Unknown Value Lists and Their Use for Semantic Analysis in IDA - the Integrated Deductive Approach to Natural Language Interface Design

Werner Winiwarter

Institute of Applied Computer Science & Information Systems
Dept. of Information Engineering, University of Vienna
Liebiggasse 4/3, A-1010 Wien, Austria

ww@ifs.univie.ac.at

Abstract

The framework of our research originates from natural language processing and deductive database technology. Deductive databases possess superior functionality relevant to the efficient solution of many problems in practical applications, yet there still exists no broad acquaintance. As main obstacle we identified the absence of any user-friendly interface. Natural language interfaces have been proposed as optimal candidate, however, in spite of the vast number of ambitious attempts to build natural language front ends, the achieved results were rather disappointing. In our opinion the main reason for this is missing integration, responsible for insufficient performance and wrong interpretation. In our Integrated Deductive Approach (IDA) the interface constitutes an integral part of the database system itself which guarantees the consistent mapping from the user query to the appropriate semantic application model. This paper focuses on the semantic analysis for which we introduce unknown value list (UVL) analysis, a technique that operates directly on the evaluation of database values and deep forms of functional words, that is, syntactic analysis is only applied if necessary for disambiguation. We prove the feasibility of the IDA approach by use of a case study, the design and implementation of a production planning and control system.

Keywords Natural language interfaces, deductive databases, semantics, information extraction, logic programming.

1 Introduction

Deductive database technology emerged during the past decade, it combines the strengths of both logic programming and relational database algebra [30]. The extended functionality led the way to solutions to practical problems which could not be handled efficiently before. In spite of the superiority in comparison with relational database systems, there still exists no broad acquaintance and acceptance with regard to practical applications [18]. Since we identified the user interface as the main obstacle for a specific user to become familiar with a new database paradigm, our objective was to supplement deductive databases with a user-friendly front end.

Starting from the first days of research on natural language processing, the use of unrestricted language has been regarded as optimal choice to the communication of casual users with sophisticated database applications. Intensive work was done during the last decades and a huge number of prototypes were developed but somehow they suffered the same fate as deductive databases: they are still far away from widespread practical use [6]. The reason for this are the many limitations that still exist and which are caused mainly by the factor of missing integration. We deal with this problem by introducing a new type of architecture, the Integrated Deductive Approach (IDA) which brings together the two 'fellow sufferers' natural language interfaces and deductive databases in that the interface constitutes an integral part of the database system itself. This signifies that the complete natural language analysis is performed by the powerful logic language supplied by deductive databases which guarantees for the first time a homogenous mapping of the semantic representation of user input to the underlying database application [35].

This paper focuses on the component of semantic analysis. We propose unknown value list (UVL) analysis as new technique which makes optimal use of the information supplied by the semantic application model of the underlying database application and uses syntactic analysis only for disambiguation of several interpretations. The rest of the paper is structured as follows. After a discussion of related work we give a brief overview of the system architecture before dealing with the UVL-analysis in detail. Finally, we give an insight into implementation details and prove

Proceedings of the 7th Australasian Database Conference, Melbourne, Australia, January 29-January 30 1996.

the feasibility of our approach by applying it to a case study, that is, the design and implementation of a production planning and control system for the reference language German.

2 Related work

Most existent natural language database interfaces deal with the access to relational database systems: the early systems RENDEZVOUS [5] and PLANES [34] as well as more recent systems like TEAM [12] or System X [20]. Also for German language some promising prototypes have been developed, e.g. HAM-ANS [15] or Datenbank-DIALOG [29].

The crucial weak point from what all these systems suffer is the mapping from the final semantic representation of the input sentence to the actual database query which incorporates a discontinuity of homogeneity as concerns the different semantic models (e.g. mapping of relations or attributes, see [25] for a detailed discussion). The second great difficulty that database interfaces differently from other natural language applications must cope with is the processing of database values as part of the user query. As especially systems that claim to be domain-independent do not access the knowledge contained in the database for use in natural language analysis, the usual approach is to assume that undefined words represent database values (see [21]). Therefore, if one considers the possibility of misspelled values, those systems are not able to distinguish between new database values for insertion or update and misspelled existent data.

The very first database interface LUNAR [38] as well as the first commercially available natural language interface INTELLECT [13] tried to overcome this situation by retrieving the concerned values from the database. However, due to the huge search spaces and the limitations of relational database technology, this method severely affects the efficiency of the application. A different approach to the resolution of unknown values is to restrict the complexity of input resulting in some kind of pseudo-natural language, examples of such systems are LADDER [14], TQA [7], ENLI [16] or HAVANE [2]. Some implementations even delimit the use of natural language to a menu-based system, e.g. NL-MENU [28]. This decrease of complexity guarantees an efficient analysis but also leads to a significant reduction of habitability which questions the main reason for using natural language instead of formal query languages [23].

Although deductive databases incorporate the power to solve all those shortcomings, there is no existent work that makes full use of this power. The only known prototype of a natural language interface to a deductive database was designed by Gal/Minker who focused their research on the generation of natural language answers to user queries [11]. However, also this prototype uses only a loosely coupled interface resulting again in the above mentioned inaccuracies for the treatment of unknown words.

With regard to the interaction of semantic and syntactic analysis it has turned out to be advantageous not to follow the strictly sequential process model but to overlap the two steps of analysis. This is justified by the reduction of problem space for the parser by eliminating meaningless or contradictory interpretations already at an early point of processing [3]. We chose the semantically driven analysis [19] as most favourable solution for the development of database systems with well-defined semantic application models within the IDA architecture. This decision is strongly motivated by laying more stress upon the information extraction paradigm rather than upon text understanding [1].

Whereas the information extraction approach is well established for information filtering systems [17], much work on natural language interfaces still contributes to the text understanding paradigm. Therefore, such prototypes suffer from serious overhead of analysis (for a critique of such systems see [26]). Only few work exists that derives benefit from the underlying database model for semantic analysis, e.g. [33, 25]. The main reason for this can be seen in the fact that so far for interfaces to relational databases no adequate semantic model existed or caused by loosely-coupled architectures no access was possible. Only the complete integration of the linguistic analysis within the database architecture makes it possible to merge the two representation schemes in a natural and consistent way. Thus, the computation effort is minimised by providing at the same time a maximum of quality of analysis.

3 System architecture

The complex task of natural language analysis is subdivided into several components as shown in Figure 1. Our system architecture differs from the standard process model [8] in several aspects which will be explained in the following in detail.

In accordance with our intention of integrating the complete natural language analysis into the deductive database system, we adapted the lexical approach by storing only canonical forms in the dictionary and assigning to them all the morphological features, including also prefixes and compound words [37].

Figure 2 shows a simple example of the assignment of morphological features to a verb. In addition to information about the conjugation of the verb, a set of possible prefixes can be declared which constitutes derived verbs. Finally, a set of suffixes together with sets of required prefix sequences can be defined for deriving nouns or adjectives.

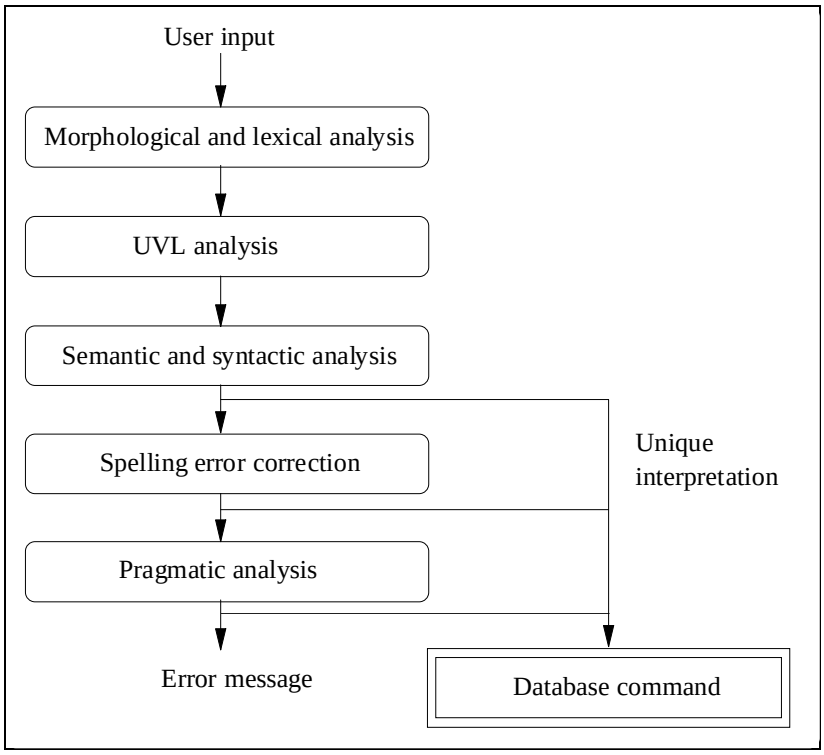

Figure 1: System architecture of IDA.

The dictionary entry shown in Figure 2 therefore covers all together 47 different surface forms (see Figure 3) including also irregular verb forms, compound verbs, and declensions of the derived nouns and of the adjectival use of both participles.

By supporting a hierarchically structured dictionary, we supplied the flexible insertion of syntactic and semantic features at the appropriate level in the hierarchy and employed inheritance mechanisms for the analysis process. All properties are inherited from the ancestors unless more specific properties defined at a lower level overwrite more general attributes. Therefore, an efficient and natural representation is obtained, also taking into account divergent specific meanings of derived words.

For syntactic analysis we selected Categorial Unification Grammar [31] as optimal basis because of two reasons. First, its requirement of assigning all grammar rules to the lexical entries which fits very well with our powerful hierarchical dictionary. Second, because its bottom-up parsing strategy is in conformity with deductive database semantics and satisfies perfectly our claim to analyse also incomplete and ungrammatical sentences in an easy and natural way. In order to deal with the free word order of German in a clear and concise way, we extended CUG by six new important concepts [36].

VERB(

mess,	stem of *to measure*
...	
11,	conjugation class
2,	past participle class
{ab},	prefix *ab* yielding *to survey*
{(er,{[durch]}),	suffix *er* in combination with prefix *durch* yielding the noun *diameter*
(ung,{[]})}	suffix *ung* yielding the noun *measurement*

).

Figure 2: Example of morphological features.

messen, messe, miß, mißt, meßt, maß, maßest, maßen, maßet, messend, messender, messendem, messenden, messende, messendes, gemessen, gemessener, gemessenem, gemessenen, gemessene, gemessenes, abmessen, messe ab, miß ab, mißt ab, meßt ab, maß ab, maßest ab, maßen ab, maßet ab, abmessend, abmessender, abmessendem, abmessenden, abmessende, abmessendes, abgemessen, abgemessener, abgemessenem, abgemessenen, abgemessene, abgemessenes, durchmesser, durchmessern, durchmessers, messung, messungen

Figure 3: Example of coverage of surface forms.

With regard to semantic analysis we applied the UVL-analysis method (see Section 4), that is, we did not produce complete grammatical structures of input sentences but based the semantic analysis directly on the deep form list produced by morphological analysis using syntactic knowledge only if necessary for disambiguation. This choice was made possible due to the well-defined semantic application model, therefore making the semantic analysis a rather straight-forward and natural task.

Since manipulation or retrieval of data is seldom performed by use of a single command but rather takes the form of a dialogue between user and computer, a great deal of research was done in pragmatic analysis aiming at extending the scope of analysis to the complete user session (for a good survey see [10]). We applied a simple but efficient technique which abstracts from specific manifestations at the surface level (ellipsis, anaphora) by using the entity and entity type of the preceding analysis to keep track of the actual focus.

Finally, we deal with spelling error correction, one of the most important features as concerns user acceptance by preventing the user of the tedious task of retyping erroneous input. In this context, IDA performs an optimal basis for the correction of misspelled database values because of the complete integration of the application data within the natural language interface. This makes it possible to verify efficiently the erroneous input word with the existent entries in order to retrieve a candidate for substitution.

SEM. FEATURES

```
SUBSTSUFFIX(
                ung,                    resulting in messung = measurement
                ...                             [measure, action]
                action
                ).

SUBSTSUFFIX(
                er,                     resulting in messer  = measurer
                ...                             [measure, subject]
                subject
                ).
VERBPREFIX(
                durch,                  resulting in durchmessen = measure in all directions
                ..                              [measure, across]
                across
                ).
 VERB(
                mess,                   resulting in durchmesser =  diameter
                measure,                        [diameter]
                ...                     overwrites the general semantic interpretation
                {(er,{([durch], diameter)}),    [measure, subject, across]
                    (ung, {([ ], ' ')})}
                ).
```

Figure 4: Example of semantic features.

4 UVL-analysis

As pre-requisite of semantic analysis we assigned semantic features to the dictionary entries at the appropriate level of abstraction by making use of inheritance. For similar approaches which also use hierarchically structured dictionaries for the efficient processing of semantic features see [9, 27, 32]. Figure 4 displays an example of the attachment of semantic features, also illustrating how divergent specific meanings of derived words can overwrite more general combined ones.

The morphological analysis computes for each word its deep form, so that the output of the morphological component takes the form of a deep form list (DFL) which gives for the individual input words a set of possible interpretations, each entry indicating the word stem, the word category, and the semantic deep form, e.g.:

Die neue Mindestbestellmenge von St 50 H ist 25 Stück (=The new minimal order quantity of St 50 H is 25 pieces)

DFL:

```
[{(die, artikel, [die]), (die, relativpronomen, [die])},
 {(neu, adjektiv, [neu])},
 {(menge, substantiv, [menge, bestell, mindest])},
 {(von, praeposition, [von])},
 {('St', unknown, string)},
 {('50', unknown, integer)},
 {('H', unknown, string)},
 {(sein, verb, [sein])},
 {('25', unknown, integer)},
 {(stueck, substantiv, [stueck])}]
```

An important difference of natural language interfaces in comparison to other fields of application for natural language processing techniques is the fact that unknown values possess a particular significance for the meaning of the sentence. Also in this context, only the IDA architecture makes it possible to distinguish between existent database values and new database values for insertion or update. If one considers also the misspelling of database values, the situation becomes even more complex. Furthermore, existent database values can serve as identifiers to entities and entity types within the database application. Again, valuable information can be obtained which reduces the number of possible interpretations and increases the efficiency of the natural language analysis.

Therefore, we propose as preliminary step for semantic analysis unknown value list (UVL) analysis. Its task is to transform the DFL produced by the morphological component to the following list presentations (see Figure 5 for the transformation of the previous example sentence):

- *unknown structure list (USL):* contains all unknown values as sub-lists, that is, compound values are split up to several list entries
- *unknown value list (UVL):* compound values are joined together, strings which represent numbers are converted
- *unknown type list (UTL):* compound and string values are looked up in the dictionary, if they represent identifiers of existent entities, the corresponding entity type is indicated, otherwise the value 'unknown' is inserted

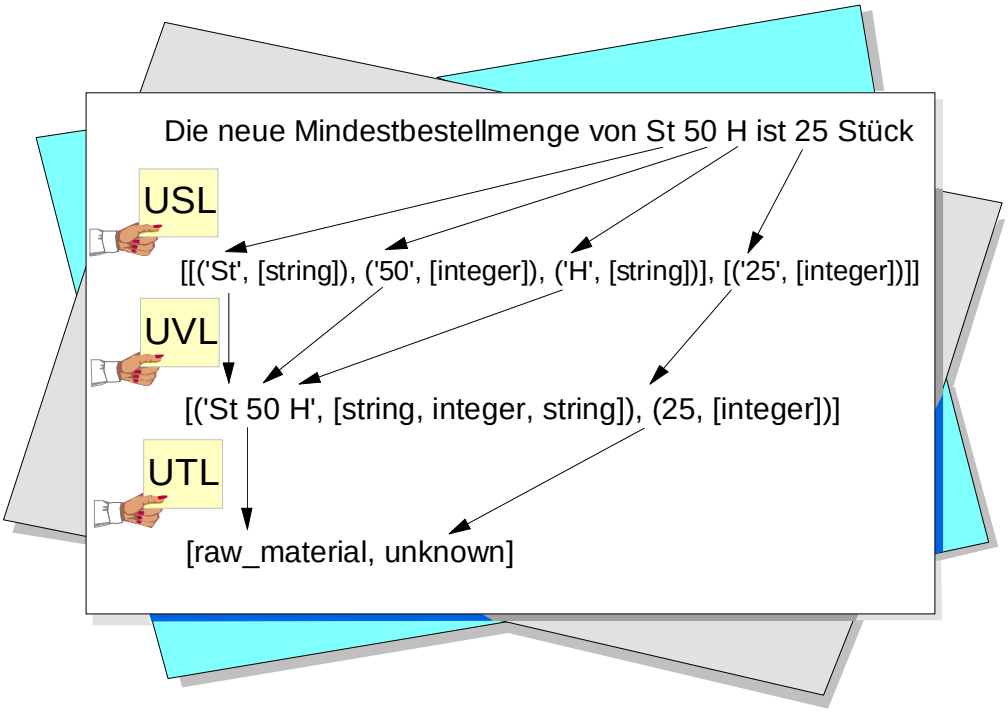

Figure 5: Example of UVL-analysis.

```
wort(Wort, (Eintr, ableitSubst, SemG)) <-      classifies word as derived substantive
                                               resulting in: stem, word category, deep form
    verb(Eintr,Sem,_, ... ,_, Suffixe),        retrieval of verbs
    affixe(Wort, Eintr, Pr, Suffix),           sub-string test of verb stems
                                               if satisfied, it returns the separated affixes
    suffixtest(Suffix, Suffixe,                checks suffix with dictionary yielding
       Praefixe, SemSuf),                      prefixes and general semantic feature
    praefixtest(Pr, Praefixe, SemPrSeq,        checks prefix with valid prefix sequences
       SpezSem),                               yielding specific or general semantic feature
    if(SpezSem ~= ' '                          if specific semantic feature exists,
    then SemG = [SpezSem]                      then it is assigned to deep form
    else SemG =                                else deep form is constructed from the
       [Sem | [SemSuf | SemPrSeq]]             semantic features of  verb, suffix, and
    ).                                         prefixes
```

Figure 6: Example of LDL code for generation of deep form.

The UVL-analysis forms a sound basis for efficient semantic analysis which maps the meaning of the user input to appropriate sentence deep structures. These deep structures correspond exactly to the semantic categories of the underlying database application, therefore they guarantee the correct and efficient semantic analysis of the input sentence, e.g. for our example:

```
[update, raw_material, 'St 50 H', quantity, 25]
```

5 Implementation

As implementation platform we used the deductive database language LDL (Logic Data Language) [22]. LDL was implemented at MCC as an efficient and portable prototype system for UNIX, called SALAD. An important facility represents the possibility of defining external predicates and functions in the procedural language C. SALAD consists of four main components which are strictly separated into different types of files:

- a schema for base predicates
- a set of facts representing the data
- a set of rules for deriving new predicates
- a set of query forms for generating access plans to stored data

In contrast to traditional logic programming languages like PROLOG, LDL contains neither navigational nor procedural semantics in favour of a purely declarative one, e.g. there exists no significance as to the order of rules. Therefore, LDL provides the basis for a 'purer' logic programming compared with most conventional logic languages [4]. In addition to that it possesses the usual features of database management systems, i.e. support for transactions, recovery, schema-based integrity, and efficient management of secondary storage. Finally, the most striking advantage is the separation of the fact base from the logical rules which allows the dynamic update of facts at run-time without the need for any recompilation.

Figure 6 shows the LDL code for the generation of the deep forms of derived substantives whereas the predicates in Figure 7 perform the first step of UVL-analysis, that is, the transformation of DFL to USL.

6 Case study

As field of application for our case study we have chosen a production planning and control system (PPC) as nucleus for a later extension to a full CIM system [24]. The PPC performs the mean-term scheduling of products and involved resources in the manufacturing processes, that is, material, machines, and labour. The resulting master production schedule forms the basis for the co-ordination of related business services such as engineering, manufacturing, and finance. The modelled enterprise makes precision tools using as basic strategies job order production and serial manufacture. Especially in this branch of industry there exists the strong need of modelling complex objects (e.g. the assembly of a part) and transitive relations such as operation sequences or sub-part hierarchies. As the efficient realisation of these demands exceeds the power of relational database technology, the application presents an excellent choice for deriving full advantage of the extended functionality of deductive database systems. Furthermore, the sophisticated functionality justifies the effective use of a natural language interface.

In order to obtain a well-defined reference model for the development of the natural language front end, we specified exactly 50 manipulations and 50 queries to the PPC which were implemented by LDL rules. The semantics of the functional part was formally represented as deep structures forming the semantic application model.

```
genusl(L, Ergebnis) ←                          generates USL out of DFL
    suchbeg(L, L2),                            searches for begin of unknown value
    if(L2 ~= [ ]                               if unknown value exists
    then                                       then
        zusfg(L2, Rest, Eintrag),                  creates sub-list for unknown value
        genusl(Rest, Eintrag2),                    recursive call
        Ergebnis = [Eintrag|Eintrag2]              inserts unknown value into USL
    else                                       else
        Ergebnis = [ ]).                           empty list is returned
genusl([ ], [ ]).                              exit rule of recursion
suchbeg([Eintrag|Rest], L) ←                   searches for next unknown value
    aggregate(auswahl, Eintrag, Eintrag2),     retrieves entry from set of interpretations
    Eintrag2=(_, Kat, _),                      retrieves category of actual entry
    if(Kat=unknown                             if category equals unknown
    then                                       then
        L=[Eintrag|Rest]                           list of remaining entries is returned
    else                                       else
        suchbeg(Rest, Rest2),                      recursive call
        L=Rest2).
suchbeg([ ], [ ]).                             exit rule of recursion
zusfg([Eintrag|Rest], Rest2,                   analysis of unknown value
        [(Wort, Typ)|Rest3]) ←
    aggregate(auswahl, Eintrag, Eintrag2),     retrieves entry from set of interpretations
    Eintrag2=(Wort, Kat, Typ),                 retrieves word stem, category, and type
    if(ntrkat(Kat)                             if no separating category
    then                                       then
        zusfueg(Rest, Rest2, Rest3)                joins parts of composed unknown value
    else                                       else
        Rest2=Rest,                                remaining categories are returned
        Rest3=[ ]).                                single unknown value is returned
zusfueg([Eintrag|Rest], Rest2, Ergebnis) ←     parts of unknown value are composed
    aggregate(auswahl, Eintrag, Eintrag2),     retrieves entry from set of interpretations
    Eintrag2=(Wort, Kat, Typ),                 retrieves word stem, category, and type
    if(fs(Kat, Typ, Rest)                      if criteria for continuation are satisfied
    then                                       then
        zusfueg(Rest, Rest2, Rest3),               recursive call
        Ergebnis=[(Wort, Typ)|Rest3]               result is computed
    else                                       else
        Rest2=[Eintrag|Rest],                      unknown value is added to rest of list
        Ergebnis=[ ]).                             empty list is returned
zusfueg([ ], [ ], [ ]).                        exit rule of recursion
```

Figure 7: LDL code for generation of USL.

As starting point for the implementation of the natural language interface, questionnaires were used to get 1000 realistic example sentences (10 for each command). The first step of implementation was to construct the dictionary as explained in Section 3. Table 1 shows the final number of entries for each category. The small total amount of 431 entries which were necessary to cover all 1000 input sentences illustrates the compact storage structure resulting from the application of the IDA architecture.

The main task of the final evaluation step was to verify the faultless mapping of the 1000 input sentences to the 100 commands of the PPC database system. After extensive testing cycles all natural language input was correctly analysed.

Besides the faultless operation, the basic requirement for the feasibility of the practical use of any database application is its performance. The main measure that has to be tested in this context is of course the response time. We performed careful tests and measuring, the results are shown in Table 2, the mean response time for each command category is given in seconds and hundredths of seconds. Furthermore, the results are divided in the response time of the interface, the database system, and the total response time.

Word category	Quantity
adjective	32
adjectival suffix	6
adverb	28
article	12
pronoun	33
conjunction	7
numeral	14
preposition	27
substantive	78
substantival suffix	9
verb	119
verb prefix	8
verb form	58

Table 1: Number of dictionary entries for PPC.

Commands	Interface	Database	Total
Insertions	5:29	5:19	10:48
Deletions	2:14	5:04	7:18
Updates	4:19	5:44	9:63
Complex manipulations	2:56	10:91	13:47
Simple queries	3:01	0:09	3:10
Queries with selection or grouping criterion	3:33	0:12	3:45
Queries for transitive relations	3:89	0:10	3:99
Complex queries	3:47	6:93	10:40

Table 2: Response times of PPC.

The overall mean response time for all commands was 7:71 (3:48 for interface and 4:23 for the database system), as hardware configuration we used a SUN SPARC 10 station.

7 Conclusion and future work

We have identified in our research the following main characteristics of natural language database interfaces in contrast to other fields of natural language processing: specific application domains with well-defined semantics, rather small delimited vocabularies, mappings to simple target representations, short input sentences without complex linguistic phenomena but including misspellings, ungrammatical or incomplete statements.

The main reason why many previous attempts to build successful natural language interfaces failed can be seen in the fact that those characteristics were neglected. The use of sophisticated techniques that maybe worked very well for other applications are simply oversized for database interfaces, therefore obstructing the way to efficient solutions.
In this context also the popular term 'domain-independent' must be regarded with critical reservation. Many authors claim to build domain-independent interfaces by ignoring the available application-specific data. As we have pointed out, only a domain-dependent interface can operate efficiently by making full use of the information which can be derived from the underlying database system. This is not necessarily in contradiction with portability because also such systems can be designed and implemented in view of later easy portation to other application areas.

Even if some previous work came to the same conclusions, the limitations of relational database technology represented an obstacle too high to overcome. Only with the emergence of deductive database technology there exists for the first time a computational framework that combines the required operational power with a purely declarative semantics leading the way to clear and concise realisations of natural language interfaces.

Our proposed architecture, the Integrated Deductive Approach to efficient natural language interfaces regards the interface in contrast to other existent work not as loosely coupled filter but as integral part of the database system itself. By the use of the powerful logic language provided by deductive databases we guarantee a homogenous mapping of the input to the corresponding database commands over all steps of analysis.

We introduced UVL-analysis as preliminary step for semantic analysis, a technique that is based on the evaluation of database values as well as the deep form of functional words. Therefore, no complete grammatical sentence structures are computed but syntactic information is only used if necessary for disambiguation.

We have proven the feasibility of our approach by an extensive case study, the design and implementation of a production planning and control system. For the creation of an appropriate test data collection we did not invent any artificial queries or manipulations but applied questionnaires in order to obtain realistic input sentences, therefore guaranteeing optimal customisation for later practical use.

Although all concepts in this paper have been developed for German, they incorporate the capacity to be applied also to other languages, especially to inflexional and free word order languages.

Further research in this topic will include portability studies to other applications and languages as well as investigations on the adaptive behaviour of natural language interfaces, e.g. the consideration of new functional words or changes to the application model. We believe that the ideas proposed in this work represent a challenging application of deductive databases as well as contribute an important step forward to the development of efficient natural language interfaces with widespread user acceptance.

References

[1] D.E. Appelt et.al. FASTUS: A Finite-state Processor for Information Extraction from Real-world Text. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1993.

[2] P. Bosc, M. Courant and S. Robin. CALIN: A User Interface Based on a Simple Natural Language. In *Proceedings of the ACM Conference in Information Retrieval*, 1986.

[3] A. Cappelli et.al. A Framework for Integrating Syntax and Semantics. In *Computational Models of Natural Language Processing*. B.G. Bara and G. Guida (eds.), North-Holland, Amsterdam, 1984.

[4] D. Chimenti et.al. The LDL System Prototype. *IEEE Transactions on Knowledge and Data Engineering*, Volume 2, Number 1, 1990.

[5] E.F. Codd. *Seven Steps to RENDEZVOUS with the Casual User*, IBM Research Report, J1333, San Jose Research Laboratory, 1974.

[6] A. Copestake and K.S. Sparck-Jones. Natural Language Interfaces to Databases. *Knowledge Engineering Review*, Volume 5, Number 5, 1990.

[7] F. Damerau. Operating Statistics for the Transformational Question Answering System. *American Journal on Computational Linguistics*, Volume 7, Number 1, 1981.

[8] T.E. Doszkocs. Natural Language Processing in Information Retrieval. *JASIS*, Volume 37, Number 4, 1986.

[9] D. Flickinger, C. Pollard and T. Wasow. Structure-sharing in Lexical Representation. In *Proceedings of the Annual Meeting of the ACL*, 1985.

[10] R.E. Frederking. *Integrated Natural Language Dialogue*. Kluwer, Boston, 1988.

[11] A. Gal and J. Minker. A Natural Language Database Interface that Provides Cooperative Answers. In *Artificial Intelligence Applications*. C.R. Weisbin (ed.), IEEE Press, Washington, 1985.

[12] B.J. Grosz. Transportable Natural-Language Interface. Problems and Techniques. In *Proceedings of the Annual Meeting of the ACL*, 1982.

[13] L.R. Harris. Natural Front Ends. In *The AI Business.* P.H. Winston and K.A. Prendergast (eds.), MIT Press, Cambridge, MA., 1984.

[14] G.G. Hendrix et.al. Developing a Natural Language Interface to Complex Data, *ACM Transactions on Database Systems*, Volume 3, Number 2, 1978.

[15] W. Höppner et.al. Beyond Domain Independence: Experience with the Development of a German Language Access System to Highly Diverse Background Systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1983.

[16] Y. Kambayashi. An Overview of a Natural Language-Assisted Database User Interface: ENLI. In *Proceedings of the IFIP World Computer Congress*, 1986.

[17] D.D. Lewis and R.M. Tong. Text Filtering in MUC-3 and MUC-4. In *Proceedings of the Message Understanding Conference*, 1992.

[18] P.C. Lockemann. Object-Oriented Databases and Deductive Databases: Systems Without Markets ? Market Without Systems ?. In *Proceedings of the International Conference on Database and Expert Systems Applications*, 1992.

[19] S.L. Lytinen. Dynamically Combining Syntax and Semantics in Natural Language Processing. In *Proceedings of the Conference of the AAAI*, 1986.

[20] P. McFetridge et.al. System X: A Portable Natural Language Interface. In *Proceedings of the Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, 1988.

[21] P. McFetridge and C. Groeneboer. Novel Terms and Coordination in a Natural Language Interface. In *Knowledge Based Computer Systems*. S. Ramani, R. Chandrasekar and K.S.R. Anjaneyulu (eds.), Springer, Berlin, 1990.

[22] S. Naqvi and S. Tsur. *A Logical Language for Data and Knowledge Bases*. Computer Science Press, Rockville, 1989.

[23] W.C. Ogden and A. Sorknes. What Do Users Say to their Natural Language Interface ?. In *Proceedings of the Conference on Human-Computer Interaction*, 1987.

[24] A.-W. Scheer. *CIM. Der computerge-steuerte Industriebetrieb* (in German). Springer, Berlin, 1988.

[25] M. Schröder. Evaluating User Utterances in Natural Language Interfaces to Databases. *Computers and AI*, Volume 7, Number 4, 1988.

[26] S.P. Schwartz. Problems with Domain-Independent Natural Language Database Access Systems. In *Proceedings of the Annual Meeting of the ACL*, 1982.

[27] S.M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes, Number 4, Chicago Univ. Press, Chicago, 1986.

[28] H.R. Tennant et.al. Menu-Based Natural Language Understanding. In *Proceedings of the Annual Meeting of the ACL*, 1983.

[29] H. Trost and E. Buchberger. Datenbank-DIALOG: How to Communicate with your Database in German (and Enjoy it). *Interacting with Computers*, Volume 2, Number 3, 1990.

[30] J.D. Ullman and C. Zaniolo. Deductive Databases: Achievements and Future Directions. *ACM SIGMOD Record*, Volume 19, Number 4, 1990.

[31] H. Uszkoreit. Categorial Unification Grammars. In *Proceedings of the International Conference on Computational Linguistics*, 1986.

[32] H. Uszkoreit. Syntaktische und semantische Generalisierungen im strukturierten Lexikon (in German). In *Proceedings of the German Workshop on Artificial Intelligence and Österreichische Artificial-Intelligence-Tagung*, 1986.

[33] M. Wallace. *Communicating with Databases in Natural Language.* Horwood, Chichester, 1984.

[34] D.L. Waltz and B.A. Goodman. Writing a Natural Language Data Base System. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1977.

[35] W. Winiwarter and A M. Tjoa. Natural Language Interfaces as Integrated Constituents of Deductive Databases. In *Proceedings of the Symposium on Next Generation Database Systems and Their Applications*, 1993.

[36] W. Winiwarter. Extended CUG for Free Word Order Languages and its Efficient Implementation within an IDA Architecture. In *Proceedings of the Joint Conference of the Asian Conference on Language, Information and Computation and the Pacific Asia Conference on Formal and Computational Linguistics*, 1994.

[37] W. Winiwarter. MIDAS - the Morphological Component of the IDA System for Efficient Natural Language Interface Design. In *Proceedings of the International Conference on Database and Expert Systems Applications*, 1995.

[38] W.A. Woods, R.M. Kaplan and B. Nash-Webber. *The Lunar Sciences Natural Language Information System*. Bolt Beranek and Newman, Cambridge, MA., 1972.