Comparisons in NLIs: DATENBANK-DIALOG and the Relevance of Habitability

Harald Trost*
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Stuhlsatzenhausweg 3
D-6600 Saarbrücken, Germany

Wolfgang Heinz, Johannes Matiasek, Ernst Buchberger
Österreichisches Forschungsinstitut für Artificial Intelligence
Schottengasse 3
A-1010 Wien, Austria
Email: {harald,wolfgang,john,ernst}@ai-vie.uucp

Abstract

This paper describes certain aspects of DATENBANK-DIALOG[1], a German language interface to relational databases developed at the Austrian Research Institute for Artificial Intelligence. Besides giving a short overview of the system architecture it emphasizes the issues of portability and habitability and how they are being tackled in the design of DATENBANK-DIALOG. To demonstrate how design strategies support the development of a habitable system we take examples from the area of comparisons and measures, both of which are important for many application domains and nontrivial from a linguistic point of view.

DATENBANK-DIALOG has been fully implemented and is accessible world-wide via email coupled to a database containing information about all Austrian AI-projects.

*Österreichisches Forschungsinstitut für Artificial Intelligence during the period of development of DATENBANK-DIALOG

[1]The development of DATENBANK-DIALOG was carried out jointly with "Software Management GmbH", Vienna and has been sponsored by the Austrian Government within the "Mikroelektronik Förderungsprogramm, Schwerpunkt S7".

1 Introduction

DATENBANK-DIALOG is a German language interface to relational databases. It enables the casual user to enter questions in the form of relatively unrestricted German sentences. These utterances are translated into SQL statements and passed on to the database management system (DBMS) which in turn provides the user with an answer. A first prototype of DATENBANK-DIALOG was developed at the Austrian Research Institute for Artificial Intelligence from 1985 to 1988. Since then the system has been tested in different environments. As a result of continuing development, the performance of DATENBANK-DIALOG has improved considerably. Currently a large field test is taking place. DATENBANK-DIALOG has been interfaced to a database containing information about Artificial Intelligence research in Austria (cf. Trappl et al. (1991)). Questions to that system can be sent by electronic mail (Email-address: `aiforsch%ai-vie.uucp@relay.eu.net`) and are answered automatically.

In this paper we will emphasize those properties of DATENBANK-DIALOG which are crucial for smooth interaction, both with the system manager and the end-user. Accordingly, the main points are portability and habitability. First, we will show how different aspects of portability have been given consideration in DATENBANK-DIALOG. Then a short overview of the architecture of DATENBANK-DIALOG is given. Finally we will go into the crucial issue of habitability, which is further exemplified by a detailed example study on how comparisons and measures are dealt with in DATENBANK-DIALOG.

2 Portability

Portability is an important aspect for every software system. With natural language interfaces there are many different levels of portability. In this section we will have a look at each of these levels in turn and show how they have been accounted for in DATENBANK-DIALOG.

- **Hardware Portability**: DATENBANK-DIALOG is highly portable in this respect: it has been ported to various types of computers, among them machines from DEC, Nixdorf and Apollo. Hardware portability was secured by the development environment chosen: DATENBANK-DIALOG was developed in LISP[2] to enable rapid prototyping. A cross compiler translates the code into standard FORTRAN IV. As a result the system runs in practically every hardware environment with minimal adaptation.

- **Portability with respect to the DBMS**: DATENBANK-DIALOG translates German language utterances into SQL queries. Since SQL can be considered to be the de-facto standard query language for relational databases, the decision to use it as

[2] actually a subset of INTERLISP

output language followed quite naturally. Although SQL is standardized to some extent, versions differ slightly for different DBMSs. In order to make DATENBANK-DIALOG easily adaptable, the query generator has been separated into a module of its own. That way adaptation to different SQL dialects is easy.

The use of DATENBANK-DIALOG with non-relational databases was not considered because older types of DBMSs do not provide a flexible enough access to data. But it is precisely this flexibility which is needed to make natural language access really useful.

- **Portability with respect to the domain**: Domain knowledge encompasses all the facts which are important for a certain application. It must be distinguished from the way these data are actually represented in the database (the so-called database model) on the one hand and the purely linguistic knowledge (the grammar) on the other hand. In DATENBANK-DIALOG domain knowledge is stored in a so-called meaning lexicon, which maps lexical entries onto domain-level predicates and provides syntactic and semantic restrictions for their arguments. Two advantages result from this intermediate level of representation: First, separating grammatical from domain knowledge leads to equal linguistic coverage in different applications, since no changes to linguistic knowledge are required when modelling a new application. Second, abstracting away from the particular database model allows for a linguistically oriented modelling of the domain, thus providing a smooth interaction between linguistic and domain knowledge.

- **Portability with respect to the data model**: As mentioned above, DATENBANK-DIALOG discriminates between the domain model and the database model. The database model (as the name suggests) is strictly modelled according to the database. There is an explicit translation step between representations in the domain model (so-called caseframes) and representations in the database model (so-called DB-caseframes). This translation step is done with the help of a translation table which specifies explicitly the mappings between the two models. This gives more flexibility with regard to linguistic coverage because the grammar component need not take into account the database model where facts may be expressed in a way very different from the way they are expressed in German.

  This separation makes it possible to use the same linguistic domain model for different databases. Only the translation table must be rewritten to conform to the respective database model. Since the bulk of the work in adapting to a new domain lies in the creation of the linguistic domain model (vocabulary, concept hierarchy, etc.) this saves a lot of effort. It is also a prerequisite for providing users of DATENBANK-DIALOG with a ready-to-use core version.

- **Portability across different natural languages**: As has already been stated, DATENBANK-DIALOG was specifically created for German. We did not seek portability between different natural languages because we think that – with the current

state-of-the-art in computational linguistics – it is not possible to write a parameterizable grammar component. Instead, we concentrated on a linguistic core system which handles a considerable subset of German in an efficient way.

3 Architecture of DATENBANK-DIALOG

DATENBANK-DIALOG consists of four main components. At the *scanner* level, the natural language query is broken up into tokens and a pattern matching module handles input following special formatting conventions such as dates (e.g. "15. 1. 1991", "91/01/15", "15-JAN-91", "8:30pm" ...), amounts ("20,000", "20.000,–" ...), numerical data along with a unit of measure (e.g. "2,54cm", "1.0in", "$20" ...), abbreviations (e.g. "Dr.", "Univ.-Prof.", "PhD" ...), etc. Then a morphological analysis is performed, which is especially important in languages like German, where many syntactic features (such as number, case, tense etc.) are marked by inflection. The resulting representation of the input query is then presented to the *parser* which performs the syntactic and semantic analysis. German

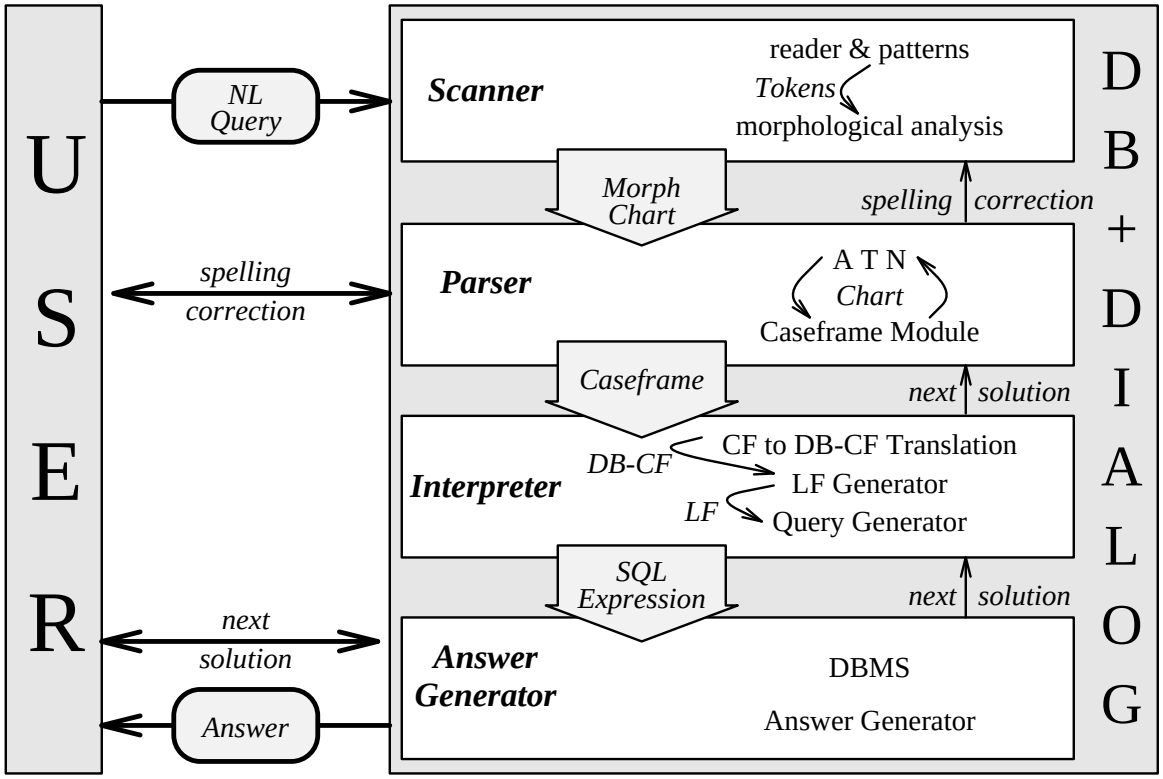

Figure 1: Architecture of DATENBANK-DIALOG

is a language with fairly free word (i.e. constituent) order in clause-like phrase types (sentences, APs), but fixed word order in other types of phrases (such as NPs, PPs). For that reason we have elected to use two different strategies for parsing: An augmented transition

network (ATN) grammar handles the phrases with fixed word order and a caseframe component deals with free constituent order. These two components interact closely with each other communicating by means of a chart (for details see Trost et al. (1988)). The parse results in one or more (if there are ambiguities) caseframe(s) containing the representation of the query at the domain level.

The *interpretation* of the query is performed in three stages. First, the mapping from domain-level to database-level predicates is performed, resulting in a DB-Caseframe, then a linearization step produces the Logical Form (LF) and finally a syntactic transformation leads to the interpretation in the form of an SQL query.

The *answer* is then given directly by the DBMS as the result of executing the SQL query. In the case of ambiguities, the user is given the possibility of selecting the different interpretations suggested before the queries are executed.

4 Habitability

Experiments with NLIs indicate that the linguistic coverage of state-of-the-art systems is adequate since savings in training time outweigh the problems encountered with queries the system cannot handle. People usually adapt very well to grammatical restrictions in the language they use (Hendler & Michaelis (1983)). What is very important though is that the system perform in a way predictable by the user, i.e. that the system is habitable (Krause (1982), p.15ff). Users should be able to learn very fast which types of queries are acceptable to the system. Otherwise they will either have to face a continuously high rejection rate or—what is more likely, owing to the fact that humans adapt much better than computers—they will formulate their queries in an unnecessarily simple and inefficient way (Tennant (1980)).

In the light of these facts, what are then useful restrictions in defining the accepted sublanguage? Some properties of natural languages seem to be so ubiquitous that humans cannot do without them. Such properties are, for example, the use of anaphoric expressions (pronouns) and elliptic constructions. Therefore, every NLI will need at least a basic capacity to deal with them. DATENBANK-DIALOG has been equipped with such a capability.

One obvious restriction concerns the vocabulary. Besides function words that have to be present in any case, there will be no problems with restricting the vocabulary. More critical are restrictions on word senses. If they are not clearly motivated by the domain model there will be problems. Separating the domain model from the database model,a factor which contributes in DATENBANK-DIALOG to increased portability (see above), also helps in this respect.

As stated above, clear syntactic restrictions—there are in fact relatively few in DATENBANK-DIALOG—are readily accepted by the user. However, syntactic coverage cannot be judged in isolation. Queries are accepted only if they are correctly interpreted syntactically, semantically and pragmatically. While syntactic coverage depends solely on the parser

of the NLI, semantic and pragmatic coverage must be considered with respect to the contents of the database to which the NLI connects. We will show the interaction of syntax, semantics and pragmatics in an example study on the treatment of comparison in the next chapter. But first, we should like to say a few more words about the grammar of the system.

Unlike some other NLIs, DATENBANK-DIALOG has a separate grammar component which is completely domain-independent. The grammar was designed to make the accepted sublanguage as consistent as possible. In order to tackle this goal, we tried to formulate the grammar in a clean and concise way. Accordingly, much effort was undertaken to incorporate recent advances of linguistic theory in the development of the grammar component for DATENBANK-DIALOG, thus also facilitating implementation and maintenance.

Two examples for this strategy are the use of Generalized Quantifiers Theory (Barwise & Cooper (1981), Keenan & Stavi (1986)) for the representation of logical form and the translation into SQL, and the implementation of a GB-type theory of verb second (Haider (1985)) for a uniform treatment of different clause types (main and subordinate).

Generalized Quantifier Theory gives a general framework for the treatment of nonstandard determiners (e.g. "seven", "between 2 and 10", "many") that includes the standard logical quantifiers "for all" and "exists" as special cases. The quantifier is viewed as a combination of the determiner and its restricting predicate. The use of the determiner as a relation between two predicates is made explicit.

Using the results of Keenan & Stavi (1986) for natural language quantifiers (e.g. conservativity) a formal correspondence between GQ-formulas (our means of representing the logical form of a query) and SQL-statements (representing formulas over the relational calculus) can be established and straightforwardly implemented. This gives us a sound theoretical basis for semantic interpretation and SQL generation. As a result, all extensional natural language determiners can be handled. Given the extensional nature of relational database systems this treatment suffices in a query system (cf. Heinz & Matiasek (1989) for details of our solution).

A cumbersome problem for systems dealing with German is the verb-second phenomenon—finite verbs occur in second position in main clauses and in final position in subordinate clauses.

(1) *a)* *Wieviele Patienten* **werden** *von Dr. Haid behandelt?*
(How many patients are treated by Doctor Haid?)

b) *Sag mir, wieviele Patienten von Dr. Haid behandelt* **werden**.
(Tell me how many patients are treated by Doctor Haid.)

Both utterances in *(1)* should get the same interpretation. To avoid writing completely different subgrammars for main and subordinate clauses ideas from GB-Theory have been used for a uniform treatment.

In GB-theory verb second is considered to be the result of a movement from an underlying

final position in the verb cluster to clause initial complementizer (C) position (cf. Haider (1985)). In constituent questions, the specifier-position in front of C is filled with the wh-phrase, in yes/no-questions it remains empty. In subordinate clauses the C position is filled with a complementizer (e.g. *ob* ("whether") or the wh-phrase).

(2) *a)* *$[_{CP}$ Wieviele Patienten $[_C$ werden$]_i$ von Dr. Haid behandelt $t_i]$*
(How many patients are treated by Doctor Haid?)

b) *Sag mir $[_{CP}$ wieviele Patienten [von Dr. Haid behandelt werden]]*
(Tell me how many patients are treated by Doctor Haid.)

c) *$[_{CP}$ $[_C$ Werden$]_i$ von Dr. Haid mehr als 30 Patienten behandelt $t_i]$*
(Are more then 30 patients being treated by Doctor Haid?)

d) *Sag mir $[_{CP}$ ob [von Dr. Haid mehr als 30 Patienten behandelt werden]]*
(Tell me whether more then 30 patients are being treated by Doctor Haid.)

In the implementation of DATENBANK-DIALOGthis movement is interpreted as a relation between the "moved" finite verb and its trace. In the case of main clauses, V_{fin} is added at the end of the verb cluster ("moved back"), and now the same mechanisms (ATN-subnet and Caseframe-principles) apply uniformly in V2 and verb-final sentences. 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.

5 Example Study: Treatment of Comparison

We will now have a closer look at the treatment of comparison and measurements in DATENBANK-DIALOG to demonstrate our efforts to create a habitable system. A central concern in querying databases is dealing with comparisons between various kinds of objects. In linguistic terms comparison is mainly associated with gradable adjectives and adverbials.

5.1 Variation in expression

The means for expressing comparison vary widely. Consider *(3)*:

(3) *a)* *Welche Ärzte haben ein höheres Gehalt als 20.000,– ?*
(Which doctors have a salary higher than 20.000,– ?)

b) *Welche Ärzte verdienen mehr als 20.000,– ?*
(Which doctors earn more than 20.000,– ?)

c) *Gib mir alle Ärzte mit einem höheren Gehalt als 20.000,– .*
(Show all doctors with a salary higher than 20.000,– .)

All the utterances in *(3)* should map onto the same database query, namely the SQL statement given in *(4)*:

(4)
```
select ID, NAME
  from DOCTOR
 where SALARY > 20000;
```

This means that the interpretation of sentences *(3a-c)* should be the same. This is achieved by using a compositional semantics and by separating the lexical item (word) from the underlying semantic relation, which may be used with different words. Thus in our example we have the underlying predicate `SALARY` with two arguments describing the thematic roles (deep cases) `RECIPIENT` and `VALUE`, both with the appropriate semantic restrictions:[3]

(5)
```
pred: SALARY
  a1: RECIPIENT (sem-restr PERSON)
  a2: VALUE     (dimension MONEY)
```

The predicate `SALARY` shown in *(5)* is associated with different words in the lexicon. Every lexical entry gives the syntactic restrictions on the fillers of the arguments. Both the verb *verdienen* (earn) and the noun *Gehalt* (salary) are mapped onto `SALARY`. The syntactic information binds `RECIPIENT` to a nominative NP (in active contexts) in the case of the verb and to a genitive NP or a PP with preposition *von* in the case of the noun.

(6) *a)* *Wieviel verdient* $[_{NP_{nom}}$ *Dr. Haid] ?*
(How much does Doctor Haid earn?)

b) *Gib mir das Gehalt* $[_{PP}$*von Dr. Haid]?*
(Show me the salary of Doctor Haid?)

More problems arise when specifying the `VALUE` for `SALARY`:

(7) *a)* *Welche Ärzte haben ein höheres Gehalt als 20.000,– ?*
(Which doctors have a higher salary than 20.000,– ?)

b) *Welche Ärzte haben ein Gehalt von mehr als 20.000,– ?*
(Which doctors have a salary of more than 20.000,– ?)

c) *Welche Ärzte haben mehr als 20.000,– Gehalt ?*
(Which doctors have more than 20.000,– salary?)

d) *Welche Ärzte verdienen mehr als 20.000,– ?*
(Which doctors earn more than 20.000,– ?)

In *(7a)* the comparison consists of the comparative *höheres*, an adjective phrase which gives the type of comparison, and the object it is compared to (i.e. the value) *als 20.000,–*. In *(7b-d)* both are adjacent but are realized as distinct syntactic entities: a postnominal PP modifier in *(7b)*, a complex determiner phrase in *(7c)*, and an adverbial phrase in *(7d)*.

[3] domain level caseframes like *(3)* and subsequent examples are given in simplified versions

All of these constructions map onto the same semantic representation, in our example a relation (>), a value (20.000) along with a dimension and a unit (money in Austrian Schillings), and a compared object. Therefore a uniform treatment is assured from a semantic point of view. Note that because a unit is associated, comparisons with values given in different units (e.g. $ or DM) are possible. We will return to this aspect later when discussing the relation between domain and database models.

5.2 Derived Comparison

Up to now we have assumed a constant value as given. But compare sentences *(8a)* and *(8b)*

(8) a) *Welche Ärzte verdienen mehr als 20.000,– ?*
(Which doctors earn more than 20.000,– ?)

b) *Welche Ärzte verdienen mehr als Dr. Haid?*
(Which doctors earn more than Doctor Haid?)

In *(8a)* the value is specified with the constant *20.000*. The interpretation of this utterance is given in *(4)*. In contrast, in *(8b)* the value is specified only implicitly by referring to the salary of Doctor Haid. The resulting SQL query should be[4]

(9)
```
select ID, NAME
  from DOCTOR
 where SALARY > (select MAX (GEHALT)
                   from DOCTOR
                  where NAME = 'Haid');
```

Although *(4)* and *(9)* have a different structure the user will hardly notice the fundamental difference between query *(8a)* and *(8b)*. For a habitable system it is therefore necessary to provide solutions to both types of comparisons.

In DATENBANK-DIALOG the different interpretations are recognized by the semantic type that is 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 (in our case ">"). Otherwise, DATENBANK-DIALOG tries to construct a subquery giving a value by using the dominating relation (in our example `SALARY`) and fitting the comparison object into the "subject" slot of the attribute. In *(9b)* this results in a subquery equivalent to

[4]SQL requires the subquery to return an unique value for comparison with >, hence then `MAX`-function.

(10) Wieviel verdient Dr. Haid?
(How much does Doctor Haid earn?)

The resulting structure is processed in a manner analogous to the top-level query. As a consequence, anaphora resolution may be applied to this structure enabling DATENBANK-DIALOG to give a correct interpretation—resulting in the correlated subquery *(11b)*[5]—for *(11a)*

(11) *a)* *Wer$_i$ verdient mehr als sein$_i$ Vorgesetzter?*
(Who$_i$ earns more than his$_i$ superior?)

b)
```
select A1.ID, A1.NAME
  from DOCTOR A1
 where A1.SALARY > (select MAX(A2.SALARY)
                      from DOCTOR A2
                     where A1.SUPERIOR = A2.ID);
```

5.3 Domain Model vs. Database Model

Domain predicates like `SALARY` need not uniquely determine the relation and attributes of a corresponding predicate in the database. DATENBANK-DIALOG therefore splits the interpretation of an utterance into two stages: first, an interpretation in the domain model (i.e. a caseframe) is given. Secondly, this caseframe is interpreted to yield an interpretation in the database model (a DB-caseframe). The transformation step between the two structures is performed with the use of a translation table. This approach enables DATENBANK-DIALOG to interpret superficially similar queries in terms of quite different SQL queries.

Let us consider the following two structurally similar utterances:

(12) *a)* *Welche Ärzte verdienen mehr als 20.000,–?*
(Which doctors earn more than 20.000,– ?)

b) *Welche Schwestern verdienen mehr als 20.000,–?*
(Which nurses earn more than 20.000,– ?)

On the domain level this will lead to quite similar structures:

(13) *a)*
```
pred: SALARY
  a1: ?, (sem-restr DOCTOR)
  a2: >  20000
```
b)
```
pred: SALARY
  a1: ?, (sem-restr NURSE)
  a2: >  20000
```

Assume that our example database stores data about doctors and nurses in different ta-

[5] The attribute `SUPERIOR` shall contain the ID of the superior of the doctor.

bles. This would lead to an ambiguity in the translation of the domain predicate SALARY. The translation table would contain two entries for SALARY with different restrictions on argument a1. Therefore utterances *(12a)* and *(12b)* would lead to SQL statements with different relations involved:

(14) *a)*
```
select D.ID, D.NAME
  from DOCTOR D, DOCTOR_SALARY DS
 where D.ID = DS.ID and DS.SALARY > 20000;
```
b)
```
select N.ID, N.NAME
  from NURSE N, NURSE_SALARY NS
 where N.ID = NS.ID and NS.SALARY > 20000;
```

Also, the salary need not be specified in a single attribute in the database, a fact that should not be of concern to the user. Such information may also be represented in the translation table (separately for each relation). Let us assume that the nurses' salary consists of a basic salary and a variable salary (contained in the attributes BASIC_SALARY and VAR_SALARY of NURSE_SALARY respectively). The interpretation for *(12b)* must then be

(15)
```
select N.ID, N.NAME
  from NURSE N, NURSE_SALARY NS
 where N.ID = NS.ID and
       (NS.BASIC_SALARY + NS.VAR_SALARY) > 20000;
```

This leads to another problem usually encountered in realistic applications af natural language database interfaces. The user should not need to know about the actual encoding of information. Consider:

(16) *Wieviele Patienten behandelt Dr. Haid?*
(How many patients does Dr. Haid treat?)

As an answer the user expects the number of patients which were treated by Doctor Haid. In the domain level the query is represented as:

(17)
```
pred: TREATMENT
  a1: x, (sem-restr DOCTOR)
         (modifier (pred: NAME
                    a1: x
                    a2: "Haid"))
  a2: (? ; COUNT)
```

The necessary information can be realized in quite different database models. Some of the possibilities are given in *(18)*:

(18)

a)

Relation	*Attributes*	*Comment*
DOCTOR	ID	*identification*
	NAME	*name*
	NR_PATIENTS	*number of patients*

b)

Relation	*Attributes*	*Comment*
PERSON	ID	*identification*
	NAME	*name*
	STATUS	*doctor/nurse/patient*
TREATMENT	ID_DOCTOR	*id of doctor*
	ID_PATIENT	*id of patient*

c)

Relation	*Attributes*	*Comment*
DOCTOR	ID	*id of doctor*
	NAME	*name of doctor*
PATIENT	ID	*id of patient*
	NAME	*name of patient*
	ID_DOCTOR	*patient's doctor*

Depending on the actual database model one gets of course quite different SQL queries:

(19) *a)*
```
select NR_PATIENTS
  from DOCTOR
 where NAME = 'Haid';
```
b)
```
select COUNT (UNIQUE T.ID_PATIENT)
 from PERSON P, TREATMENT T
where P.NAME = 'Haid' and
      T.ID_DOCTOR = P.ID and
      P.STATUS = 'DR';
```
c)
```
select COUNT (UNIQUE C.ID)
 from DOCTOR D, PATIENT C
where D.NAME = 'Haid' and
      C.ID_DOCTOR = D.ID;
```

There is a fundamental difference between *(19a)* and *(19b,c)*. Whereas in the database model *(18a)* the attribute NR_PATIENTS is contained in the database *explicitly* and can be treated analogously to SALARY above, the other two database models contain this "attribute" only *implicitly*. This means that the number of patients has to be computed (i.e. counted) by the SQL query. To obtain these quite different interpretations for *(17)*, DATENBANK-DIALOG requires only a different mapping of the (contents of the) a2-slot of TREATMENT in the translation step between domain and database level.

A special case, where implicit attributes have to be made explicit in the database, occurs with queries such as

(20) Wer behandelt mehr Patienten als Dr. Haid?
Who is treating more patients than Dr. Haid?

Since comparison of two subqueries is not possible within a single SQL query, the query has to be split up into two parts. First, a temporary table has to be created containing the relevant count-attribute together with information on the object bearing that attribute. Secondly, the actual comparison can be made with the now explicit attribute. This results in[6]

(21)
```
create table TEMP as
   (select COUNT(UNIQUE P.ID) "ANZAHL", D.ID "ID", D.NAME "NAME"
      from DOCTOR D, PATIENT P
     where P.ID_DOCTOR = D.ID
  group by D,ID, D.NAME) ;
select *
  from TEMP T
 where T.ANZAHL > (select MAX(COUNT(UNIQUE P.ID))
                     from DOCTOR D, PATIENT P
                    where D.NAME = 'Haid' and
                          P.ID_DOCTOR = D.ID);
```

5.4 Superlatives

Most of the problems encountered with comparatives also occur with superlatives. They are dealt with in an analogous way. An interesting phenomenon which has no direct parallel in comparative structures is shown in *(22)*:

(22) Welcher Arzt, der in der Unfallambulanz arbeitet, verdient am meisten?
(Which doctor, who works in the casualty department, earns the most?)

There are at least two interpretations of utterance *(22)*, i.e:

(23) a) Who has the highest salary among the doctors working in the casualty department?

b) Who has the highest salary of all persons and is, by the way, a doctor working in the casualty department?

Although interpretation *(23b)* can easily be derived by DATENBANK-DIALOG using the same formalism as for comparatives (copying the dominating relation) as in[7]

[6] for the database model given in *(18c)*
[7] assuming an extended version of database model *(18b)*

(24)
```
select P.ID, P.NAME
  from PERSON P
 where P.DEPT = 'CASUALTY' and
       P.STATUS = 'DR' and
       P.SALARY = (select MAX(P1.SALARY)
                     from PERSON P1);
```

in most circumstances *(23a)* is the most plausible interpretation and should be preferred. To produce this reading, another 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. This results in[8]

(25)
```
select P.ID, P.NAME
  from PERSON P
 where P.DEPT = 'CASUALTY' and
       P.STATUS = 'DR' and
       P.SALARY = (select MAX(P1.SALARY)
                     from PERSON P1
                    where P1.DEPT = 'CASUALTY' and
                          P1.STATUS = 'DR');
```

This copying in DATENBANK-DIALOG works on the caseframe representation, and thus is able to handle restrictions resulting from different syntactic constructions, such as:

(26) Welcher x verdient am meisten?
where restrictions on x may result from

a) the lexicon: *Unfallambulanzarzt*
"casualty department doctor"

b) adjective phrases: *in der Unfallambulanz arbeitende Arzt*
"in the casualty department working doctor"

c) prepositional phrases: *Arzt aus der Unfallambulanz*
"doctor from the casualty department"

d) noun phrases: *Arzt der Unfallambulanz*
"doctor of the *(gen)* casualty department"

e) relative clauses: *Arzt, der in der Unfallambulanz arbeitet*
"doctor, who works in the casualty department"

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.

[8]Note that restrictions on the selected `PERSON P` (in the first part of the query) must not be omitted, since persons from other departments or with other `STATUS` may accidentally have the same salary!

5.5 Dimensions and Units

A correct comparison is only possible if the values compared 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, 20.000,–,...) possibly combined with a unit of measure (e.g. 0,3m, 30cm, 12,34in, $20, ...)
- In the NL expression, when the user specifies a dimension and unit of measure verbatim (e.g. "10 Meter", "vor 3 Jahren" ...)
- In the encoding of the database, where comparable columns (e.g. `DOCTOR.SALARY` and `NURSE.SALARY`) may have different associated units of measure.

DATENBANK-DIALOG solves this problem by defining a normalized form with a value associated with a unit and associated transformation rules between measures of different units. The transformation rules operate on different levels:

- at the *scanner* level: Patterns are defined that transform different formats of numbers to the corresponding numeric values and normalize abbreviations of units. At this level "compound" numbers like dates are also normalized (see above).
- at the *parser* level: linguistic information is used to fill the slots in the normalized value frame.
- at the *interpretation* level: associated procedures are used to transform constant values from one unit to another.
- at the *database* level: transformation functions of the query language are used to perform conversions on variable data.

This enables DATENBANK-DIALOG to give the correct interpretation for

(27) *Welche Krankenschwester verdient genau so viel wie Dr. Haid ?*
Which nurse has the same salary as Dr. Haid?

with the (admittedly pathological) database encoding mentioned above, assuming that `DOCTOR.SALARY` is stored in dollars and `NURSE.SALARY` in cents.

(28)
```
select N.ID, N.NAME
  from NURSE N
 where (N.SALARY / 100) in (select D.SALARY
                              from DOCTOR D
                             where D.NAME = 'Haid');
```

6 Summary

Two important desiderata for natural language interfaces are portability and habitability. We have shown how these aspects have been accounted for in DATENBANK-DIALOG, a German language database interface. Portability is ensured for DATENBANK-DIALOG at a number of levels that are enumerated in detail and it is shown how they have influenced the overall design of the system. The consequences of postulating habitability are discussed and exemplified by a study of the treatment of comparison and measures that is tackled in DATENBANK-DIALOG in a way that

- gives a uniform interpretation to user queries of different syntactic and morphological appearance (equivalent, but syntactically different queries get the same semantic representation)
- enables users to enter data in the format and unit most convenient to them (formatting, unit conversion)
- removes 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)
- makes ambiguities explicit and incorporates presuppositions (relation and restriction copying).

DATENBANK-DIALOG is fully implemented. It is accessible world-wide via email coupled to a database containing information about all Austrian AI-projects.

Acknowledgments

Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry for Science and Research. We would like to thank Professor Robert Trappl for his continuing support and Elizabeth Garner proofreading and stylistic improvements.

References

Barwise, J. and R. Cooper (1981) 'Generalized Quantifiers and Natural Language', *Linguistics and Philosophy* **4**, 159-219

Haider, H. (1985) 'Verb Second in German', in H. Haider and M. Prinzhorn (eds.) *Verb Second Phenomena in Germanic Languages*, Foris, Dordrecht

Heinz, W. and J. Matiasek (1989) 'Die Anwendung Generalisierter Quantoren in einem natürlichsprachigen Datenbank-Interface', in J. Retti and K. Leidlmair (eds.) *5. Österreichische Artificial Intelligence Tagung*, Springer, Berlin

Hendler, J.A. and P.R. Michaelis (1983) 'The Effects of Limited Grammar on Interactive Natural Language', *Proceedings of CHI'83: Human Factors in Comput. Systems*, 190-192, ACM, New York

Keenan, E.L. and J. Stavi (1986) 'A Semantic Characterization of Natural Language Determiners', *Linguistics and Philosophy* **9**, 253-326

Krause, J. (1982) *Mensch-Maschine-Interaktion in natürlicher Sprache*, Niemeyer, Tübingen

Tennant, H.R. (1980) 'Evaluation of Natural Language Processors', University of Illinois, Report T-103

Trappl, R., J. Matiasek and G. Helscher (1991) 'Artificial Intelligence-Forschung in Österreich', *Künstliche Intelligenz* 2/91, 78-82

Trost, H., E. Buchberger and W. Heinz (1988) 'On the Interaction of Syntax and Semantics in a Syntactically Guided Caseframe Parser', in *Proceedings of the 12th COLING*, Budapest, 677-682