Architecture

The Eta Interface

The lexical analyser performs a number of functions. Primarily, individual words are identified with their directory entries. If a word is not in the dictionary then the spelling corrector is invoked. The spelling corrector is based on the work of Szanser [Szanser68, Szanser70, Muth77]. In addition certain idioms are replaced with standard system labels. For example the interrogative interjection, "Could you please tell me..." would be replaced by the symbol, 'II'. When words with multiple syntactic roles are encountered, (for example 'like' can be an adjective, a verb or a preposition) each distinct dictionary entry is noted and passed along.

The phrase recognizer marks certain key types that can be used as starting points for the ATN driver. Specifically nouns and articles identify noun phrases, prepositions introduce prepositional phrases and so on.

The grammar is specified by an augmented transition network (ATN) [Woods70]. The ATN driver analyses the syntax of the input while traversing the arcs of the ATN. In Eta the driver has been enhanced in three ways. First, the conditions on the arcs (which specify agreement in number, person, case and the like) are treated as optional [Kwasny79]. Second, parsing may begin at any of the starting points supplied by the phrase marker rather than being restricted to the beginning of the input. Third, parsing may move in either direction from the starting point rather than in

strict left to right order. For example, if we start looking for a noun phrase by examining a noun, we will want to back up and look for preceding adjectives. If the ATN driver cannot complete the analysis, it invokes the corrector.

The corrector essentially shuffles the words and nonterminals of the partially analysed input in order to produce a modified input that will fit one of the paths of the ATN. It may also delete symbols or insert dummy items. This can be a complex computation (there are over 3.5 million permutations of a 10 word sentence). Eta uses two heuristics to limit this problem. First, the ATN driver uses its multiple starting points to parse as far as it can before invoking the corrector. Thus entire sequences of words may be reduced to single symbols. For example the noun phrase, "the big red car" would be reduced to the symbol, 'NP'. If by this process we reduce our 10 word sentence to a 5 symbol sequence then we have reduced the number of permutations from over 3.5 million to just 520. Second, the corrector uses the ATN as a guide for rearrangement. Thus if a NP (noun phrase) is out of place, the corrector tries to position it on a path of the ATN. Notice that entire syntactic units, as well as individual words may be shuffled in this manner[*]. The corrector keeps track of the number

---

[*] Although it is beyond the scope of this paper, it should be mentioned that we must occasionally back up the parse. For example when identifying "The liberals has been ledd by who in 74?" with "Who led the liberals in 1974?" we must break up the prepositional phrase, "by who" (which the ATN driver would have previously reduced to 'PP').

of changes it makes and uses the count to assign a score to each prospective interpretation [Wagner74, Selkow77].

We will define the kernel of Eta's grammar to be those sentential forms described by the system's augmented transition network. It is the goal of the corrector to find the member of the kernel that most closely resembles the partially parsed input. A variety of similarity measures could be used. Currently the measure is based on the number of editing operations required to change the input into an element of the kernel.

The purpose of the garbage disposal unit is to reject nonsensical corrections. It will be discussed further in the next section.

Finally the sorter uses the scores produced by the corrector to rank the possible parses in decreasing order of likelihood. These choices are presented to the user who selects the one that reflects his intent. The possibility that none of the alternatives are acceptable is discussed in the summary.

Let us trace the parsing of our sample query about liberal leadership in detail. First the lexical analyser matches words with their dictionary entries. We denote this as follows:

```
  The    liberals       has         been ledd   by
   |        |             |           |    |     |
article proper-noun auxiliary-verb verb  ? preposition

        who                in      74
         |                 |       |
interrogative-pronoun preposition ?
```

The spelling corrector and the user agree that "ledd" should be "led". The lexical analyser looks up "led" and notes that it is a verb. The digit string, 74 appears after a preposition and must therefore act as a noun. There are two lexical entries for a digit string. As an adjective, a digit string is interpreted as a number (eg. 74 bolts). As a noun, a digit string may be interpreted as a date (provided it is of a suitable format). Here "74" matches the format, "19-digit-digit" and the date, 1974 is proposed. The user verifies this and 1974 is tagged as a noun.

The phrase marker marks "the", "liberals", "who" and "1974" as starting points for noun phrases. It also marks "by" and "in" as starting points for prepositional phrases.

The ATN driver performs the following reductions:

"The liberals" to 'NP' (noun phrase)

"has been led" to 'CV' (complete verb)

"by who" to 'PP' (prepositional phrase)

"in 1974" to 'PP' (prepositional phrase)

The input is thereby reduced to: "NP CV PP PP". Our goal ("Who led the liberals in 1974?") is of the form: "NP CV NP PP" [*].

The input can be converted to this form in a number of ways. The word "by" can be deleted, turning the first 'PP' into a 'NP'. Alternatively the word "by" can be deleted and the resulting 'NP' exchanged with the first 'NP' (ie "The liberals"). A partial list of corrections and the number of editing operations required to produce them appears below.

Reference #	# of Changes	Correction
i	1	The liberals led who in 1974?
ii	2	Who led the liberals in 1974?
iii	2	The liberals led 1974 by whom?
iv	3	1974 led the liberals by whom?

These alternatives are sent to the garbage disposal unit which applies some elementary semantic constraints. It notes that a date is incapable of leading or being led and thus discards alternatives iii and iv.

Finally the sorter ranks the surviving alternatives and presents them to the user for verification.

---

[*] The corrector may actually match the partially parsed input with several sentential forms in the kernel. However, for the sake of simplicity we will consider only the indicated form.

The Garbage Disposal Unit

In the previous section the third and fourth interpretations of the input were rejected as meaningless. Such semantic garbage is apt to arise from error correction based solely on syntax [*]. Consider the following input:

"To drink did John like milk?"

which was intended to be: "Did John like to drink milk?". The following are some of the meaningless corrections that might arise:

a) Did milk like to drink John?
(Syntactically equivalent to: Did John like to drink milk?)

b) Did John like to milk drink?
(Syntactically equivalent to: Did John like to drink milk?)

c) Did milk drink to John?
(Syntactically equivalent to: Did John drink to happiness?)

d) Did the drink milk John?
(Syntactically equivalent to: Did John milk the cow?)

The garbage disposal unit is included in Eta as an enhancement. The user could of course, reject nonsensical corrections during the clarification dialogues. However satisfaction will be considerably improved if the user is not subjected to an avalanche of meaningless corrections every time he makes a mistake.

Some fairly simple semantic rules can dispose of quite a bit of garbage. For example the constraint:

Verbs of type: "ingest" take nouns of type: "animate" as their subject.

---

[*] Although it is beyond the scope of this paper, an ATN may provide more than a purely syntactic description of a language. If some of the language semantics are embodied in the grammar then fewer nonsensical corrections arise.

disposes of alternatives a) and c) above. The constraint:

> Verbs of type: "milk" take nouns of type: "cow" as their object.

disposes of alternatives b) and d) above.

The garbage disposal unit is still being designed. It is anticipated that the form of the constraints will be based on the conceptual graphs of Sowa [Sowa76]. The design will trade off thoroughness with brevity. An occasional silly interpretation may amuse the user, but avalanches of nonsense must be prevented. The garbage disposal unit will be constructed to reject most, if not all of the nonsensical corrections.

Summary

An error correcting parser should not be too powerful. When presented with inputs like:

> "Rubber baby buggy bumpers."
> "ABC 123 #?!"

Eta will try to abort the parse. The interface informs the user that all of its alternatives are unlikely matches and asks whether it ought to proceed with the parse. The determined user may forge ahead since the spelling and grammar correctors can eventually present him with all possible valid queries.

A special problem concerns queries that are outside the permissible domain of discourse [Shneiderman78]. To deal with this, certain key words that identify common misconcep-

tions about the domain of discourse may be included in the lexicon. The presence of these words will cause the ATN driver to display an appropriate message and ask the user if parsing should continue.

With Eta, the data base user may always insist that the error corrector match his query with some valid form. Eta will however, inform the user whenever the system's heuristics indicate that a successful match is unlikely.

In the near future a number of very large data bases may become accessible to large groups of untrained users. This is particularly true for the various Videotex, two-way television systems such as Telidon. In this kind of environment it is important that a natural language interface be robust and easy to use. Eta is an experimental interface designed specifically for this new application area. Experience gained through such systems as Eta may lead to greatly increased accessibility to very large data bases for the naive user.

References

[Brown79] H. G. Brown, "Telidon Videotex System," IEEE Transactions on Consumer Electronics, Vol 25 #3, July 1979, pp 256-268.

[Codd78] E. F. Codd, R. S. Arnold, J-M Cadiou, C. L. Chang, N. Roussopoulos, "RENDEZVOUS version 1: An Experimental English-Language Query Formulation System for Casual Users of Relational Data Bases," IBM Research Report #RJ2144(29407), January 1978, IBM Research Laboratory, San Jose, California.

[Date77] C. J. Date, An Introduction to Database Systems, second edition, Addison-Wesley, 1977, p 79 and p 100.

[Grishman73] R. Grishman, N. Sager, C. Raze, B. Bookchin, "The Linguistic String Parser," AFIPS Conference Proceedings, Vol 42, June 1973, New York, pp 427-434.

[Grishman78] R. Grishman, L. Hirschman, "Question Answering from Natural Language Medical Data Bases," Artificial Intelligence, Vol 12 #1,2, August 1978, pp 25-43.

[Hendrix78] G. G. Hendrix, E. D. Sacerdoti, D. Sagalowicz, J. Slocum, "Developing a Natural Language Interface to Complex Data," ACM Transactions on Database Systems, Vol 3 #2, June 1978, pp 105-147.

[Kameny78] I. Kameny, J. Weiner, M. Crilley, J. Burger, R. Gates, D. Brill, "EUFID: The End User Friendly Interface to Data Management Systems," Proceedings of the 4th International Conference on Very Large Data Bases, West-Berlin, September 1978, pp 380-391.

[Kwasny79] S. C. Kwasny, N. K. Sondheimer, "Ungrammaticality and Extra-Grammaticality in Natural Language Understanding Systems," Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics, San Diego, August 1979, pp 19-23.

[Muth77] F. E. Muth Jr., A. K. Tharp, "Correcting Human Error in Alphanumeric Terminal Input," Information Processing and Management, Vol 13 #6, 1977, pp 329-337.

[Mylopoulos76] J. Mylopoulos, A. Borgida, P. Cohen, N. Roussopoulos, J. Tsotsos, H. Wong, "TORUS: A Step Towards Bridging the Gap Between Data Bases and the Casual User," (An extended and updated version of the paper that was presented in The Proceedings of the 4th International Joint Conference on Artificial Intelligence, Tbilisi USSR) Department of Computer Science, University of Toronto, 1976.

[Plath76] W. J. Plath, "REQUEST: A Natural Language Question-Answering System," IBM Journal of Research and Development, Vol 20 #4, July 1976, pp 326-335.

[Rohrich78] J. Rohrich "Automatic Construction of Error Correcting Parsers," Institut fur Informatik II, Universitat Karlsruhe, Bericht Nr 8, September 1978 pp 2-3.

[Selkow77] S. M. Selkow, "The Tree-To-Tree Editing Problem," Information Processing Letters, Vol 6 #6, December 1977, pp 184-186.

[Shneiderman78] B. Shneiderman, "Improving the Human Factors Aspect of Database Interactions," ACM Transactions on Database Systems, Vol 3 #4, December 1978, pp 417-439.

[Sowa76] J. F.Sowa "Conceptual Graphs for a Data Base Interface," IBM Journal of Research and Development, Vol 20 #4, July 1976, pp 336-357.

[Szanser68] A. J. Szanser, "Error-Correcting Methods in Natural Language Processing," IFIP Information Processing 68, Vol 2, 1968, pp 1412-1416.

[Szanser70] A. J. Szanser, "Automatic Error Correction in Natural Languages," Information Storage and Retrieval, Vol 5, 1970, pp 169-174.

[Wagner74] R. A. Wagner, M. J. Fischer, "The String to String Correction Problem," Journal of the ACM, Vol 21 #1, January 1974, pp 168-173.

[Waltz78] D. L. Waltz, "An English Language Question Answering System for a Large Relational Database," Communications of the ACM, Vol 21 #7, July 1978, pp 526-539.

[Woods70] W. A. Woods, "Transition Network Grammars for Natural Language Analysis," Communications of the ACM, Vol 13 #10, October 1970, pp 591-606.