Lassy - A System for Analyzing Grammatical Dependencies in Latin

H. Gärtner, M. Gogolla
Universität Bremen
FB 3 Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme
Postfach 330 440
D-28334 Bremen
e-mail: {gaertner|gogolla}@informatik.uni-bremen.de

Abstract

We introduce the Latin analysis system Lassy, a slim system for analyzing Latin sentences. Tools for interacting with a database for morphological features are integrated as well as an interface for defining and applying a grammar database. We also give a small outline of the grammar model used in Lassy which is a pragmatic integration of linguistic theories known from both dependency and phrase structure grammar.

1 Introduction

There is a magnitude of approaches in the field of computer aided natural language processing and machine translation. In the context of databases there are also numerous connections to natural language processing [Sow88]. Most prominent, natural language interfaces seem to be one of the future front-end paradigms [SNM+88]. In general, two main lines can be identified with respect to natural language processing in databases, namely formulation of database schemas and database queries. Both lines have been studied in the direction starting from natural language yielding formal language and also vice versa.

Translation of natural language to queries: The best studied field is probably natural language interfaces for querying, see for example [SM92a, ART94] for an introduction and overview. To mention just some efforts, the spectrum of proposals ranges from questions of portability [Kap84], customization [Dam85], human factors [Shn81], and especial acceptance by database users [Kra80] to approaches for solving natural language ambiguities [JR91] and handling standardized query languages [Ott92].

Translation of natural language to schemas: In the context of information system design, analysis of natural language descriptions with respect to linguistics [BvdR95] and database schema validation [WF95], verification [vdVGvdR95] and integration [MML94] and treatment of semantic data models [TB94, BCD+95] have been studied. Other lines of research emphasize requirements engineering [RP92], the construction [DO92] of conceptual schemas or specialized databases [JME95].

Translation of queries to natural language: The opposite direction to the approaches above concerns generation of natural language starting from a formal artificial language. One of the first proposals in this line was translation of relational algebra [LdR86] continuing with a graphical language [BF92] and SQL [AK92], to mention only some relevant work.

Translation of schemas to natural language: A topic which has not been studied in depth up to now is natural language generation [Dal92, Dal95] of a formally specified conceptual model. Such questions are however most relevant in the context of information system design environments when formal descriptions are to be discussed with possible later database users.

Our approach does not directly follow one of the cited lines but is designed to provide a comfortable database-supported interface for analyzing Latin sentences. With respect to Latin, analysis of languages with 'free' word-order has been investigated before [Cov90], and Latin has served as an example in that context [Koc93].

Nevertheless, most proposals for natural language translation are often unification, i.e., rule-based and thus work only for a subset of natural language. The philosophy of our Latin analysis system Lassy is to provide a database supported tool which does less on its own but can cope due to interactivity and extensibility with any Latin sentence. Morphological as well as structural information is kept in a database and is accessed by a graphical user interface using Tcl/Tk [Ous94]. Lassy is a slim tool for investigating 'real' Latin sentences. It does therefore not stand directly in the tradition of Intelligent Tutoring Systems (ITS) (see for example [SM92b]) or Intelligent Language Tutoring Systems (ILTS) because it does not contain the necessary complex components for an ILTS according to [Sch90]. We see our system as a basic system which might give an inspiration for future systems to deal with a language as a whole. In particular, we will use derived versions of it in the context of information system design based on extended Entity-Relationship [GH91] and Object-Oriented [GHC+94] approaches.

The structure of our paper is as follows. In Sect. 2 we present a Latin example sentence which will be used throughout to demonstrate Lassy's capabilities. Section 3 explains the main features of Lassy with respect to sentence analysis. The paper is closed with Sect. 4 giving some concluding remarks.

2 An example sentence

The basic idea of the system is to support the advanced Latin learner in analyzing the structure of a given sentence. Neither will it parse the text nor will it apply language specific grammar rules or generate a translation for the user. To explain the different parts of the system, we will first regard a small example (taken from Caesar, B.G. VII. 84, 5 [Cae52]) provided with a suggestion for a translation.

> omnia enim plerumque, quae absunt, vehementius hominum mentes perturbant.

> *Indeed, all [things] that are absent commonly disturb the minds of men more forcefully.*

There are two verbs: *perturbant* which governs the main clause and *absunt* which governs a relative clause. The main verb has four dependent parts:

1. A subject *omnia* which is augmented with a relative clause *quae absunt*: *all things that are absent.* In this sub-clause the verb *absunt* again has a dependent subject *quae.*

2. An object *mentes* which is augmented with an genitive attribute *hominum*: *the minds of men.*

3. An adverb *plerumque*: *commonly.*

4. Another adverb *vehementius*: *more forcefully.*

Thus, if we use an ad hoc tree representation for the description of the syntactic structure, it could look like the one depicted in Fig. 1. Lines without arrows denote dependency while arrows suggest semantic connection. Thus 'enim' connects our example sentence with (hypothetical) preceding text.

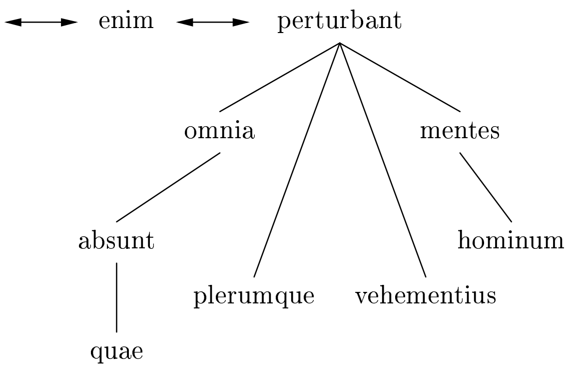

Figure 1: Dependency-like analysis of the example sentence

The description given above is only one way to explain the syntactic structure of our example. There are various ways to describe such a structure ranging from traditional methods [KnS66] to newer approaches like [Pin88] which are based on concepts of dependency. For a bibliography on this subject see for example [Hay65] or [ST76], and for phrase structure grammars we refer to [GP82]. A general dependency Grammar for Latin has been investigated by [Hap76].

3 Phases in sentence analysis

The above analysis might make the beginner wonder, how one can identify the described relationships among the words and hence figure out the syntactic structure of the sentence.

3.1 Morphological analysis

One can divide the different natural languages into classes according to their properties. One of these properties important in our context is whether a language has inflection or not. Inflection means that there exists a set of roots which is used to derive the different forms of a word. In *you walk* and *he walks*, for example, *walk*

	Singular	Plural
Nominative	mens	mentes
Genitive	mentis	mentum
Dative	menti	mentibus
Accusative	mentem	mentes
Ablative	mente	mentibus

Figure 2: A traditional grammar table

and *walks* derive from the same root *walk*. Figure 2 shows a Latin example.

Such tables are well-known from Latin grammars. The catchword, i.e., the form which appears in a dictionary - in our example *mens* - will also be called *lemma*. All the items in the table (*mens, mentis, menti, ...*) will be referred to as *forms*, while *Nominative, Genitive, ...* and *Singular, Plural* will be called *features*. When analyzing a sentence, the situation is different. Here a form gives references to features. The problem is, that this reference may be ambiguous, as the form *mentes* in our example shows because *mentes* may be interpreted as nominative or accusative. This is however only one kind of ambiguity. Another problem arises, when a form can be derived from two or more different catchwords.

3.1.1 Database support for morphological data

The table in Fig. 2 is only a clearly arranged way to connect a form with a set of features. For example, the form *menti* could be expressed as the following feature term:

(menti, {mens Singular Dative})

Thus we get a tuple, consisting of a form and a set of features, in this case the lemma, number and case. This is in fact nearly the way Lassy stores forms. Apart from abbreviations there are only a few differences:

1. A word-class like verb or substantive - separated by a colon - is stored with every lemma. In this case ':s' for substantive, i.e., noun.

2. Any number of additional features can be stored with the form. In this concrete case we will add *f* for feminine gender.

Hence, the tuple in our example will look like this:

(menti, { mens:s sg dat f })

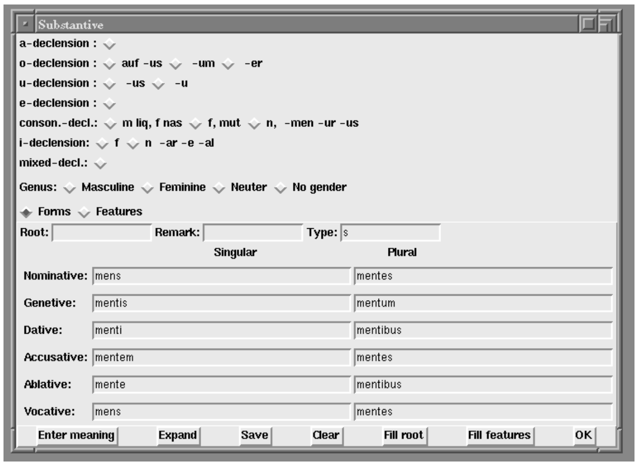

Figure 3: The interface for editing forms

Note that we do not use a fixed set of attributes for each word-class. Thus each form is connected with a simplified kind of feature term. In Lassy we do not even use explicit types for the feature terms but simple lists of strings. If you insist on having explicit types, you can augment your features with types:

(menti, {mens:s number:sg case:dat gender:f})

For a more elaborated feature based framework see for example [Wal90] or [HR91]. In Lassy there are windows for each word-class which can be regarded as Latin-specific interfaces for database access (and natural language processing). Each window can be used to generate sets of pairs with form and feature set. These system generated sets contain only the 'standard' features known from classical grammar - like *case*, *number*, *gender*, etc. - but they can easily be modified and extended by the user. Figure 3 shows how the interface for substantives looks like whereas Fig. 4 shows the corresponding features.

The interfaces provided by Lassy are a comfortable way of interacting with the underlying database. A Latin verb for example can have about 245 forms. In the regular case the task of adding a new verb to the database is handled by performing about five 'mouse-clicks', putting in the three roots of the verb and finally entering the meaning, as an ordinary dictionary would have it. Thus Lassy maintains the following structures in its database:

Forms:
Key: *form* Values: *list of feature terms*

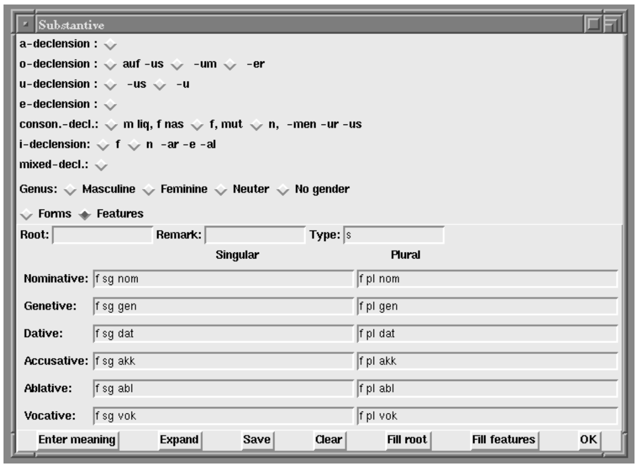

Figure 4: The interface for editing features

Dictionary:
Key: *lemma* Values: *article*

So far we have seen, how we can enter and manipulate forms and features in Lassy. In the next section we will demonstrate, how the system can be used to analyze the forms in a given sentence.

3.2 Morphological analysis in Lassy

As the simple example *mentes* of the preceding section shows, it is not always possible to determine a word in a sentence simply by its form. The question is which feature set applies to the current context. In our example we have to choose between (mentes, {mens:s pl nom f}) and (mentes, {mens:s pl acc f}). A quite natural approach to handle problems like this is to divide all possible feature sets in two sets:

1. The first set P will contain all possible forms - i.e., the forms applying to any context - after the analysis.

2. The second set C will contain after the analysis only one feature set, namely the one that fits the current context.

Initially all feature sets have to be regarded as candidates for the current context, because the system maintains no context knowledge. The task of the user is to find the feature set which fits the current context by successively excluding the feature sets which do not. Thus elements from C have to be moved to P until only one element remains in C. Figure 5 shows an example

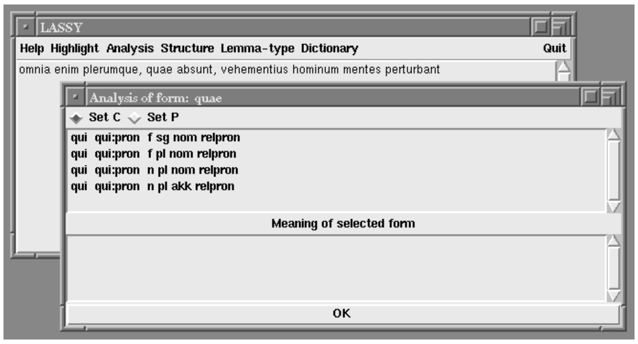

Figure 5: Analysis of a form

of a form analysis. There are four feature sets in C displayed in the foreground window. The user can switch between the sets C and P using the radio buttons and transfer elements by mouse-click.

In the process of identifying a particular form one has to take the congruences into account. Two forms are said to be congruent if they match in a specific feature for example in case, e.g., if they are both nominative. To visualize congruences one has to select the form which should be matched. In Fig. 6 the form with the features *f, sg, nom* and *relpron* has been selected and afterwards double-clicked. The double-click resulted in the appearance of the five check-buttons-buttons above the OK-button. These buttons correspond to the features of the selected form. Selecting such a button - in Fig. 6 the *f*-button - causes the system to match the selected feature with the features of the other words in the text. If a word agrees in that feature, this is depicted by underlining it. If multiple buttons are selected, the matching is done with the conjunction of the corresponding features. Lassy offers three ways of matching: Matching exclusively among set C, matching exclusively among set P and matching among both sets.

In this case the matching process can be used to exclude the selected form. As we know, the Latin relative pronoun has to be congruent with its antecedent in gender and number. In our example we found the word 'mentes' to match in gender. But if we try to match the feature 'sg' - singular, i.e., number - in conjunction with the feature 'f' we will fail to find a word in the sentence with matching features. Using this knowledge, the form with the features 'f sg nom relpron' can be excluded from set C.

With this and other methods (the system offers for example functions for highlighting possible predicates or subjects) one can successively identify the basic com-

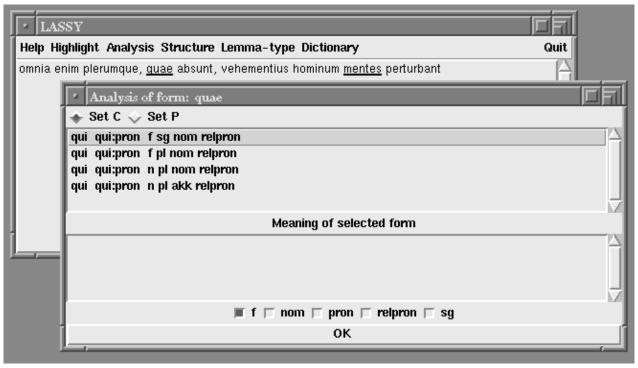

Figure 6: Matching features

ponents of the sentence.

Before we start to describe the visualization conventions in Lassy, a general remark has to be made concerning the different ways grammar structures can be presented: An important factor is on which level a sentence is analyzed. If we will use a sub-phrase level we can regard the whole sub-phrase of our example

'... hominum mentes ...'

as an object. If we like to point out the structure of that sub-phrase itself, we might call just 'mentes' the object and 'hominum' the dependent attribute.

The goal of Lassy was to provide a flexible way to handle these different levels. Lassy is very permissive in that point and shifts a lot of responsibility to the user. When, for example, analyzing a sentence in the first part on sub-phrase level and in the second part on word level one has to be aware of the linguistic problems, that arise from such an analysis.

3.3 Syntactic analysis

Up to now we have analyzed the text only on the basis of isolated words and their features. The task of the next analysis step will be to group the words to syntactic units and find out the relationships among them. These two tasks are of course interdependent: On the one hand, the syntactic structure is based on the congruences among the words, on the other hand one has to know which congruences can cause two or more words to form a unit.

To be able to represent the grammatic structure of a sentence, one has to use some kind of syntactic formalism. Our system comes along with an basic formalism, which combines elements from dependency-, phrase-structure and traditional grammar.

A grammar model

The grammar can be defined by the user in a tree-like schema. A concrete syntactic analysis is an instance of such a tree, which also contains groups of words in its nodes. We have chosen a tree-structure with basically two different kinds of nodes: Word-groups and functions. Functions are names for different (functional) parts of a sentence, e.g., subject or predicate, while word-groups denote classes of groups of words, e.g., noun or nominal phrase. The nodes can be connected with three different edges:

1. **Realization**: Each sentence part (i.e., each function in a sentence) can be realized by different classes of phrases (i.e., of groups of words). In

   'omnia ..., quae absunt, ...'

   the function *attribute* is realized by the nominal phrase (i.e., the relative clause) 'quae absunt'. In

   '... hominum mentes ...'

   the function *attribute* is realized by a simple noun 'hominum'. Realization is denoted by $\in$, where $X \in Y$ means X realizes Y. Thus we get 'hominum' $\in$ *attribute* and 'quae absunt' $\in$ *attribute*.

2. **Dependency**: The nodes can be connected via dependency edges which denote grammatical dependency. Dependency is denoted by Δ, where $X \ \Delta \ Y$ means X is dependent of Y. Thus we get for example 'hominum'$_{\text{noun}}$ Δ 'mentes'$_{\text{noun}}$ and 'quae absunt'$_{\text{NP}}$ Δ 'omnia'$_{\text{noun}}$. The second dependency relationship is not expressible in classical dependency grammar, because each node is only allowed to be connected with one word.

3. **Part-of**: The part-of relationship is needed for structures to which the dependency concept does not apply. This is a pragmatic solution which yields linguistic problems but appears to be sufficient for our purpose. The part-of-relationship is denoted by Π, where $X \ \Pi \ Y$ means X is part of Y. Thus we get 'perturbant'$_{\text{predicate}}$ Π *sentence*, 'enim'$_{\text{conjunction}}$ Π *sentence* in our example. Note that you are not forced to use this relation in the definition of your grammar. If you simply leave it out, use only one single word per node and define an order among the relations, you can use the system for visualizing pure dependency grammar.

sentence
Π *conjunction*
  enim
Π *verb* $\in$ *predicate*
  perturbant
    Δ *noun* $\in$ *subject*
      omnia
        Δ *relative clause* $\in$ *attribute*
          Π *pronoun*
            quae
          Π *verb* $\in$ *predicate*
            absunt
    Δ *noun* $\in$ *object*
      mentes
        Δ *noun* $\in$ *attribute*
          hominum
    Δ *adverb* $\in$ *adverbial qualification*
      plerumque
    Δ *adverb* $\in$ *adverbial qualification*
      vehementius

Figure 7: Analysis in indentation convention

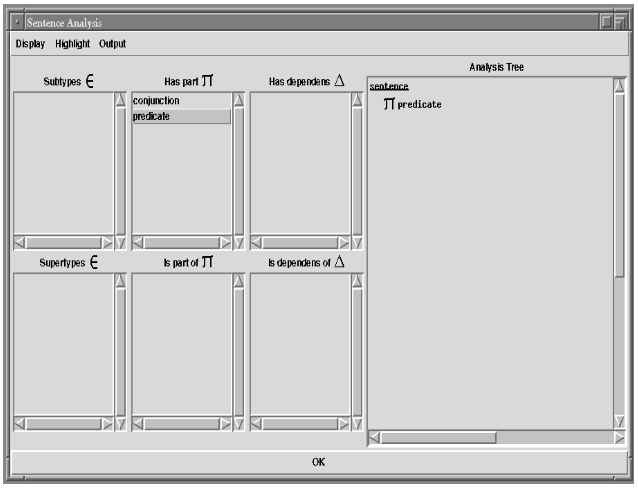

Figure 8: The interface for syntactic analysis

Visualization conventions

While dependency grammar or phrase-structure-like grammars are usually displayed with trees (stemmata), we chose a simpler way by representing the tree structure of the sentence in different levels of indentation. An example of this visualization is shown in Fig. 7.

Each node may be represented by two lines. The first line contains a Π or a Δ which determines the node's relationship to the parent node, i.e., the next preceding line on a lower level of indentation. The root node is denoted by *sentence*. The sentence structure is easy to survey, if we regard the vertical arrangement of the symbols Π and Δ.

The user interface for the syntactic level

The convention for visualizing syntactic structures described above is directly realized in the system with the only difference that we do not display the names of the nodes and the associated text at a time. In fact, LaTeX output as seen in Fig. 7 can be generated by the system. Figure 8 shows the interface, which may be used to build such an analysis.

The rightmost sub-window is used to display the analysis. The underlined node is the current node. The three different kinds of edges are displayed in the other six sub-windows. Note that the edges are directed, therefore each edge needs an extra sub-window for its reverse. A 'predicate' node has just been inserted using a 'part-of'-edge in the situation above.

Fig. 9 shows a realization of our analysis in Lassy. The left part shows the names of the nodes in the convention described above while in the right part the corresponding text is depicted. The text can be attached to the nodes by choosing a node in the window for syntactic analysis (see for example Fig. 8) and selecting the desired words by mouse-click in the main window (see for example the rear window in Fig. 5). Words which are already selected are marked with a stippled background. Thus the words remaining for analysis will be easier to survey.

Lassy offers also a another link between the original text and the derived syntactic analysis. When the user moves the mouse-cursor in the window for syntactic analysis over a node, the text either of this specific node or of the whole subtree is highlighted. This can be especially useful if the constituent is discontinuously distributed over the sentence. Figure 10 illustrates this feature with the subject constituent of our running example. In this case the mouse-cursor points to the word 'subject' in the analysis tree and the highlight mode is set to highlight whole subtrees.

Additional features

In general the user has to keep his own dictionary of forms. The current version contains forms derived from about some hundred catchwords with brief German translations. Lassy has however an interface to a Latin-English dictionary. Additionally, you can also load and

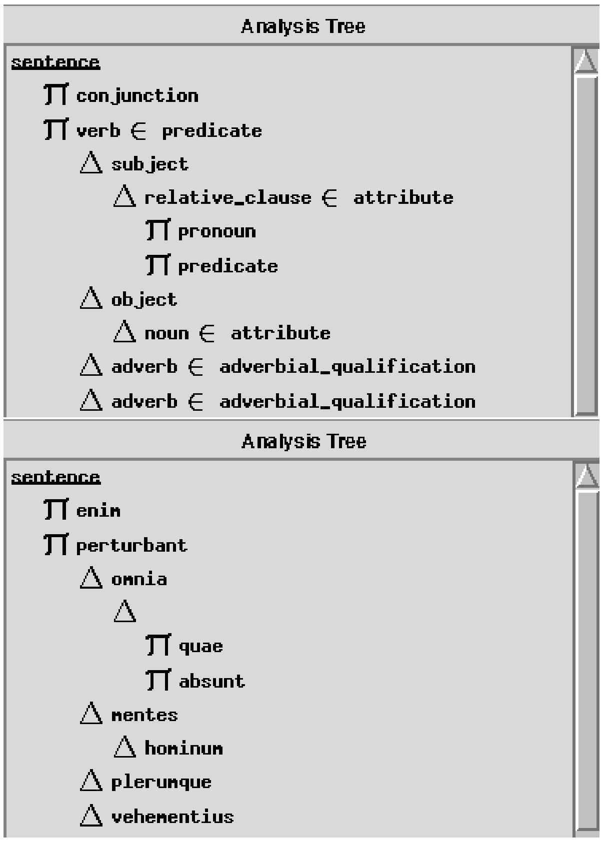

Figure 9: A syntactic analysis

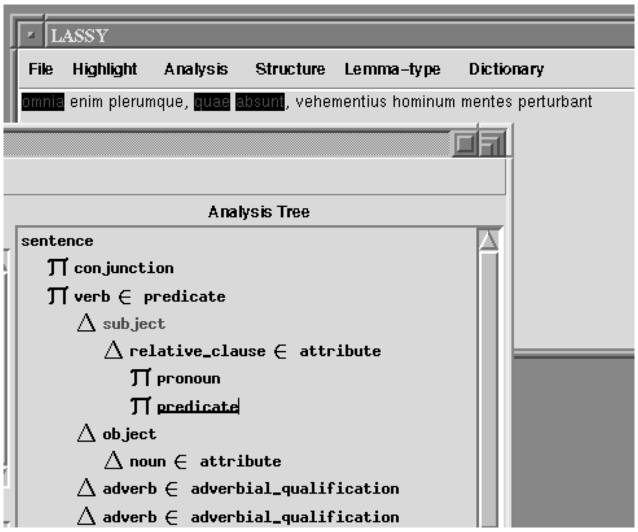

Figure 10: Highlighting subtrees

save your analysis and generate LaTeX-output of it, as mentioned above.

4 Conclusion

Lassy is a system, which will work fine for the advanced Latin learner and may be of help when analyzing a concrete sentence. It has however some limitations which keep it far from being a complete tutoring system. Due to its emphasis on flexibility and extensibility it lacks a complete linguistic theory as well as a complete description of the Latin language. This starts already at the morphological level: All forms of a word-class are treated in the same way, so that when trying to figure out a form the user is always bothered with solutions which seldom apply. What is even worse is, that when entering a new set of forms, the user often does not know whether the semi-automatic generated forms really exist or if they can be build only theoretically.

On the other hand it has turned out, that Lassy can cope with 'real world' problems and is especially helpful in the 'pattern matching' part of sentence analysis. This enables even the user with poor morphological knowledge to concentrate on syntactic aspects of sentence analysis. Of course, another question which can be rightfully asked is, if it is really a pedagogical benefit to have a complete form analysis at mouse-click.

Lassy is a basic system which might give an inspiration for future systems to deal with a language as a whole. The ideas implemented in Lassy seem to be general enough to be employed for other natural languages as well.

References

[AK92] H. Amano and Y. Kambayashi. Translation of SQL Queries Containing Nested Predicates into Pseudo-Natural Language. In A. Makinouchi, editor, *Proc. 2nd Int. Symp. Database Systems for Advanced Applications (DASFAA'91)*, pages 116–125. World Scientific, Singapore, 1992.

[ART94] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. Natural Language Interfaces to Databases - An Introduction. DAI Research Paper 709, Department of Artificial Intelligence, Edinburgh University, 1994.

[BCD+95] E. Buchholz, H. Cyriaks, A. Duesterhoeft, H. Mehlan, and B. Thalheim. Acquiring Complex Information from Natural Language for EER Database Design. In M. Bouzeghoub and E. Metais, editors, *Proc. 1st Int. Workshop on Applications of Natural Languages to Databases (NLDB'95)*. AFCET, Paris, 1995.

[BF92] G. Bono and P. Ficorelli. Natural Language Restatement of Queries Expressed in a Graphical Language. In G. Pernul and A.M. Tjoa, editors, *Proc. 11th Int. Conf. on Entity-Relationship Approach (ER'92)*, pages 357–374. Springer, LNCS 645, Berlin, 1992.

[BvdR95] J.F.M. Burg and R.P. van de Riet. The Impact of Linguistics on Conceptual Models - Consistency and Understandability. In M. Bouzeghoub and E. Metais, editors, *Proc. 1st Int. Workshop on Applications of Natural Languages to Databases (NLDB'95)*. AFCET, Paris, 1995.

[Cae52] G.I. Caesar. *Commentarii Belli Gallici - G.I. Caesaris Comentarii - Teil I*. Bibliotheca scriptorum Graecorum et Romanorum Teubneriana. Teubner, Leipzig, 1952.

[Cov90] M.A. Covington. A Dependency Parser for Variable-Word-Order Languages. Research Report AI-1990-01, University of Georgia, 1990.

[Dal92] H. Dalianis. A Method for Validating a Conceptual Model by Natural Language Discourse Generation. In C. Rolland, F. Bodart, and C. Cauvet, editors, *Proc. 5th Int. Conf. Advanved Information Systems Engineering (CAiSE'92)*, pages 425–444. Springer, LNCS 685, Berlin, 1992.

[Dal95] H. Dalianis. Aggregation, Formal Specification and Natural Language Generation. In M. Bouzeghoub and E. Metais, editors, *Proc. 1st Int. Workshop on Applications of Natural Languages to Databases (NLDB'95)*. AFCET, Paris, 1995.

[Dam85] F. Damerau. Problems and Some Solutions in Customization of Natural Language Database Front Ends. *ACM Transactions on Office Information Systems*, 3(2):165–184, 1985.

[DO92] L. Dunn and M.E. Orlowska. A Natural Language Interpreter for Construction of Conceptual Schemas. In C. Rolland, F. Bodart, and C. Cauvet, editors, *Proc. 5th Int. Conf. Advanved Information Systems Engineering (CAiSE'92)*, pages 175–194. Springer, LNCS 685, Berlin, 1992.

[GH91] M. Gogolla and U. Hohenstein. Towards a Semantic View of an Extended Entity-Relationship Model. *ACM Trans. on Database Systems*, 16(3):369–416, 1991.

[GHC+94] M. Gogolla, R. Herzig, S. Conrad, G. Denker, and N. Vlachantonis. Integrating the ER Approach in an OO Environment. In R. Elmasri, V. Kouramajian, and B. Thalheim, editors, *Proc. 12th Int. Conf. on Entity-Relationship Approach (ER'93)*, pages 376–389. Springer, LNCS 823, Berlin, 1994.

[GP82] G. Gazdar and K. Pullum. *Generalized Phrase Structure Grammar. A Theoretical Synopsis*. Indiana University Linguistics Club, 1982.

[Hap76] H. Happ. *Grundfragen einer Dependenz-Grammatik des Lateinischen*. Vandenhoek und Ruprecht, Göttingen, 1976.

[Hay65] D.G. Hays. *An Annotated Bibliography of Publications on Dependency Theory*. Santa Monica (CA), 1965.

[HR91] O. Herzog and C.-R. Rollinger, editors. *Text Understanding in LILOG. Integrating Computational Linguistics and Artificial Intelligence. Final Report on the IBM Germany LILOG-Project*. Springer, LNCS 546, Berlin, 1991.

[JME95] C. Jouis and W. Mustapha-Elhadi. Conceptual Modelling of Database Schema Using Linguistic Knowledge - Application to Terminological Databases. In M. Bouzeghoub and E. Metais, editors, *Proc. 1st Int. Workshop on Applications of Natural Languages to Databases (NLDB'95)*. AFCET, Paris, 1995.

[JR91] J.A. Johnson and R.S. Rosenberg. A Measure of Semantic Relatedness for Resolving Ambiguities in Natural Language Database Requests. *Data & Knowledge Engineering*, 7:210–225, 1991.

[Kap84] J.J. Kaplan. Designing a Portable Natural Language Database Query System. *ACM Transactions on Database Systems*, 9(1):1–19, 1984.

[KnS66] R. Kühner and C. Stegmann. *Ausführliche Grammatik der lateinischen Sprache. Zweiter Teil. Satzlehre. Mit Zusätzen und Berichtigungen von A. Thierfelder.* Hahnsche Buchhandlung, Hannover, 1966.

[Koc93] U. Koch. The Enhancement of a Dependency Parser for Latin. Research Report AI-1993-03, University of Georgia, 1993.

[Kra80] J. Krause. Natural Language Access to Information Systems - An Evaluation Study of its Acceptance by End Users. *Information Systems*, 5(4):297–318, 1980.

[LdR86] B.G.T. Lowden and A.N. de Roeck. The REMIT System for Paraphrasing Relational Query Expressions into Natural Language. In W. Chu, G. Gardarin, S. Ohsuga, and Y. Kambayashi, editors, *Proc. 12th Int. Conf. Very Large Databases (VLDB'86)*, pages 365–371. Morgan Kaufmann, San Mateo (CA), 1986.

[MML94] E. Metais, J.-N. Meunier, and G. Levreau. Database Schema Design: A Perspective From Natural Language Techniques to Validation and View Integration. In R. Elmasri, V. Kouramajian, and B. Thalheim, editors, *Proc. 12th Int. Conf. on Entity-Relationship Approach (ER'93)*, pages 190–205. Springer, LNCS 823, Berlin, 1994.

[Ott92] N. Ott. Aspects of the Automatic Generation of SQL Statements in the Natural Language Query Interface. *Information Systems*, 17(2):147–159, 1992.

[Ous94] J.K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, 1994.

[Pin88] H. Pinkster. *Lateinische Syntax und Semantik.* Francke, UTB 1462, Tübingen, 1988.

[RP92] C. Rolland and C. Proix. A Natural Language Approach for Requirements Engineering. In C. Rolland, F. Bodart, and C. Cauvet, editors, *Proc. 5th Int. Conf. Advanved Information Systems Engineering (CAiSE'92)*, pages 257–277. Springer, LNCS 685, Berlin, 1992.

[Sch90] C.B. Schwind. An Intelligent Language Tutoring System. *Int. Journal of Man-Machine Studies*, 33(5):557–579, 1990.

[Shn81] B. Shneiderman. A Note on Human Factors Issues of Natural Language Interaction with Database Systems. *Information Systems*, 6(2):125–129, 1981.

[SM92a] M.N.M. Sap and D.R. McGregor. Natural Language Interfaces to Databases - State of the Art. IKBS-7-92, University of Strathclyde, Department of Computer Science, 1992.

[SM92b] M.L. Swarz and Y. Masoud, editors. *Intelligent Tutoring Systems for Foreign Language Learning.* Springer, Berlin, 1992.

[SNM+88] S. Spaccapietra, Y. Nummenmaa, R. Melchert, M. Schrefl, and P. Velardi. Interfaces for Advanced Data Base Systems - Iconic, Graphical, Natural Language Oriented. In C. Battini, editor, *Proc. 7th Int. Conf. on Entity-Relationship Approach (ER'88)*, pages 25–46. North-Holland, Amsterdam, 1988.

[Sow88] J.F. Sowa. Knowledge Representation in Databases, Expert Systems, and Natural Language. In R.A. Meersman, Z. Shi, and C.-H. Kung, editors, *Proc. 3rd IFIP Working Conf. on Database Semantics (DS-3)*, pages 17–50. North-Holland, Amsterdam, 1988.

[ST76] H. Schumacher and N. Trautz. Bibliographie zur Valenz und Dependenz. In H. Schuhmacher, editor, *Untersuchungen zur Verbvalenz*, pages 317–343. Tübingen, 1976.

[TB94] A.M. Tjoa and L. Berger. Transformation of Requirement Specifications Expressed in Natural Language into an EER Model. In R. Elmasri, V. Kouramajian, and B. Thalheim, editors, *Proc. 12th Int. Conf. on Entity-Relationship Approach (ER'93)*, pages 206–217. Springer, LNCS 823, Berlin, 1994.

[vdVGvdR95] B. van der Vos, J.A. Gulla, and R.P. van de Riet. Verification of Conceptual Models based on Linguistic Knowledge. In M. Bouzeghoub and E. Metais, editors, *Proc. 1st Int. Workshop on Applications of Natural Languages to Databases (NLDB'95)*. AFCET, Paris, 1995.

[Wal90] B. Walter. Datenbankkonzepte für wissensbasierte Systeme. IWBS Report 151, Institut für wissensbasierte Systeme, Stuttgart, 1990.

[WF95] C. Wu and D.J. Flynn. NALABA - A Natural Language Based Integrity Requirements Validation System. In M. Bouzeghoub and E. Metais, editors, *Proc. 1st Int. Workshop on Applications of Natural Languages to Databases (NLDB'95)*. AFCET, Paris, 1995.