Datenbank-DIALOG and the Relevance of Habitability

Harald Trost*

DFKI GmbH
Stuhlsatzenhausweg 3
D-6600 Saarbrücken, Germany

Wolfgang Heinz, Johannes Matiasek
Ernst Buchberger
ÖFAI
Schottengasse 3, A-1010 Wien, Austria
john@ai.univie.ac.at

1 Introduction

The paper focusses on the issue of habitability and how it is accounted for in *Datenbank-DIALOG*[1]. Examples from the area of comparisons and measures—both important for many application domains and non-trivial from a linguistic point of view—demonstrate how design strategies can support the development of a habitable system.

Datenbank-DIALOG is a German language interface to relational databases. Since the development of a first prototype (1985-88) it has been tested in different environments and continually been improved. Currently, in a large field test, *Datenbank-DIALOG* interfaces to a database about AI research in Austria. Questions sent by email[2] are answered automatically.

The system consists of four main components. The *scanner* breaks up the natural language query into tokens for morphological analysis. The *parser* performs syntactic and semantic analysis creating one or—in case of ambiguities—more caseframes containing the query representation at the domain level. The *interpretation* of the query is performed in three stages. The mapping from domain-level to database-level predicates results in a DB-Caseframe, then a linearization step produces the Logical Form and finally a syntactic transformation leads to an SQL query. The *answer* is then given directly by the DBMS as the result of executing the SQL query.

2 Habitability

Experiments with NLIs indicate that the linguistic coverage of state-of-the-art systems is adequate. Savings in training time outweigh problems with queries not handled. Very important though is that the system behaves in a predictable way, i.e. is habitable, so that users can learn the types of acceptable queries very fast. Otherwise, they will either face a continuously high rejection rate or—more likely, since humans adapt much better than computers—formulate their queries in an unnecessarily simple and inefficient way.

Habitability cannot be judged solely on syntactic coverage. Queries must be correctly interpreted syntactically, semantically and pragmatically. While syntactic coverage depends solely on the parser, semantic and pragmatic coverage must be considered with respect to the contents of the database to which the NLI connects.

The grammar of *Datenbank-DIALOG* is completely domain-independent, designed to make the accepted sublanguage as consistent as possible. Recent advances of linguistic theory were incorporated in its development, thus also facilitating implementation and maintenance. Two examples for this strategy are the treatment of determiners and verb-second (V2).

Using results of Generalized Quantifiers Theory for natural language quantifiers (e.g. conservativity) a formal correspondence between GQ-formulas (representing the logical form of a query) and SQL-statements (formulas over the relational calculus) was established and implemented. This gives a sound theoretical basis for semantic interpretation and SQL generation. All extensional natural language determiners can be handled—matching the extensional nature of databases.

In German finite verbs occur in second position in main clauses (V2) and in final position in subordinate clauses. Ideas from GB-Theory are used for a uniform treatment. V2 is considered to be the result of a movement from an underlying final position in the verb cluster to clause initial complementizer position. This movement is implemented as a relation between the "moved" finite verb and its trace. In the case of main clauses, V_{fin} is "moved back" to the end of the verb cluster, and now the same mechanism applies uniformly. Thus both clause types are subject to the same syntactic and semantic constraints (which thus need only be stated once) and give rise to the same interpretation.

3 Comparisons

A central concern in querying databases are comparisons between various kinds of objects. Comparisons involve a relation between values associated with a dimension and units of measure. Values may be given explicitly or implicitly by derivation (thus including superlatives).

*at ÖFAI during development of *Datenbank-DIALOG*

[1]The development was carried out at the Austrian Research Institute for Artificial Intelligence (ÖFAI) jointly with "Software Management GmbH", Vienna and has been sponsored by the Austrian Government within the "Mikroelektronik Förderungsprogramm, Schwerpunkt S7". See also: H. Trost, E. Buchberger, W. Heinz, Ch. Hörtnagl and J. Matiasek. Datenbank-DIALOG - A German Language Interface for Relational Databases. In *Applied Artificial Intelligence*, 1:181-203, 1987.

[2]Email address: aiforsch@ai.univie.ac.at

Linguistic means for **expressing comparison** vary widely. Usually, comparison is associated with gradable adjectives and adverbs in various syntactic constructions: *hat ein höheres Gehalt* (Aux+A/NP); *verdient mehr* (V+Adv); *mit einem höheren Gehalt* (A/PP). The interpretation of those expressions should be the same. *Datenbank-DIALOG* uses a compositional semantics and separates the lexical item from the underlying semantic relation, which may be shared by different words.

More problems arise when specifying the value for the comparison: *ein höheres Gehalt als 20.000,–*; *ein Gehalt von mehr als 20.000,–*; *mehr als 20.000,– Gehalt*; *verdient mehr als 20.000,–*. The comparative and the value may be adjacent or not, and show up as PPs, complex determiners or adverbial phrases. All these constructions map onto the same semantic representation, a relation, a value along with a dimension and a unit—thus allowing to compare values with different units—and a compared object. This assures a uniform semantic treatment.

In *verdient mehr als Dr. Haid* the value is specified only implicitly by referring to the salary of Dr. Haid. Despite the different structure of the corresponding SQL queries the user will hardly notice this fundamental difference. For a habitable system it is necessary to provide solutions to both types of comparisons. *Datenbank-DIALOG* recognizes the different interpretations by the semantic type associated with the value of the phrase to be compared. If the value has the correct dimension, it may safely be inserted as an argument into the comparison relation. Otherwise, *Datenbank-DIALOG* constructs a subquery giving the value by using the dominating relation and fitting the comparison object into the "subject" slot of the attribute. The resulting structure is processed analogously to a top-level query. As a consequence, anaphora resolution may be applied enabling *Datenbank-DIALOG* to give a correct interpretation of *Wer$_i$ verdient mehr als sein$_i$ Vorgesetzter?*

Domain predicates need not uniquely determine the relation and attributes of a corresponding predicate in the **database**. *Datenbank-DIALOG* splits the interpretation of an utterance into two stages: An interpretation in the domain model, i.e. a caseframe, which is then mapped (using a translation table) to an interpretation in the database model, i.e. a DB-caseframe.

This approach allows to interpret superficially similar queries as quite different SQL queries. Attributes with the same meaning stored in different tables (*nurse-salary* vs. *doctor-salary*) can be treated as well as derived attributes (salary computed from *basic + variable salary*)—in short, the user should not need to know about the actual encoding of information. An interesting instance of this principle is the interpretation of *Wieviele Patienten behandelt Dr. Haid.* Whereas in one database model the number of patients is stored *explicitly* and can be treated analogously to the salary above, other database models contain this "attribute" only *implicitly*: the number of patients has to be computed (counted) by the SQL query. To obtain these quite different interpretations, only a different mapping of the (contents of the) argument-slot of the predicate *Treatment* in the translation step between domain and database level is required.

A special case (where implicit attributes must be made explicit) are queries involving the comparison of two subqueries. This cannot be expressed in a single SQL query. A temporary table has to be created containing the relevant count-attribute together with information on the object bearing that attribute. The actual comparison can then be made with the now explicit attribute.

Most problems with comparatives also occur with **superlatives** and are dealt with in an analogous way. One interesting phenomenon which has no direct parallel in comparative structures shows in *Welcher Arzt, der in der Ambulanz arbeitet, verdient am meisten?* In most cases *Who has the highest salary among the doctors working in the casualty department?* is the most plausible interpretation and should be preferred. To produce this reading, a kind of copying has to be performed: not only must the dominating relation be copied but also the restrictions on the subject slot (i.e., on the bearer of the attribute) have to be inherited. In *Datenbank-DIALOG* this copying works on the caseframe representation, and thus is able to handle restrictions resulting from different syntactic constructions, such as the lexicon (*Ambulanzarzt*), APs (*in der Ambulanz arbeitende Arzt*), PPs (*Arzt aus der Ambulanz*), NPs (*Arzt der Ambulanz*) and relative clauses (*Arzt, der in der Ambulanz arbeitet*). All these constructions end up as modifications in the caseframe due to the compositional nature of our approach. Thus a unified solution for inheritance of modifiers in their various forms is achieved.

A correct comparison is only possible if compared values are of the same **dimension** and share a **unit of measure**. Differences and incompatibilities may arise in different places: from special formatting conventions (e.g. 20000, 20.000,–, $20), when the user specifies a dimension and unit of measure verbatim (e.g. "10 Meter"), from the database, where comparable columns may be associated with different units of measure. *Datenbank-DIALOG* solves this problem—by defining a normalized form associating values with units and transformation rules between measures of different units—at the *scanner* level (patterns, e.g. date formats), at the *parser* level (linguistic information to fill the slots in the normalized value frame), at the *interpretation* level (procedures to transform constant values from one unit to another), at the *database* level (transformation functions of the query language).

4 Summary

Habitability is a most important feature of NLIs. Using comparison as example we have shown how the design of *Datenbank-DIALOG* enhances habitability, in particular by: giving a uniform interpretation to semantically equivalent user queries of different syntactic and morphological appearance; enabling users to enter data in the form most convenient to them (formatting, unit conversion); removing the need for users to know about the database representations of the concepts they use (domain concepts vs. database relations and attributes, implicit functions, unit conversion); making ambiguities explicit; and incorporating presuppositions (relation and restriction copying).