*Human–Computer Interaction — INTERACT'99*
*Angela Sasse and Chris Johnson (Editors)*
*Published by IOS Press, © IFIP TC.13, 1999*

# Register-domain Separation as a Methodology for Development of Natural Language Interfaces to Databases

**Serge Sharoff & Vlad Zhigalov**

Russian Research Institute for Artificial Intelligence, PO Box 111,103001, Moscow, Russia.

{sharoff,zhigalov}@aha.ru

**Abstract:** Wider application of interfaces to access databases in natural language is hindered by the problem of portability, i.e. customization of a general-purpose language-processing component to a particular database. In this paper we propose a methodology which is based on a separation of the domain model of a database from the register of database queries, i.e. a system of meanings employed by natural language for making queries. The paper describes basic concepts of the domain model, the procedure required for tuning the parsing engine to a database and the semantic-oriented approach for parsing queries. The described methodology is implemented as InBase, a system which allows easy and efficient development of interfaces to arbitrary SQL databases.

**Keywords:** natural language, databases, portability, domain model, parsing, vocabulary content.

## 1 Introduction

A recent introduction to the field of natural language interfaces to databases (NLIDBs) states that it is no longer a fashionable topic of academic research (Androutsopoulos et al., 1995). However, this still does not undermine the practical importance of this task, which is aimed at extraction of data from a database using a natural language (NL) as the universal medium for human communication instead of learning a formal query language with its artificial and rigid syntax. The second thing to be learned by a user is the data model of a database: a user usually knows a database problem domain, however, arrangement of this knowledge in tables, attributes and values of a corresponding (typically relational) database can be completely strange. These two types of problems are presented by a simple query in English *Average salary of managers in support services?* which corresponds to an SQL statement:

```
select AVG(Personnel.Salary)
from Personnel, Departments
where (Personnel.DeptID =
Departments.DeptID)
and (Departments.Name = ”Service”)
and (Personnel.Category ¡ 2)
```

Thus, structures of natural language expressions are mapped into SQL statements keeping the following conventions:

- Keeping the formal syntax of SQL, for example, the averaging operation is expressed as AVG(Personnel.Salary).
- Department names are not stored in the same table as information on personnel (so the join operation between two tables is necessary).
- The department of support services is stored under the title 'Service'.
- Categories of personnel are encoded by integers; in particular, managers are denoted by numbers 0 and 1.

All these conventions are hard to be kept in mind for a computer-illiterate user. Often, command-line interfaces, which input is based on a formal language, are replaced by graphical user interfaces (GUIs). This achieves both simplification of formal syntax (the user chooses a respective label, like AVG, from a list of options) and simplification of data presentation (data distributed over several tables are combined in a single screen object). However, GUIs require skills in interaction, thus constituting an interface language, which sometimes is more tedious to use than a formal language proper (in MS Access, for example, formulation of the expression AVG(Personnel.Salary) requires a sequence of 15 mouse clicks and very deliberate choices from appeared options). On the other hand, expression of queries in user-oriented representations, like Seeheim model applications (Olsen, Jr, 1992), (Wegener, 1995), is hindered by contextual dependency of what user treats as objects and their slots. For example, a country may be treated as an object in such queries as *Which countries supply products costing above twenty pounds?*, while the domain model treats Country as an attribute of Supplier, which is an attribute of the Product. In contrast, the system of meanings encoded in natural language with its relations and induced metaphors is by far the most natural system of meanings for a naïve user.

By such reasons development of NLIDBs has been one of the most popular directions of computational linguistics research since late sixties, scored close to machine translation. However, modern reviews of the state of the art, for example, (Androutsopoulos et al., 1995), (Copestake & Sparck Jones, 1990) admit that NLIDBs are not wide-spread and existing systems for development of them are far from commercial applications. Obstacles for wider application of NLIDBs are partly related to the principal incompleteness of existing technologies for parsing queries (this is also related to the problem of user's false negative and positive expectations about queries acceptable by the system). However, portability issues are of the same importance, relating to amount of efforts required for tuning a general analyser to a particular database.

The technology described in this paper mostly addresses the latter problem; it allows semi-automatic creation of a simple interface to user's database. In several person-days or weeks this interface can be improved to handle complex queries.

The basic design principles of InBase are discussed in Section 2; Section 3 is devoted to the Domain Model and its relation to the Database Model; Section 4 describes operations required for customization of InBase for a new database. Section 5 gives an overview of a parsing engine used for analysis of NL queries. Throughout this paper, examples refer to a database for the personnel of a sample company. The database consists of two tables: Personnel and Departments, first of which consists of attributes of Name, BirthDate, HireDate, Post, Category, DeptId, Sex, ChildNum, Marriage, Salary, Telephone. The second table consists of attributes of DeptId, Name, Chief, Operations.

## 2 Design Principles of InBase

The need for a separation of the core processing engine from the database-specific component is widely accepted in design of modern NLIDBs (Alshawi, 1992). However, the core engine mostly deals with syntactic analysis, which results are hard to map into database query statements, thus increasing the amount of efforts for customization. In contrast, the core parsing engine of InBase operates in terms of functions (language-oriented meanings) and follows the notion of register, which, according to Halliday (1978), describes a functional variation engendered by language use in a problem domain, cf. an analysis of the register of mathematics in Ch.11 of (Halliday, 1978). The register of the database queries differs from a problem domain, which is represented by a database: the register is a semiotic system of functions and means, which express these functions. Functions of the register of database queries include, for example, *Attribute, Value, Aggregate-Value*, etc.; configurations of functions are *Predicate, Interval, Class-Instance*, etc. The notion of register in InBase provides possibility to reuse semantic knowledge expressed in linguistic structures across different domains

Basically, tuning of the parsing engine of InBase to a user's database consists in development of a vocabulary (which domain-dependent part is extracted from the database automatically), mapping it onto domain model resources, including functional classes, which are included in the system, and testing for a proper mapping of register-based functions into database statements (the procedure is described in greater detail in Section 4).

Methods for analysis of user's queries also follows the classification of functions using, in the first place, semantic and pragmatic components of the communication as well as its context, instead of complete syntactic parsing of a query. The background of this original semantic-oriented approach (Narin'yani, 1980) is close to such approaches as Wilks' preferential semantics (Wilks, 1968), which was also applied to analysis of database queries by Boguraev & Sparck Jones (1981). So, instead of looking into syntactic constructions and lexical meanings (thus slipping into notorious problems like attachment of prepositional phrases or meanings of nominal compounds), the parsing engine determines "What function can be denoted by this particular lexical item?", "Which configurations of functions are possible?" and "What are linguistic constructions for their expression?". Another advantage of using functions is a greater congruence of their system across languages which are even not closely related typologically (the work reported in this paper has been done primarily for Russian and English with experiments for French, German, Georgian, and Czech, which happened to have relatively minor differences in their registers of database queries).

Analysis of a query results in a partial specification of an object requested (cf. Figure 1). Conceptually this partial specification is represented as a typed-feature structure constraining a set of objects that conform to this query. Structures of the intermediate representation language (Q) for partial specifications correspond to user's expectations for meanings encoded in the database.

From the user's viewpoint, a retrieved set of objects corresponding to this specification is either presented as a list (using forms corresponding to the class of retrieved objects and some conditions of the query) or mapped into a value (when an aggregate value, like *average* or *how many* is requested). A value returned from an aggregate value can be used in embedded queries, as in the query shown in Figure 1.

The Q-level corresponds to notions at the level of problem domain and is independent from the internal organization of a database. For example, it is natural for the database to store the birth date, as a constant value instead of person's age, while in the NL-query it is more natural to refer to this date from the current moment perspective (*Who is older than 40?* instead of *Who was born earlier than 01/01/59?*). A partial specification in Q-language is translated into an SQL query using joint operations for virtual attributes and substitution rules for such values as 'engineers' (which is a set of different positions) and '40 years' using information from the domain model of a database.

## 3 Domain Model

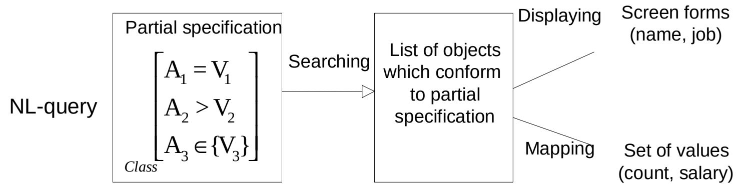
The core structure of a domain model (DM) in InBase is based on entity-relationship diagrams (ERDs). A DM consists of elements (including classes, attributes, values of attributes), and relations between them. Structural relations (the inheritance relation between classes, the slot relation between a class and its attributes) define a set of types of user's DM; they assemble entities in terms of ERD. Other relations may be defined between entities; they are whole-part and class-instance. The latter relation is of particular importance in relational databases, since often one table defines types (for example, goods) and another one — instances (for example, sold items). This distinction and its implications for the content of a formal query are rarely understood by a naïve user (*Who deals with spare parts?* vs. *Who sold spare parts to John Smith?*). The methodology for DM development in InBase significantly differs from other DM methodologies, like (Prieto-Diaz & Arango, 1991) due to the static nature of DMs in InBase. Such DM encodes only meanings pertaining to user's database, so it lacks some elements typical for CHI DMs, like tasks and scripts for interaction. DMs in InBase are represented using Resource Description Framework, RDF (W3C, 1999). Though RDF was designed as a language to describe meta-data of Web documents, it captures all the DM semantics necessary for the InBase. The Figure 2 shows the ERD for our database and fragments of its RDF-description. Note that the class Employee inherits the class Person, which belongs to the library of InBase classes facilitating development of user's DMs. This class belongs to built-in classes, since data about persons and their standard attributes (like name, age, etc.) are often stored in databases and NL-queries employ specific constructions referring to them, so this knowledge facilitates understanding of such queries. Other classes in this library are locations, units of measurements, and so on.

Partially an ERD maps onto the relational schema of a database (database tables correspond to classes, and attributes to properties of classes). In many cases, however, some semantically significant features of the DM are not reflected in the database model, and even in its content. For example, instances of classes Employee and Chief are stored in a single table; some attributes of the class Employee are virtual; they represent relations to other objects (Department is stored in another table). The database model also defines no relation between values of Education, but in user's terms, types of education constitute a scale, so in order to answer such questions as *Who has no higher education?*, the education scale should be modelled in the DM. The RDF representation of the education scale is the following:

```
<ib:Class ID="Education"/>
<dm:Education ID="Primary"/>
<dm:Education ID="Secondary"/>
<dm:Education ID="Higher_Ed"/>
<dm:Higher_Ed ID="BA"/>
<dm:Higher_Ed ID="MS"/>
<dm:Higher_Ed ID="PhD"/>
<ib:Seq><dm:Primary/><dm:Secondary/>
<dm:Higher/></ib:Seq>
<ib:Seq><dm:BA/><dm:MS/><dm:PhD/>
</ib:Seq>
```

In SQL the condition:

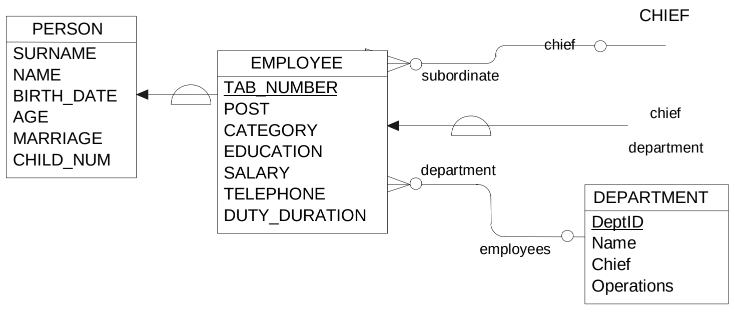
```
Employee[Education < higher_ed]
```



*Who has the maximum salary among engineers employed for more than 10 years.*

$$\underset{EMPLOYEE}{}\left[ \text{salary} = MAX \left( \text{salary} \leftarrow \underset{EMPLOYEE}{} \left[ \begin{matrix} \text{job}_{t}\text{itle} \supseteq \text{‘engineer}' \\ \text{duty}_{d}\text{uration} > 10\text{years} \end{matrix} \right] \right) \right]$$

**Figure 1:** The query processing scheme.



```
<!– EMPLOYEE class description –>
<ib:Class ID="EMPLOYEE">
  <ib:subClassOf subClassOf="PERSON"/>
  <ib:relationTo class="DEPARTMENT" type="one" label="department"/>
  <ib:relationFrom label="employees" resource="EMPLOYEE" type="many"/>
</ib:Class>
<!– AGE property description –>
<ib:Property ID="AGE">
  <ib:range resource="age"/>
  <ib:domain resource="PERSON"/>
</ib:Property>
```

**Figure 2:** The ERD for the sample DM and a part of its RDF description.

is represented using an 'or' clause:

(Education="primary" or Education="secondary")

## 4 Customization

Customization of a new NLIDB consists in:

- Development of the domain model.
- Stuffing the vocabulary.
- Debugging the newly created NLIDB.

The DM is created from the relational schema of the database semi-automatically; initially DM classes with their properties are created from database tables. Then the user can customize the DM by changing, adding and deleting classes and properties and binding them by relations (including virtual attributes, which are linked to other tables). This procedure is facilitated by inheriting classes from the library of predefined classes. Also the NLIDB designer specifies rules for presentation of objects retrieved by search. One more type of information to be specified upon customization is used for handling meta-questions about the DM, for example, *Which information is known about employees? What are the possible job titles?*

The content of the vocabulary of the constructed NLIDB consists of three parts. The first part includes the general-purpose lexicon, which forms the initial vocabulary content of any L-processor (*list, find, more than, not, except, average,* etc., approximately

600 words). The second part, which is also the largest one, is extracted automatically by scanning the database. Such words and phrases are added to the vocabulary as values with corresponding semantic descriptions referring to attributes of the DM classes (‘orientations’ of words). The third part of the lexicon is formed during the NLIDB tuning stage and consists of words specific for the database problem domain. This includes words and phrases referring to attributes (*salary, wage, date of issue*), synonyms to values taken from the database (for example, *woman* for *female* denoting sex in the database).

New words to the vocabulary are also added from the built-in vocabulary in the course of customization, when built-in classes and their attributes are added to user’s DM. For example, adding a class inheriting the class Person results in a prompt for adding specific lexical items referring to possible attributes of a human being (*who, name, born, sex,* etc.); presence of dates in the DM adds names of months and rules for assembling complex dates (*during November 25–29*), and so on.

Another type of information to be specified for the vocabulary of InBase is rarely addressed in NLIDBs, but is often encountered in applications. These are regular expressions (templates), which do not belong to a lexicon, but are meaningful in a problem domain. Examples of templates are standard identifiers (ISBN 0–465-05154–5), telephone numbers, which may be expressed in several ways: 255 4530, 255-4530, or 2554530, etc. Regular expressions for templates and their orientations are defined automatically by scanning the database. Additional templates and rules for their conversion to a standard representation stored in user’s database are added upon customization.

Reliability of the created NLIDB and completeness of its lexicon are evaluated by the designer issuing test queries, which may be collected in a special list to save user’s time for typing typical queries. A screen shot of InBase processing a query is shown in Figure 3.

## 5 Analysis of Queries

The parsing engine of InBase operates in terms of the register of database queries using entities, their attributes, values, and relations between entities as defined in the DM. In itself the parsing engine is implemented in a special language for development of L-processors, SNOOP (Sharoff, 1993), which integrates object-oriented approach with network representations and a production rule control mechanism, so entities of the domain model in SNOOP are represented as nodes of a semantic network.

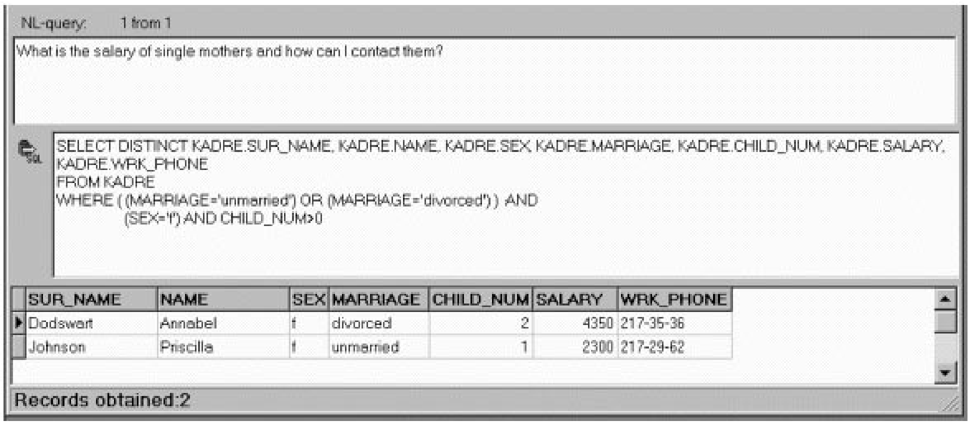
According to the semantic-oriented approach (Narin’yani, 1980), lexical semantics of words and phrases at the surface level is represented by the ‘orientation’ referring to a number of DM concepts that could be related to these words, for example, date in this database has two orientations: birth and hire dates. Several types of semantic units that appear in queries (attributes, values, comparisons, etc.) are combined into semantic structures on the basis of their orientations (so a date in a predicate with a value larger than 01/01/1980 in this database receives the only orientation of a hire date). Also analysis uses information about semantic classes, for example, existential predicates (like *has children, without higher education*) are interpreted according to types of their arguments (respectively, a numerical attribute, which value is greater than zero, or a non-numerical value, which represents a sequence).

Morphological and syntactic analyses are locally involved in cases of structural and semantic ambiguity, for example, local syntactic analysis is used for the recovery of ellipsis, when comparative and coordinate constructions are processed. The local top-down syntactic procedure may be supported by the bottom-up analysis in the case of more than one candidate for the role of the noun phrase nucleus. Detection of syntactic relations (government and agreement) is useful for solving the problem of the influence region of the negation, degree/modality modifiers, and certain locative units in elliptic constructions. Application of this technology to understanding of short texts and representing their domain model is described in (Kononenko & Sharoff, 1996).

## 6 Conclusion

The principles of the semantic-oriented analysis used in InBase have been developed by the end of 70s (Narin’yani, 1980), when several NLIDBs were developed manually. These experiments have led to InterBASE, a prototype system for development of NLIDBs to dBase-type databases in Russian and English (Narin’yani, 1991), (Trapeznikov et al., 1993). The system presented in this paper has extended capabilities of InterBASE by achieving a proper separation between the register and the domain model. Now the system operates with arbitrary SQL-access databases using the Borland Database Engine in Microsoft Windows. Currently the parsing engine of InBase has some evident limitations:

- A static world model, it has no support of databases designed to store information about changes (Tansel et al., 1993), temporal



NL-query: 1 from 1

What is the salary of single mothers and how can I contact them?

```
SELECT DISTINCT KADRE.SUR_NAME, KADRE.NAME, KADRE.SEX, KADRE.MARRIAGE, KADRE.CHILD_NUM, KADRE.SALARY,
KADRE.WRK_PHONE
FROM KADRE
WHERE ( (MARRIAGE='unmarried') OR (MARRIAGE='divorced') ) AND
        (SEX='f') AND CHILD_NUM>0
```

| SUR_NAME | NAME | SEX | MARRIAGE | CHILD_NUM | SALARY | WRK_PHONE |
|---|---|---|---|---|---|---|
| Dodswart | Annabel | f | divorced | 2 | 4350 | 217-35-36 |
| Johnson | Priscilla | f | unmarried | 1 | 2300 | 217-29-62 |

Records obtained:2

**Figure 3:** Data access in natural language.

databases require extensions in domain and linguistic models.

- No dialogue handling, the system answers to queries separately, being not able to resolve anaphoric links (like *Has he a PhD?* following the question in Figure 1).
- No handling of complex quantification (in such queries as *In which departments every employee has higher education?*).
- Results are presented in tables or screen forms (no NL-response generation component).

These limitations are topics for further developments. An advantage of the proposed technology is that it allows rapid development of NLIDBS by an inexperienced user and robustness of analysis. As Palmer & Finin (1990) note, evaluation of an NLP system is based on complex criteria, primarily involving its success in accomplishing user's goals. Despite of evident gaps in parsing, InBase addresses much of the real questions arising in interaction with databases. Experiments with several databases show that our interfaces analyse more than 85% of queries. Even in the case of false understanding the final SQL representation of a query provides a feedback which helps in detection of an error in analysis, so that either the source NL query or its SQL representation can be corrected to improve results of analysis. Alterations of the DM and the vocabulary of the NLIDB provide a way to reduce this error for further queries. The crucial questions for application of computational linguistics to HCI are: "For what and under what conditions is NL access effective for interaction of the end user with computer?". NLIDBs enable the end user with easy and efficient access to data stored in a database without mastery in artificial query languages and knowledge of precise relations between its tables. At present, our abilities in modelling NL features used for information delivery are inadequate for complete and correct understanding of all user queries. By this reason, the methodology adopted in InBase is based on customization of the general-purpose system of functions to the DM of a particular database.

One of the most promising research directions in NLIDBs is an interface to voice processing. The semantic-oriented approach described in this paper facilitates voice recognition, since in speech a syntactic norm is often violated, morphological information contained in flexions is missed during recognition, semantically insignificant words are not stressed, so they are often recognized with errors. Because the DM is represented using object notions, another research direction for InBase consists in development of interfaces to object-oriented DBMSs; this also helps in mapping the intermediate representation language to a database query language.

Access to WWW using NL queries as a prospective extension of this technology would dramatically increase precision and recall ratios in comparison to keyword-search facilities offered by the modern search engines. NL interface for a restricted problem domain can analyse a query taking into account its topic, such linguistic relations as synonymy, part-whole, generic-specific, and so on, in order to search documents with RDF-meta-data conforming to the content of the query.

## References

Alshawi, H. (1992), *The Core Language Engine*, MIT Press.

Androutsopoulos, I., Ritchie, G. & Thanisch, P. (1995), “Natural Language Interfaces to Databases: An Introduction”, *Natural Language Engineering* **1**(1), 29–81.

Boguraev, B. & Sparck Jones, K. (1981), A General Semantic Analyzer for Database Access, *in* **EDITOR*** (ed.), *Proceedings of the Seventh International Joint Conference on Artificial Intelligence (IJCAI’81)*, p.***PAGES***.

Copestake, A. & Sparck Jones, K. (1990), “Natural Language Interfaces to Databases”, *Knowledge Engineering Review* **5**(4), 225–49.

Halliday, M. (1978), *Language as a Social Semiotic: The Social Interpretation of Language and Meaning*, Edward Arnold.

Kononenko, I. & Sharoff, S. (1996), Understanding Short Texts with Integration of Knowledge Representation Methods, *in* D. Bjorner, M. Broy & I. Pottosin (eds.), *Perspectives of System Informatics*, Vol. 1181 of *Lecture Notes in Computer Science*, Springer-Verlag, pp.111–121.

Narin’yani, A. (1980), Interaction with a Limited Object Domain — ZAPSIB Project, *in* ***EDITOR*** (ed.), *Proceedings of COLING-1980*, ***PUBLISHER***, p.***PAGES***.

Narin’yani, A. (1991), “Intelligent Software Technology for the New Decade”, *Communications of the ACM* **34**(6), 60–7.

Olsen, Jr, D. R. (1992), *User Interface Management Systems: Models and Algorithms*, Morgan-Kaufmann.

Palmer, M. & Finin, T. (1990), “Workshop on the Evaluation of Natural Language Processing Systems”, *Computational Linguistics* **16**(3), 175–81.

Prieto-Diaz, R. & Arango, G. (1991), “Domain Analysis and Software Systems Modeling”, *IEEE Computer Society Press* *****VOLUME*****(***NUMBER***), ***PAGES***.

Sharoff, S. (1993), SNOOP: A System for Development of Linguistic Processors, *in Proceedings of the East–West Conference on Artificial Intelligence*, pp.184–8.

Tansel, A., Clifford, J., Gadia, S., Jajodia, S., Segev, A. & Snodgrass, R. (1993), *Temporal Databases — Theory, Design, and Implementation*, Benjamin/Cummings (Addison–Wesley).

Trapeznikov, S., Dinenberg, F. & Kuchin, S. (1993), InterBase: A Natural Language Interface System for Popular Commercial DBMSs, *in Proeedings of the East–West Conference on Artificial Intelligence*, pp.189–93.

W3C (1999), Resource Description Framework (RDF) Model and Syntax Specification, Technical Report, W3C. http://www.w3.org/TR/PR-rdf-syntax.

Wegener, H. (1995), The Myth of the Separable Dialogue: Software Engineering vs. User Models, *in* K. Nordby, P. H. Helmersen, D. J. Gilmore & S. A. Arnessen (eds.), *Human–Computer Interaction — INTERACT’95: Proceedings of the Fifth IFIP Conference on Human–Computer Interaction*, Chapman & Hall, pp.169–72.

Wilks, Y. (1968), “On-line Semantic Analysis of English Texts”, *Machine Translation* **11**(3-4), 59–72.

*Human–Computer Interaction — INTERACT'99*
*Angela Sasse and Chris Johnson (Editors)*
*Published by IOS Press, © IFIP TC.13, 1999*

# Author Index

*Human–Computer Interaction — INTERACT’99*
*Angela Sasse and Chris Johnson (Editors)*
*Published by IOS Press, © IFIP TC.13, 1999*

# Keyword Index