Portable Natural Language Front Ends - A Review

Flávia A. Barros & Anne DeRoeck
University of Essex
Department of Computer Science
Colchester - CO4 3SQ - U.K.

August 1993

Abstract

This report reviews Natural Language Front-Ends developed with the goals of portability and robustness.

Some of the most prominent system in the area are presented here in the light of well-defined criteria which lead to a clear judgement of the adequacy of such systems as portable NL Front-ends.

A comparison among the system is provided, also guided by the established criteria for the analysis of these Front-ends.

1 Introduction

Attempts for providing Natural Language systems to access data date from the late 60's with systems like STUDENT[Bobrow 1967] and REL[Thompson et al. 1969]. Such systems represented the landmark of a new conception of data access either in isolated files or in data bases. The next decade started with systems like LUNAR[Woods et al. 1973] and SHRDLU[Winograd 1972], which, like the earlier systems, were still developed for a specific application in a particular domain.

The preoccupation of providing the user with portable interfaces came only with PLANES[Waltz 1975] and LADDER[Hendrix et al. 1978], Front-ends for Database interrogation. Although both systems were also dedicated to one domain of application, the authors showed interest in providing for some portability in what concerns the domain of application. However, such adaptation would be a demanding and slow process.

The 80's shed new light on the development of such Front-ends, with the emergence of systems like TEAM[Martin et al. 1983, Grosz et al. 1987], LDC[Ballard & Tinkham 1984], Ginsparg's system [Ginsparg 1983], among others. Such systems present a much higher degree of portability than the previous ones. However, they lost some capability for treating Natural Language expressions and phenomena like elliptical queries or anaphoric references due to the lack of domain-dependent information available before the query reaches the Database.

Therefore, it is easy to identify tension in Front-ends between the degree of portability and coverage of contextual NL phenomena they offer (portability *versus* NL facilities).

The ideal Front-end should maintain a high degree of portability (associated domain, application, DBMS, Database - single or a DB network) without loosing desirable linguistic facilities (treatment for ellipsis and anaphora, enabling for dialog-like consultations).

The motivation for such a challenge comes from the original goal of developing tools for helping casual, non-expert, users in dealing with their varied Database applications. As a consequence, commercial viability is an essential factor in portable Front-ends, which only make sense in a commercial context.

This report presents a survey of the area, showing some of the most prominent systems, in the light of some desirable features they should convey. Because the majority of the portable Front-ends do not offer flexibility in what concerns the language of application, being all limited to work with the English language, we decided to include here a Front-end developed for the German language, the Datenbank-DIALOG Interface[Trost et al. 1988, Heinz et al. 1992], a portable Front-end for relational Databases.

The next section specifies criteria for analysing the chosen Front-ends, which are presented in section 3 and 4. Finally, a summary offers a comparison among these systems based on the chosen principles.

2 Establishing Criteria

In order to analyse the systems being presented here, it would be helpful to establish criteria covering the basic points we should look at in order to be able to judge for some essential characteristics expected from a portable NLFE: domain-independence, transparency of system behaviour and NL coverage. We try to estimate the ease of portability which affects the system's commercial viability.

For each systems, we will try to comment on:

(a) General description and main goals
(b) Acquisition session
(c) Natural Language coverage
(d) Degree of portability

Most of the systems being presented here bear a similar design in what concerns the Natural Language processor, *ie* they include a general grammar, providing, after a parsing phase, some level of semantic representation. However, they differ in the way each of these features has been realised.

Acquisition sessions aim to obtain the domain-dependent information necessary for the system to work. Each system has a different way of dealing with this process, which can be manual,

semi-automatic, or fully automatic (controlled by the system). The knowledge acquired also varies according to the system's demands. Most of them have a manual acquisition session, and acquire world knowledge in order to build their internal knowledge base. Another point to be looked at is the kind of user in charge of that session (DB expert, system designer, application expert, end user). This choice is closely related to the system's design and to the type of information required.

The coverage of contextually sensitive NL phenomena is also an important point to be examined, since this is directly related to effectiveness of the interface between the user and the Database system. We emphasize the presence of mechanisms for anaphora resolution since it would provide for the linkage between queries, allowing for continuity during the query session (dialog-like consultations).

Finally we reach our initial aim, a judgement on the system's degree of portability. We believe that there exists a trade off between portability and NL coverage, specially in what concerns anaphora resolution. Such facility requires specific domain information, which is not available to the NL processors in systems with a high degree of portability. A summary will be presented in the end of this report, in order to facilitate the comparison among the systems.

3 Early Systems

As we mentioned above, systems developed in the 60's and early 70's present a very satisfactory NL coverage, although lacking facilities allowing for portability. As examples of those systems, we chose PLANES and LADDER for they already present some degree of portability, however far from the ideal standards.

PLANES

PLANES is a Natural Language Interface for a data base containing U.S. aircraft maintenance and flight information over a period of time. There is no parsing phase. Instead, requests are matched against a network of pre-stored request patterns which is similar to an Augmented Transition Network (ATN)[Woods 1970], with nodes corresponding either to specific words or to phrase types (*eg* *time-period, matches phrases like 'during April 1974', 'in 1973', etc). Request meanings are stored similar to the approach taken by LUNAR, which can be seen as a precursor of PLANES.

When all words in a request have been processed by the network, the systems checks whether the request can be associated with a unique pre-stored request form. If more than one association is possible, the *context registers* are used to decide which form is appropriated for the request.

These context registers consist of program global areas with information about the request and the DB answer. There is a context register for each phrase type in the network, as well as for the DB answer. These registers help instantiating pre-stored request forms with missing

information, and also to handle pronouns and referential phrases (*eg* 'the plane'). The system proceeds by binding the pronoun/referential phrase to the current context register value for this phrase type, after the pronoun/referential phrase type has been identified in the pre-stored request network. The context registers have only information about the last processed request; new information overwrites the previous state.

If no exact matching is possible, the system tries to find a partial or fuzzy matching. This is done by dropping one or more words from the original request in order to force a match.

As such, the system manages to provide some desirable dialog facilities, like dealing with incomplete requests, and some level of pronoun reference resolution. However, portability is made difficult by the choice of working within the paradigm of semantic grammars. Most of the words are implicitly defined within the network nodes, instead of being entries to a system dictionary. This decision seriously affects the portability of the system, requiring a large amount of work to adapt the interface to a different application.

Additional difficulties arise when trying to enumerate all possible request patterns. A more serious drawback affects the system reliability from a user's point of view. The design will tend to collapse a class of similar, but nevertheless distinct, queries into a single pre-stored pattern. Fine distinctions in questions can thus be lost in a process which remains opaque to the user. This behaviour weakness the user position when interpreting the answer received. Clearly, the gravity of the consequences depend initially on the coverage of the Front-end, but the flaw is endemic in systems designed along this lines.

LADDER

LADDER (for Language Access to Distributed Data with Error Recovery) was developed as a management aid to Navy decision makers, providing Natural Language access to a distributed Database over a computer network. Unlike PLANES, LADDER is modular, consisting of three independent components: INLAND (Informal Natural Language Access to Navy Data), IDA (Intelligent Data Access) and FAM (File Access Manager).

INLAND accepts a wide range of Natural Language questions in the particular domain of application, producing an intermediate level of representation for the query which is independent of the DB structure. IDA takes this query representation as input and, employing a Data Model, breaks it into a sequence of queries to the specific files in the language of the DBMS in use. FAM, the last component, uses a locally stored Data Model to provide information about the physical location of each file within the distributed Database, and is also responsible for the access to such files.

INLAND was developed within the *LIFER* framework (Language Interface Facility with Ellipsis and Recursion)[Hendrix 77], a package for Natural Language Interfaces construction which consists of two basic parts: a set of interactive functions for defining application languages, and a parser. An application language consists of a subset of a natural language (English, in LADDER) that is appropriate for the interaction with DBMS software. The definition of such a (sub-)language includes the specification of a lexicon and a semantic grammar for the

specific domain of the application. The parser consists of 'transition trees' (a simplification of Augmented Transition Networks [Woods 70]).

The LIFER parser provides special features for dealing with spelling correction, incomplete(elliptical) input, and simple cases of anaphora. After the parsing phase, each word/phrase of the input query is associated with a symbol or metasymbol of the application language. Anaphora resolution is based on simple symbol/metasymbol matching between the representations of the current query and the previous one, which constitutes the current context. The matching is realised by a binding function provided by LIFER. INLAND can also use the idea of global context (registers) to hold the context of previous queries, one register for each type of symbol/metasymbol. However, unless the same type of symbol/metasymbol is used in the most recent query, no binding is possible. This is a limitation of the LIFER binding procedure.

INLAND interprets definite noun phrases like 'the american ships' as referring to all american ships in the Database, unless there is a group of american ships in the context established by the previous query.

LIFER also provides the user with facilities for extending the application language by defining synonyms and paraphrases for words and phrases already in the language.

Both systems PLANES and LADDER offer the user some very desirable language facilities, *eg* treatment for incomplete inputs and some anaphora resolution. Their weak point actually resides in the portability, since the constraints imposed by the adoption of semantic grammars as queries' internal representation make it very difficult to rapidly adapt the system to a different application. However, LADDER, for being modular, is clearly more portable than PLANES.

4 Towards a Higher Degree of Portability

The evolution of the early NL systems led to the development of Front-ends presenting higher degree of portability in the 80's. However, on the other hand, those systems seemed not to be able to reproduce some of linguistic facilities provided by the previous ones, such as anaphora and ellipsis resolution, spelling correction, continuous consultation, etc. This seems to be due to the impossibility of accessing information about the domain of application before the Database is reached by the query.

To support our claims, we present here four portable Front-ends, LDC, Ginsparg's system, TEAM and Datenbank-Dialog, as clear examples of a satisfactory degree of portability, implying in the loss of NL facilities.

LDC

LDC (for Layer Domain Class) is a Natural Language processor designed for office domain applications with two primary goals: allow queries by untrained users to access informally

structured text-edited data files; and enable a super-user (experienced with the system, but not necessarily one of the system designers) to adapt the system to new domains of application whilst maintaining the interface reliability. The initial interest of the system's designers was studying the specification of complex semantics by users.

The system consists of three modules: the Preprocessor, the English-language processor and the Retrieval module. The Preprocessor is responsible for the knowledge acquisition session, during which the super-user provides specific information about the new application (vocabulary and domain structure) by entering the words together with their major category (nouns, verbs or modifiers) and their morphological and semantic properties; the names of each type of entity of the domain, and the nature of relationships among them. The relationship among domain entities is organised as a type hierarchy. Information about the composition of the physical data files (text-edited) is also acquired during this session, in order to allow for LDC to build up its internal Database system. The provided information is stored in separate files to be used by the two subsequent modules. The super-user determines the termination of the session.

The English-language processor parses the input query by using files created during the initial session together with a grammar specially designed to allow for portability. This combination is claimed to provide the capabilities of semantic grammars, whilst maintaining domain independence. The grammar is a collection of phrase-structure rules with context-sensitive augmentations, bearing resemblance to both phrase structure grammars and ATNs.

Words with more than one entry in the dictionary will have all their possible meanings sent to the parser, which selects the appropriate one by means of (a) augmentations of the applied rule (specifying context-sensitive constraints); (b) word features found in the dictionary entry (providing for simple context dependencies); (c) information on acceptable attachments for units as prepositional phrases or relative clauses, provided by the *compatibility* file created during the acquisition session.

The three items above can be seen as a "context" for the word/phrase being parsed, used for disambiguation and also for anaphora and ellipsis resolution. However, the accepted cases of anaphora and ellipsis are very limited. The system handles only few cases of intra-sentential anaphora, dealing just with referential expressions (no treatment for pronouns in general is provided).

The parsed input is translated into a formal query in the system's retrieval language, incorporating specific domain structure information from the files created during the Preprocessor phase. LDC has its own retrieval processor, which acts as a knowledge base, processing semantic information about the query.

In helping the parser to perform disambiguations, the augmentations provide a valuable facility in a portable interface, since it is not possible to predict the syntactic, semantic and other forms of ambiguity arising in each new application.

LDC can be adapted to a new Database covering the same domain quite quickly, the modification of the data files is easy and fast. However the degree of complexity of the information needed to change domains is very high, since the super-user has to define a world model

(domain structure) for each new application. The conception of such models can be time consuming and misleading, since the super-user is not necessarily an expert in the application's area.

Ginsparg's System

Ginsparg's System is a general Natural Language Interface for relational Database interrogation, having been designed to show robustness and portability. The system consists of two modules: a general purpose Natural Language Processor and a Database Application Program.

The NL Processor builds a semantic representation of the input query using a formalism based on semantic networks [Quillian 1968]. The network is a knowledge base containing the system's concepts (vocabulary), which have their meaning implicitly determined by their relation to other existing concepts, and the way they are manipulated by the system.

The Database Application Program takes the NL Processor output, building up a query in an augmented relational algebra, which is straightforwardly translated into the language of the DBMS being used on the application. This module also deals with a range of linguistic constructions appearing in the query (*eg* coordination, negation, ellipsis, etc).

There is no automatic acquisition session; the system designer is responsible for setting up the interface to the new application. The necessary information consists of defining concepts not present in the network, and the definition of the *Data Base Connection*, a Data Model with information on how to link such concepts to the Database.

The network should not be changed, but only augmented to cope with different DB schemes, as different representations for the same word might be required. The concepts are extended case frames [Fillmore 1968], having case preferences associated with each concept case. The system designer provides the concepts and cases for the system to derive the relations among the concepts based on the case preferences.

The information provided by the DB Connection file is mainly used by the DB Application Program for the query building process. This module handles **negation**, by introducing a set difference when necessary; **conjunction**, by introducing either an intersection (*and*) or a union (*or*); **quantifiers**, via a post processing phase, which adds a set of commands to check sets difference created before; among others.

The system is able to deal with false assumptions made by the user (queries that do not make sense in relation to the current Database), generating cooperative responses. It also provides a broad coverage of cases of ellipsis, intra and inter-sentential anaphora, allowing for the reference to be resolved in relation to the previous query or to the DB answer. No details are provided on how this mechanism works.

The assignment of semantic meaning to some modifiers (*eg* **red** pencil) might be misleading, since there is not a precise way to determine it. An example taken from [Ginsparg 1983 - pp 25] illustrates the matter: "red pencil" is either a pencil which is red or one that writes red,

depending on the path chosen. If no contextual information is available, the NL Processor chooses the shortest path.

As mentioned above, the system induces the network links on the basis of concept case preferences. However, since the network cannot be changed, the DB Connection provides additional preference information to overwrite the NL Processor's case preferences. The new case preferences, together with a comparison among path-lengths in the semantic network are used to induce coercion, such that the concept with the broken preference is coerced to another one via a path in the network.

Two problems can be anticipated with the use of this mechanism: there is no guarantee that the shortest path will correspond to the query the user intended (affecting system's transparency); the same information (case preferences) comes from two different sources (the network and the DB Connection), leading to a large amount of data to be manipulated by the system.

The system can be moved into a new linguistic domain and/or new Database application. However, difficulties are to be faced in adapting the semantic network, which carries a world model, since it is likely to change considerably from one application to another. The system designer has to provide all new concepts, their cases, and new case preferences (via the DB Connection) for the concepts already in the systems which need to be adjusted to the new application.

TEAM

TEAM (for Transportable English database Access Medium) is a portable Natural Language Interface to relational Databases designed to be adapted to new applications by users non-expert in NL processing, but rather DB Experts.

The system comprises three main components: *Acquisition*, responsible for obtaining linguistic and DB information from the DB Expert in order to expand and update the user's world model; *DIALOGIC*, which takes the user's input sentence, producing a logical form representation to be used by the third component, the *Schema Translator*, in order to produce a query in the appropriate Database language.

The acquisition dialogue is totally controlled by the system, which also decides when enough information has been acquired. The DB Expert provides Database related information, such as DB structure, the conceptual content of its files and fields, how they encode it, and words and phrases used to refer to these concepts.

Because the DB Expert is not expected to have knowledge of linguistics, the necessary information required by TEAM is inferred from the acquisition dialogue. The acquired information is incorporated into the lexicon, conceptual schema and DB schema. For each entry, the lexicon provides morphological, syntactic and semantic information for each of its senses. There is also a link between words and concepts which they can refer to in the conceptual schema.

The conceptual schema is a type hierarchy network containing the classification of all concepts

in the application. The network carries some core concepts, acquiring the domain-specific ones from the DB expert.

The NL processor, DIALOGIC, includes a parser, the lexicon, a grammar, a semantic translator, basic pragmatic functions and procedures for determining scope of quantifiers. All the domain-dependent information used by this module can be obtained from the conceptual schema and the lexicon.

The parser together with the lexicon and the augmented phrase structure grammar analyse all common English sentences, producing a set of syntactically acceptable parse trees. The grammar rules are annotated with constructors providing for context-sensitive constraints, which help choosing the syntactically "best" resulting trees, which will be taken by the semantic translation process.

Such mechanism compromises the system's transparency, since the choice is done on syntactic bases; the user's desired query meaning might be ruled out before reaching the Database.

Each syntactic rule has an associated semantic function, its *translator*. Nevertheless, the semantic translation is separate from the parsing process. Translators are domain-independent, although they derive much of their behaviour from the information associated with the lexical entries. The translators use basic semantic functions to construct the particular representation of the interpretation of the phrase.

The semantic representation of the query (syntactic tree annotated with semantic information) goes through two more levels of transformation: pragmatic functions, which resolve remaining ambiguity; and assigning of relative scope of quantifiers and operators (such as negation, superlatives, etc.). A final transformation phase is applied to the query by a post-processor, producing a logical form, which is an unambiguous representation of the English query.

And finally, the Schema Translator takes the logical form, generating a query in SODA query language [Moore 79].

Together with the lexicon, the system also acquires gender information, which helps resolving ambiguity in anaphora treatment. However, no details are provided either about the routine for anaphora and ellipsis resolution, or about the treatment of queries' context. Part of the context information is probably embedded in the grammar augmentations and in the conceptual schema (world knowledge).

The DB expert is in charge of choosing the appropriate classification in the conceptual schema for each new concept, and also informing the system whether these concepts represent sets that are disjoint. This requires precise and complete world knowledge.

Datenbank-DIALOG

Datenbank-DIALOG is one of the few portable German language interfaces for relational Database. It was designed to be easily handled by casual users, offering a good coverage

of a subset of German relevant for Database retrieval.

The system consists of four basic modules: The *scanner* gets the input sentence, building up a chart of lexical entries with case markers carring morphological and syntactic information; The chart is passed onto the *parser*, which performs syntactic analysis (based on an ATN grammar) together with a superficial semantic analysis (based on a caseframe matcher), providing one or, in case of ambiguity, more caseframes which represent the query at the domain level; the *interpreter* includes mapping the domain-level caseframes into a Database-dependent structure (DB-caseframes), which contains exclusively Database specific information (*eg* relations and column names), translating the DB-caseframes into the Predicate Logical Form, and finally, generating the query in SQL; the last module is the processing of the SQL query by the DBMS.

The scanner presents a phrasal lexicon, which treats pragmatic void standard phrases (*eg* "I would like to know"). The aim is speeding up the parsing process, since those phrases do not need to be analysed in depth, but only matched against patterns. There is also a spelling corrector controlled by the scanner, covering mistakes like omission, insertion, replacement and change of position.

There is no mention of an acquisition session, what information should be provided, and by what kind of user (*eg* system designer, DB expert, superuser, etc). They only cite a start-up procedure, where information about the current user is provided (*eg* name, status in relation to the domain of application - employee, manager -, etc). This information is recorded into the global context for the current consultation.

The caseframe of every successfully parsed sentence is pushed onto a Discourse Stack , which is used by the routines for ellipsis and anaphora resolution.

The system deals with elliptical sentences exhibiting incompleteness on the syntactic and semantic levels. The sentence is parsed, and the caseframes for the constituents are generated. Such caseframes are matched against the last caseframe in the Discourse Stack, based on syntactic agreement and semantic compatibility of the predicates. In case the match occurs, the constituent in the previous caseframe (top of the Discourse Stack) is replaced by the elliptical one, and the system continues the processing of the query in the ordinary way.

The system distinguishes between anaphoric pronouns and adverbs, and deictic pronouns ("I", "we", "my", etc) and adverbs (*eg* "today"). The values for the deictic referents are indicated in the global context, set by the start-up process. The procedure to deal with cases of inter and intra-sentential anaphora is based on Chomsky's (1981) Government-binding theory. Whenever an occurrence of intra-sentential anaphora is detected, the system proceeds by binding the caseframe of the pronoun to its antecedent (mere binding of variables). However, in inter-sentential anaphora cases, antecedents have to be chosen among candidates in the Discourse Stack. The procedure is similar to the one used for ellipsis resolution: once an antecedent is encountered, its caseframe will replace the pronoun's one, and the processing of the sentences goes on as usual.

The system also accounts for conjoined elliptic and anaphoric constructions, trying first to resolve the ellipsis, and then applying the anaphora resolution procedure to the result.

There is no mention on how to resolve ambiguity resultant from more than one possible matching during ellipsis or anaphora resolution. The constraints come only from the matching of caseframes plus the algorithm based on Chomsky's theory.

As in Ginsparg's system, we repeat here the same sort of criticism in what concerns the use of caseframes as part of the system's implementation. Caseframes carry domain information, which is likely to change considerably from one application to the other.

5 Summary

This section presents a table comparing the systems reviewed in this report in in the light of some criteria that might clarify the system's adequacy for transportation to a new domain, application and Database.

Information that was not found in the literature consulted is represented by the sign "!".

Systems X *Features*	PLANES	LADDER	LDC	Ginsparg's	TEAM	Datenbank -DIALOG
Acquisition Session	manual	LIFER semi-automatic	semi-automatic	manual	automatic	!
Provided By	!	!	super-user	system designer	DB expert	!
Grammar	semantic grammar	semantic grammar	augmented PSG	case grammar	augmented PSG	ATN grammar
Knowledge Base	ATN pattern matching	ATN based transition trees	type hierarchy network	semantic networks	type hierarchy network	caseframe matcher
NL Coverage	anaphora ellipsis	anaphora ellipsis spell corr.	anaphora ellipsis no pronoun	anaphora ellipsis	anaphora	anaphora ellipsis spell corr.
Provided Context	last query DB answer	last query	embedded in grammar rules	last query DB answer	!	discourse stack
Degree of Portability	low	low	medium	medium	high	high

Fig. 1 - Summary of Reviewed Systems *versus* Features

Finally, we would like to make some general comments on the highlighted systems' features and the consequences for portability.

All systems shown in this report present some anaphora and ellipsis resolution. However, as mentioned before, the level of refinement of those NL processors decreases as the degree of portability increases. This behaviour seems to be a consequence of the loss of domain-dependent information available for the NL processor. Such information is only found in the Database. The solution found was the construction of internal Knowledge Bases, which would acquire the necessary information during the adaptation of the system to the new application.

However, such Knowledge Bases require modeling of world knowledge, and world models are known to be difficult and time consuming to build. The person in charge of providing such information will necessarily have to be familiar with the domain of application, the application itself, the design of the knowledge base and the Database organisation.

Another feature shared by these systems is the demand for a complete lexicon, *ie* sentences containing words not in the lexicon (although in the Database) are not parsed. In general, this feature is undesirable for the following reasons: (a) Information will be duplicated (lexicon and Database), which constitutes a problem for applications where the amount of data is large (*eg* Industries - names of employees and products manufactured); (b) all Database updates will require lexicon updates also, creating an ongoing customisation requirement and leading to problems related to consistency of information.

On the other hand, complete lexica provide one of the most important sources of information for anaphora resolution, which are gender and number. In the light of such data, the problem of ambiguity among candidates decreases considerably.

As a conclusion of what was shown in this report, we would like the define the "ideal NL Front-end" as carring the following characteristics:

(a) domain-independence - which blocks the use of knowledge bases with domain-information;

(b) non-complete lexicon - unknown words should be looked at as placeholders to be bound once the query reaches the database;

(c) transparency - the user should be able to clearly understand how the system works out queries' meaning, in order to be able to produce them appropriately;

(d) a proper treatment for contextual NL phenomena - routines for anaphora and ellipses resolution should also be able to get the necessary information from the Database, using continuous contexts with information about all previous query (not only the most recent one), and which should also include the Database answer. This possibility would add continuity to the consultations.

6 References

[Ballard et al. 1984] Ballard, B.W., Lusth, J.C., Tinkham, N.L.; "Transportable English-language Processing for Office Environments"; in *Proc. of AFIPS National Computer Conference*; pp 643-649.

[Ballard & Tinkham 1984] Ballard, B.W., Tinkham, N.L.; "A Phrase-Structured Grammar Framework for Transportable Natural Language Processing"; in *Computational Linguistics*, vol 10(2), pp 81-96.

[Bobrow 1967] Bobrow, D.G.; "Natural Language Input for a Computer Problems Solving"; in (ed.) M. Minsky *Semantic Information Processing*; Cambridge, Mass; MIT Press; pp 133-215.

[Chomsky 1981] Chomsky, N.; *Lectures Notes on Government and Binding*; Dordrecht: Foris.

[Fillmore 1968] Fillmore, C.J.; "The Case for Case"; in (eds.) E. Bach, R. Harms, *Universals in Linguistic Theory*; New York: Holt, Rinehart and Wiston.

[Ginsparg 1983] Ginsparg, J.M.; "A Robust Portable Natural Language Data Base Interface"; in *Proceedings of the Conference on Applied Natural Language Processing*; Santa Monica, CA; pp 25-30.

[Grosz 1982] Grosz, B.J.; "Transportable Natural-Language Interfaces: Problems and Techniques"; in *Proceedings of the 20th Meeting of ACL*; Toronto, Ontario; pp 46-50.

[Grosz et al. 1987] Grosz, B.J., Appelt, D.A., Martin, P.A. and Pereira, F.C.N; "TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces"; in *Artificial Intelligence*, vol 32; Elsevier Science Publisher B.V., North-Holland; pp 173-243.

[Heinz et al. 1992] Heinz, W., Matiasek, J., Trost, H., Buchberger, E.; "Comparison in NLIs - Habitability and Database Reality"; in (ed.) B. Neumann *Proceedings of ECAI 92 - 10th European Conference on Artificial Intelligence*; Pub. John Wiley $ Sons; pp 548-552.

[Hendrix 1977] Hendrix, G.G.; "The LIFER Manual: A Guide to Building Practical Natural Language Interfaces"; Technical Note 138, Artificial Intelligence Center, SRI, Menlo Park, CA.

[Hendrix et al. 1978] Hendrix, G.G.; Sacerdoti, E.D.; Sagalowowicz, D.; Slocum, J.; "Developing a Natural Language Interface to Complex Data"; in ACM Transactions on Database Systems; vol 3(2)-June 78; pp 105-147.

[Martin et al. 1983] Martin, P.A., Appelt, D.A. and Pereira, F.C.N; "Transportability and Generality in a Natural-Language Interface System"; in *Proceedings of the 8th IJCAI; Karlsruhe, Germany*; pp 573-581.

[Moore 1979] Moore, R.C.; "Handling Complex Queries in a Distributed Database"; Technical Note 170, Artificial Intelligence Center, SRI International, Menlo Park, CA.

[Quillian 1968] Quillian, M.R.; "Semantic Memory"; in M. Minsky *Semantic Informat Processing*; The MIT Press.

[Thompson et al. 1969] Thompson, F.B.; Lockemann, P.C.; Dostert, B.H.; "REL: a Rapi Extensible Language System"; in *Proc. of the 24th National Conference of the ACM*; 339-345.

[Trost et al. 1988] Trost, H., Buchberger, E., Heinz, W., Matiasek, J.; "DATENBAN DIALOG: A German language Interface for Relational Database"; in *Applied Artificial telligence in Japan*, vol 1, pp 181-203; Hemisphere Publishing Corporation.

[Waltz 1975] Waltz, D.; "Natural Language Access to a Large Data Base: An Engineeri Approach"; in *Proc. of the 4th IJCAI*; Tbilisi, USSR; pp 868-872.

[Winograd 1972] Winograd, T.; *Understanding Natural Language*; Academic Press; New Yo

[Woods 1970] Woods, W.A.; "Transition Network Grammars for Natural Language Analysis in *Comm. ACM*; vol 13(10); pp 591-606.

[Woods 1973] Woods, W.A.; "Progress in Natural Language Understanding: An Applicatio to Lunar Geology"; in *Proc. of AFIPS National Computer Conference*; pp 441-450.