Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-94-38

DISCO—An HPSG-based NLP System and its Application for Appointment Scheduling

Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman

November 1994

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Sema Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct systems with technical knowledge and common sense which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- ☐ Intelligent Engineering Systems
- ☐ Intelligent User Interfaces
- ☐ Computer Linguistics
- ☐ Programming Systems
- ☐ Deduction and Multiagent Systems
- ☐ Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

DISCO—An HPSG-based NLP System and its Application for Appointment Scheduling

Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman

DFKI-RR-94-38

This paper has appeared in: Proceedings of the 15th International Conference on Computational Linguistics, August 5–9, 1994, Kyoto, Japan.

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITWM-9002 0).

ISSN 0946-008X

DISCO—An HPSG-based NLP System and its Application for Appointment Scheduling

Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman

German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany

⟨*name*⟩`@dfki.uni-sb.de`

Abstract

The natural language system DISCO is described. It combines

- a powerful and flexible grammar development system;
- linguistic competence for German including morphology, syntax and semantics;
- new methods for linguistic performance modelling on the basis of high-level competence grammars;
- new methods for modelling multi-agent dialogue competence;
- an interesting sample application for appointment scheduling and calendar management.

Contents

1 Introduction

We will describe results of a project in natural language research carried out during the last four years at the German Research Center for Artificial Intelligence in Saarbrücken. All system building took place during the last three years. The special approach of this project is a combination of linguistically sound high-level grammatical description and specialized methods for linguistic performance modelling.

During the last decade it has become obvious to the majority of researchers in our field that the linguistically designed elegant and transparent grammars written in high-level representation languages such as HPSG or LFG could not be employed for efficient and robust processing in a straightforward way.

Many researchers have therefore resorted to well-known older methods such as ATNs or other augmented finite-state methods, to statistical or connectionist methods, or to combinations of these. Several projects participating in the demanding ARPA competitions fall in this category.

Many others have decided to settle for a compromise between high-level description and efficient processing by strongly constraining their formalisms. The resulting formalisms are usually much closer to PROLOG and do not contain a powerful multiple-inheritance type system; e.g. the Core Language Engine (CLE) of SRI Cambridge [1], its derivative the GEMINI system of SRI International in Menlo Park, the LKP of SIEMENS in Munich [3]. As a consequence of their design philosophy, these systems usually do not feature a powerful development platform.

Only a few groups have continued to work in high-level formalisms driven by the expectation that better processing methods for these formalisms can be developed. Our work belongs in this category. However, we have decided to allow for combinations of high-level grammatical description and low-level processing methods by strictly distinguishing between a general linguistic competence model and very application-specific performance models.

Our decision was based on some fundamental criteria:

- It is necessary to build general, reusable competence systems that can be the basis for different applications, since the development of linguistic competence is too costly to redo for each new application.
- In the foreseeable future, NL applications will have very limited linguistic competence, where the limitations depend on the task to be performed. They determine coverage and depth of analysis.
- The general competence system as such will not be used for individual applications because each application type imposes specific requirements on the performance model. Depending on the task there are quite different constraints on robustness, accuracy, and processing speed.

On the basis of these assumptions, we took a rather uncompromising stand. We decided to utilize the most suitable and most advanced methods for the development of linguistic competence. Our development platform is based on a powerful typed feature unification formalism, and the grammar follows the HPSG theory. These choices were made since we wanted on the one hand to facilitate the difficult and time-consuming process of grammar development, and on the other to save our grammar from the fate of several older large-coverage grammars which cannot be reused or extended today because of their idiosyncratic representations.

Since research on systems with multiple cooperating agents constitutes one of the focal areas of our institute, we tried to develop the system in such a way that it would support dialogue among such agents. At the same time, we undertook serious efforts in research on methods that would allow us to derive adequate performance models from the core competence system.

We also built a sample application (COSMA) for appointment scheduling and management based on the competence model, in order to test the grammar of German, methods for dialogue modelling, and certain new methods for deriving a performance model from the competence system.

In the remainder of this paper we will present an overview of the following components and methods:

- development platform including shell, formalism, morphology, parser, generator, semantics;
- German competence including morphology, syntax, semantics;
- methods for providing multi-agent dialogue competence;
- methods for linguistic performance modelling;
- the NL functionality of the sample application COSMA.

Some individual components and methods have been described in more detail in previous publications. However, this paper is the first attempt to present an overview of the integrated system and to describe its parts from the perspective of our overall research strategy.

2 Formalism

For the grammar, the lexicon and parts of the morphology a powerful typed unification formalism $\mathcal{TDL}$ (Type Description Language) has been developed. Reasoning is performed by two specialized inference engines, viz. the $\mathcal{TDL}$ type engine and the feature constraint-solver $\mathcal{UD}i\mathcal{N}e$. The modules are connected via a flexible interface to allow for mutual control.

Type System $\mathcal{TDL}$ is a powerful typed feature-based language and inference system, specifically suited for highly lexicalized grammars [8] (in this volume). Type definitions in $\mathcal{TDL}$ consist of type constraints and feature constraints over the standard boolean connectives $\wedge$, $\vee$, and $\neg$. The operators are generalized in that they can connect feature descriptions, coreference tags (logical variables) and types. $\mathcal{TDL}$ distinguishes between *avm* types (open-world reasoning), *sort* types (closed-world reasoning), *built-in* types, and atoms. Recursive types are explicitly allowed and handled by a sophisticated lazy type expansion mechanism.

$\mathcal{TDL}$ allows the definition of *partitions* and the declaration of sets of types as *incompatible*, meaning that the conjunction of them yields $\perp$. Working with partially as well as with fully expanded types is possible through the use of a sophisticated type expansion mechanism, both at definition and at run time. $\mathcal{TDL}$ is fully incremental in that it allows the redefinition of types and the use of undefined types. $\mathcal{TDL}$ allows a grammarian to define and use parameterized templates (macros). Input given to $\mathcal{TDL}$ is parsed by a LALR(1) parser to allow for an intuitive, high-level input syntax.

Efficient reasoning in the system is accomplished through specialized modules: (i) bit vector encoding of the type subsumption hierarchy; (ii) fast symbolic simplification for complex type expressions; (iii) memoization to cache precomputed results; and (iv) type expansion to make constraints explicit, to determine the global satisfiability of a description, and to work with partially expanded types during processing.

Constraint Solver $\mathcal{UD}i\mathcal{N}e$ is a feature constraint solver capable of dealing with distributed disjunctions over arbitrary structures, negative coreferences, full negation, and functional and relational constraints. It is the first (and to our knowledge the only) implemented feature constraint solver that *integrates* both full negation and distributed disjunctions [2]. $\mathcal{UD}i\mathcal{N}e$ does not use distributed disjunction only as a tool for efficient processing. It also forms part of the input syntax, which allows for very compact representation of the input data. In contrast with other systems using distributed disjunctions, we do not restrict disjunctions to length two, thus reducing the size of the feature structure representations massively.

The functionality of $\mathcal{UD}i\mathcal{N}e$ is completed by several auxiliary functions. It is possible to remove inconsistent alternatives, to simplify structures, to extract subterms or to evaluate functional constraints. One can also construct disjunctive normal form if desired.

Semantic Representation A specialized meaning representation formalism, $\mathcal{NLL}$, developed at Hewlett Packard [9], is used for semantic reasoning and as a flexible interface to various application systems. $\mathcal{NLL}$ is a linguistically motivated extension of sorted first-order predicate logic, integrating also concepts from Situation Semantics and DRT. It provides a large range of representational mechanisms for natural language phenomena.

3 Linguistic Resources

The core of the linguistic resources consists of a two-level morphology with feature constraints, an HPSG oriented grammar of German with integrated syntax and semantics, and a module for surface speech act recognition, all implemented in $\mathcal{TDL}$.

Morphology The component X2MorF, analyzing and generating word forms, is based on a two-level morphology which is extended by a word-formation grammar (described in $\mathcal{TDL}$) for handling the concatenative parts of morphosyntax [15].

Grammar The style of the grammar closely follows the spirit of HPSG, but also incorporates insights from other grammar frameworks (e.g. categorial grammar) and further extensions to the theory [12].

The grammar distinguishes various types of linguistic objects, such as lexical entries, phrase structure schemata, lexical rules, multi-word lexemes etc., all of which are specified as typed feature structures. Lexical rules are defined as unary rules and applied at run-time. Multi-word lexemes are complex lexemes with a non-compositional semantics, such as fixed idiomatic expressions. HPSG principles and constraints are represented by inheritance links in the type lattice. The grammar covers a fair number of the standard constructions of German, and exhibits a more detailed coverage in some specific application oriented areas.

Semantics Feature structure descriptions of the semantic contribution of linguistic items are represented in $\mathcal{TDL}$ and are fully integrated into the grammar. Additionally, the $\mathcal{TDL}$ type system is used to encode and check sortal constraints as they occur in selectional restrictions. For further processing such as scope normalization and anaphora resolution, inferences and application dependent interpretation, the (initial) $\mathcal{TDL}$ semantic descriptions are translated into $\mathcal{NLL}$ formulae.

Speech Act Recognition and Dialogue The grammar provides a typed interface to a speech act recognition module based on HPSG representations of utterances. The assignments of illocutionary force take into account syntactic features, a marking of performative verbs and assignments of fixed illocutionary force to relevant idiomatic expressions.

Recently inference-based dialogue facilities using a quasi-modal logic for multiagent belief and goal attribution [5] have been added to the system. Incoming surface speech act structures are subjected to anaphora and reference resolution, translated into a frame-based action representation, and disambiguated using inferential context. The effects, including communicated beliefs and goals, of the first acceptable speech act interpretation are then asserted.

4 Processing components

Parser and generator provide the basic processing functionality needed for grammar development and sample applications. In addition to the separate modules for parsing and generation, we also experiment with a uniform reversible processing module based on generalized Earley deduction.

Parser The parser is a bidirectional bottom-up chart parser which operates on a context-free backbone implicitly contained in the grammar [6]. The parser can be parameterized according to various processing strategies (e.g. breadth first, preference of certain rules etc.). Moreover, it is possible to specify the processing order for the daughters of individual rules. An elaborate statistics component supports the grammar developer in tuning these control strategies.

In addition, the parser provides the facility to filter out useless tasks, i.e. tasks where a rule application can be predicted to fail by a cheaper mechanism than unification. There is a facility to precompute a filter automatically by determining the possible and impossible combinations of rules; some additional filtering information is hand-coded.

The parser is implemented in an object-oriented manner to allow for different parser classes using different constraint solving mechanisms or different parser instances using different parsing strategies in the same system. With differing parameter settings instances of the parser module are used in the X2MorF and surface speech act recognition modules as well.

Generator Surface generation in DISCO is performed with the SeReal (Sentence Realization) system [4], which is based on the semantic-head-driven algorithm by Shieber et al. SeReal takes a $\mathcal{TDL}$ semantic sentence representation as its input and can deliver all derivations for the input admitted by the grammar. Efficient lexical access is achieved by having the lexicon indexed according to semantic predicates. Each index is associated with a small set of lemmata containing the semantic predicate. Using the same indexing scheme at run-time for lexical access allows us to restrict unification tests to a few lexical items. Subsumption-based methods for lexical access were considered too expensive for dealing with distributed disjunctions. The grammar used for generation is the same as the one used for parsing except for some compilation steps performed by SeReal that, among other things, introduce suitable information wherever 'semantically empty' items are referred to. Rule application is restricted by rule accessibility tables which are computed off-line.

5 Performance Modelling

In our search for methods that get us from the transparent and extensible competence grammar to efficient and robust performance systems we have been following several leads in parallel. We assume that methods for compilation, control and learning need to be investigated. The best combination of these methods will depend on the specific application. In the following some initial results of our efforts are summarized.

Acquisition of Sublanguages by EBL It is a matter of common experience that different domains make different demands on the grammar. This observation has given rise to the notion of sublanguage; efficient processing is achieved by the exploitation of restricted language use in well specified domains.

In the DISCO system we have integrated such an approach based on Explanation-Based Learning (EBL) [14]. The idea is to generalize the derivations of training instances created by normal parsing automatically and to use these generalized derivations (called templates) in the run-time mode of the system. If a template can be instantiated for a new input, no further grammatical analysis is necessary. The approach is not restricted to the sentential level but can also be applied to arbitrary subsentential phrases, allowing it to interleave with normal processing.

Intelligent Backtracking in Processing Disjunctions In [16] a method is outlined for controlling the order in which conjuncts and disjuncts are to be processed. The ordering of disjuncts is useful when the system is supposed to find only the best result(s), which is the case for any reasonably practical NL application. An extension of $\mathcal{UD}i\mathcal{N}e$ has been implemented that exploits distributed disjunctions for preference-based backtracking.

Compilation of HPSG into Lexicalized TAG [7] describes an approach for compiling HPSG into lexicalized feature-based TAG. Besides our hope to achieve more efficient processing, we want to gain a better understanding of the correlation between HPSG and TAG. The compilation algorithm has been implemented and covers almost all constructions contained in our HPSG grammar.

6 Environment

The DISCO DEVELOPMENT SHELL serves as the basic architectural platform for the integration of natural language components in the DISCO core system, as well as for the COSMA application system [13]. Following an *object oriented* architectural model we followed a *two-step approach*, where in the first step the architecture is developed independently of specific components to be used and of a particular flow of control. In the second phase the resulting 'frame system' is instantiated by the integration of existing components and by defining the particular flow of control between these components. Using an object-oriented design together with multiple inheritance has been shown fruitful for the system's modifiability, extensibility and incremental usability.

Several editing and visualization tools greatly facilitate the work of the grammar developer. The most prominent of them, FEGRAMED, provides the user with a fully interactive feature editor and viewer. There are many possibilities to customize the view onto a feature structure, such as hiding certain features or parts of a structure, specifying the feature order and many more. The large feature structures emerging in the process of constraint based formalisms make such a tool absolutely indispensable for grammar debugging. Main goals of the development of FEGRAMED were high portability and interfacing to different systems. Written in ANSI-C, it exists in Macintosh and OSF/Motif versions and is already used at several external sites.

There exists a graphical chart display with mouse-sensitive chart nodes and edges directly linked to the feature viewer, thus making debugging much simpler. It also provides a view of the running parser and enables you to inspect the effects of the chosen parsing strategy visually. A browser for the $\mathcal{TDL}$ type system permits navigation through a type lattice and is coupled with the feature editor. There are other tools as well, e.g., a $\mathcal{TDL}$2LaTeXutility, an Emacs $\mathcal{TDL}$ mode, global switches which affect the behaviour of the whole system etc.

The diagnostics tool (DiTo) [11] containing close to 1500 annotated diagnostic sentences of German facilitates consistency maintenance and measuring of competence. The tool has been ported to several sites that participate in extending the test-sentence database.

7 Putting it to the Test

Cooperative Schedule Management In building the COSMA prototype the DISCO core system has been successfully integrated into an application domain with both scientific interest and practical plausibility, viz. multi-agent appointment scheduling (see Figure 1). Understanding and sending messages in natural language is crucial for this application since it cannot be expected that all participants will have a COSMA system. The use of natural language also makes it easier for the owner of the system to monitor the progress of an appointment scheduling process. Each COSMA instance functions as a personal secretarial assistant providing the following services: (i) storage and organization of a personal appointment date-book; (ii) graphical display and manipulation of appointment data; and (iii) natural language understanding and generation in communication with other agents via electronic mail. The current scheduling functionality includes the arrangement of multi-participant meetings (possibly with vague or underspecified details) as well as the modification and cancellation of appointments that are under arrangement or have already been committed to.

Accordingly, the current COSMA architecture has three major components: a prototype appointment planner (developed by the DFKI project AKA-MOD) that keeps the calendar database, provides temporal resolution and drives the communication with other agents; a graphical user interface (developed inside the DISCO project) monitoring the calendar state and supporting the mouse- and menu-driven arrangement of new appointments and, finally, the DISCO core system (enriched with a set of application specific modules) that provides the natural language and linguistic dialogue capabilities.

Interface to the Core Engine The communication between the DISCO system and the appointment planner is modelled in a restricted appointment task interface language and roughly meets the internal representation of the appointment planner. To connect the two components, DISCO is enriched

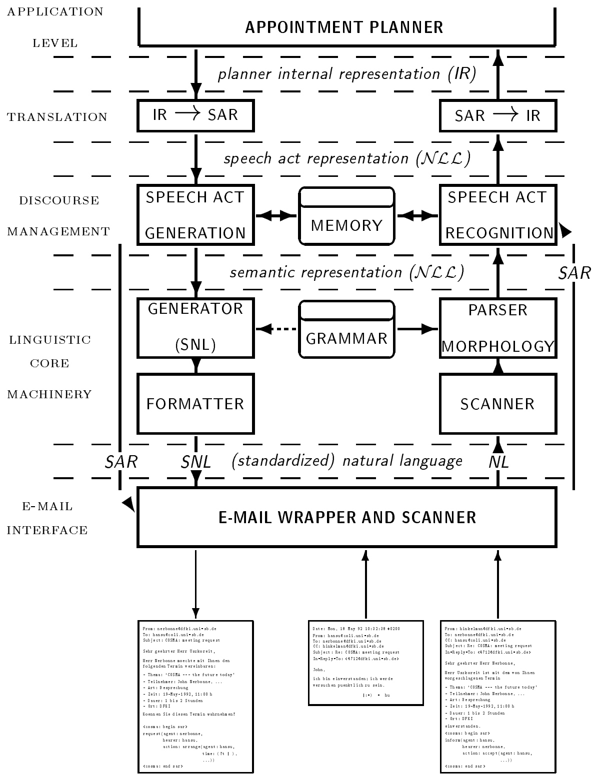

Figure 1: Rough sketch of the DISCO system in its application to the COSMA scenario. The entire COSMA prototype has been built on top of the DISCO DEVELOPMENT SHELL as a monotonic extension to the core system.

with a dedicated interface module that translates between the DISCO internal semantics representation language $\mathcal{NLL}$ and the appointment planner representation. The translation process (maintaining the substantial difference in expressive power between $\mathcal{NLL}$ and the restricted planner language) builds on ideas from current compiler technology with a limited set of domain- and application-specific inference rules [10].

On its opposite end DISCO is hooked up to plain electronic mail facilities through a general purpose e-mail interface that allows it to receive and send e-mail (and in case of processing failures to 'respool' messages to the user mailbox).

8 Discussion and Outlook

We have presented an overview of the DISCO system that illustrates our general research strategy. The system is implemented in Common Lisp and runs on Sun and HP workstations. Some tools and peripheral components are coded in C. A port of the system to another hardware platform (Apple Macintosh) is currently being carried out. The total size of the system is about 100,000 lines of code. Parts of the system were demonstrated at several conferences, at trade fairs and on other occasions.

The work will be continued in two smaller projects. First of all we plan to extend the system's linguistic competence of German. The diagnostic tool DiTo will be expanded in joint work with other groups to provide an instrument for measuring progress and for comparing grammars. We will also continue work on building up dialogue competence. The application demonstrator will be further developed in collaboration with other projects at the DFKI.

In the area of performance modelling, we will continue exploring different approaches for control, compilation, and competence-based learning in parallel. At this point nobody can really foresee which strategy or combination of strategies will yield the best practical results. As we pointed out in the introduction, different application types will require different performance models. High priority is given to the integration of statistical methods in all pursued approaches, since in contrast to competence modelling, statistical data are essential for developing adequate performance models.

Acknowledgements

We acknowledge the invaluable input of our former colleagues, viz., John Nerbonne who substantially contributed to the design and success of the DISCO project, Harald Trost and Jan Alexandersson.

As in many academic environments, major parts of the daily system building have been carried out by our wonderful research assistants, especially Sabine Buchholz, Stephan Diehl, Thomas Fettig, Stefan Haas, Judith Klein, Karsten Konrad, Ingo Neis, Hannes Pirker, Ulrich Schäfer, Oliver Scherf, Jörg Steffen, and Christoph Weyers.

This work has been supported by research grant ITW 9002 0 from the German Bundesministerium für Forschung und Technologie to the DISCO project.

References

[1] Hiyan Alshawi, editor. *The Core Language Engine.* ACL-MIT Press Series in Natural Language Processing. MIT Press, Cambridge MA., 1992.

[2] Rolf Backofen and Christoph Weyers. *UDiNe*—A Feature Constraint Solver with Distributed Disjunction and Classical Negation. Technical report, DFKI, Saarbrücken, Germany, 1994. Forthcoming.

[3] Hans-Ulrich Block. Compiling Trace & Unification Grammar. In Tomek Strzalkowski, editor, *Reversible Grammar in Natural language Processing*, pages 155–174. Kluwer Academic Press, London, 1994.

[4] Stephan Busemann. The SeReal System: putting semantic-head-driven generation to the limits. Technical document, DFKI, Saarbrücken, Germany, 1994. Forthcoming.

[5] Elizabeth A. Hinkelman and Stephen P. Spackman. Abductive Speech Act Recognition, Corporate Agents and the COSMA System. In W.J. Black et al., editor, *Abduction, Beliefs and Context: Proceedings of the Second ESPRIT PLUS Workshop in Computational Pragmatics.* Academic Press, 1992.

[6] Bernd Kiefer and Oliver Scherf. Gimme More HQ Parsers. The Generic Parser Class of DISCO. Technical report, DFKI, Saarbrücken, Germany, 1994. Forthcoming.

[7] Bernd Kiefer, Klaus Netter, and K. Vijay-Shanker. Compilation of HPSG to TAG. Research report, DFKI, Saarbrücken, Germany, 1994. Forthcoming.

[8] Hans-Ulrich Krieger and Ulrich Schäfer. *TDL* — A Type Description Language for Constraint-Based Grammars. In *Proceedings of COLING-94*, 893–899, 1994. An extended version of this paper is available as DFKI Research Report RR-94-37.

[9] Joachim Laubsch and John Nerbonne. An Overview of *NLL*. Technical report, Hewlett-Packard Laboratories, Palo Alto, July 1991.

[10] John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, and Stephan Oepen. Software for Applied Semantics. In Chu-Ren Huang et al., editor, *Proceedings of Pacific Asia Conference on Formal and Computational Linguistics*, pages 35–56, Taipei, 1993. Academica Sinica. Also available as DFKI Research Report RR-92-55.

[11] John Nerbonne, Klaus Netter, Kader Diagne, Ludwig Dickmann, and Judith Klein. A Diagnostic Tool for German Syntax. *Machine Translation:8*, 85–107, 1993.

[12] Klaus Netter. Architecture and Coverage of the DISCO Grammar. In Stephan Busemann and Karin Harbusch, editors, *Workshop on Natural Language Systems: Re-usability and Modularity. Proceedings*, pages 1–10. DFKI, Saarbrücken, Germany, 1993.

[13] Günter Neumann. Design Principles of the Disco System. In *Proceedings of the TWLT 5*, Twente, Netherlands, 1993.

[14] Günter Neumann. Application of Explanation-based Learning for Efficient Processing of Constraint-based Grammars. In *Proceedings of the Tenth IEEE Conference of Artifical Intelligence for Application*, Marriott Riverwalk, San Antonio, Texas, March 1994.

[15] Harald Trost. The Application of Two-Level Morphology to Non-concatenative German Morphology. In *Proceedings of COLING-90*, 1990.

[16] Hans Uszkoreit. Adding Control Information to Declarative Grammars. In *Proceedings of the 29th Annual Meeting of the Association of Computational Linguistics in Berkeley*, 1991.