Lean Formalisms, Linguistic Theory, and Applications. Grammar Development in ALEP.

Paul Schmidt & Axel Theofilidis
& Sibylle Rieder
IAI
Martin-Luther-Str. 14
D-66 111 Saarbrücken
{paul,axel,sibylle}@iai.uni-sb.de

Thierry Declerck
IMS, University of Stuttgart
Azenbergstr. 12
D-70174 Stuttgart
thierry@ims.uni-stuttgart.de

Abstract

This paper describes results achieved in a project which addresses the issue of how the gap between unification-based grammars as a scientific concept and real world applications can be narrowed down[1]. Application-oriented grammar development has to take into account the following parameters: Efficiency: The project chose a so called 'lean' formalism, a term-encodable language providing efficient term unification, ALEP. Coverage: The project adopted a corpus-based approach. Completeness: All modules needed from text handling to semantics must be there. The paper reports on a text handling component, Two Level morphology, word structure, phrase structure, semantics and the interfaces between these components. Mainstream approach: The approach claims to be mainstream, very much indebted to HPSG, thus based on the currently most prominent and recent linguistic theory. The relation (and tension) between these parameters are described in this paper.

1 Introduction

Applications on the basis of unification-based grammars (UG) so far are rather rare to say the least. Though the advantages of UGs are obvious in that properties such as monotonicity, declarativity, perspicuity are important for maintaining and easily extending grammars, their popularity (despite 15 years of history) is still restricted to the academia. This paper reports of a project, LS-GRAM, which tried to make a step further in bringing UGs closer to applications.

Application-oriented grammar development has to take into account the following parameters:

- Efficiency: A major problem of UGs is (lacking) efficiency. For the LS-GRAM project this

---

[1]This project is LS-GRAM, sponsored by the Commission of the European Union under LRE 61029.

led to the decision to use a so called 'lean' formalism, ALEP, providing efficient term unification. 'Leanness' means that computationally expensive formal constructs are sacrificed to gain efficiency. Though this is at the cost of expressiveness, it is claimed that by 'leanness' 'linguistic felicity' does not suffer.

- Coverage: Most grammar development projects are not based on an investigation of real texts, but start from 'the linguists" text book'. This is different here in that a corpus-based approach to grammar development has been adopted which is the implementation of the simple principle that if a grammar is supposed to cover real texts, that the coverage of these texts has to be determined first. The was a corpus investigation in the in the beginning, in the course of which tools have been used and developed which allow for automatic and semi-automatic determination of linguistic phenomena.

- Completeness: All modules needed from text handling to semantics had to ve developed. This is why this paper does not focus on one single topic, but tries to represent the major achievements of the whole of the system. The paper reports on a text handling component, Two Level morphology, word structure, phrase structure, semantics and and very importantly the interaction of these components.

- Mainstream approach: None-the-less, the approach we adopted claims to be mainstream, very much indebted to HPSG, thus based on the currently most prominent and recent linguistic theory.

The relation (and tension) between these parameters is the topic of this paper.

First, we will show, how a corpus-investigation established the basis for the coverage, second, how various phenomena determined by corpus-investigation are treated in text handling (TH), third, how the linguistic modules, Two Level Morphology (TLM), word and phrase structure, the lexicons look like. The last section is devoted to the

efficiency and performance of the system. Figures are given which prove that the system is not so far from real applications[2]

2 Corpus-Investigation with the MPRO-System

The project started with a corpus investigation. It consisted of 150 newspaper articles from the German 'Die ZEIT'. They are descriptive texts from the domain of economy. They were investigated automatically by the (non-statistical) 'MPRO' tagger. 'MPRO' provides the attachment of rich linguistic information to the words. In addition, 'MPRO' provides a built-in facility for resolving categorial ambiguities on the basis of homograph reductions and a facility for handling unknown words which are written on a file. Encoding the missing stems (which were very few) ensured complete tagging of the corpus.

'MPRO' also provides a facility for searching syntactic structures in corpora. A detailed analysis on the internal structure of main clauses, subordinate clauses, verbal clusters, clausal topoi (e.g. structure of Vorfeld and Nachfeld), NPs, PPs, APs, CARDPs, coordinate structures, occurrence of expletives, pronominals and negation occurring in the corpus was made which then guided grammar development.

Another major result of the corpus investigation was that most sentences contain so called 'messy details', brackets, figures, dates, proper names, appositions. Most sentences contain compounds.

In general, most of the known linguistic phenomena occur in all known variations. Description has to be done in great detail (all frames, all syntactic realizations of frames). (Long distant) discontinuities popular in theoretical linguistics did not play a role. In order to give a 'general flavour' of the corpus-investigation one noteworthy result should be reported: 25% of the number of words occur in NPs of the structure [DET (A) (A) N]. But 'A' and 'N' are of a complex and unexpected nature:

- A: (name) - er: Dortmund-er, Schweiz-er
- (figure) - A: 5-jährig, fünfjährig, 68-prozentig
- N: Ex-DDR-Monopolisten (Hyphenated compounding, including names and abbreviations).

The corpus-investigation guided the grammar development. A.o. it showed the necessity to devlop a TH component and separate out specific phenomena from the treatment in the grammar. (This was also necessary from an efficiency point of view).

3 Text Handling

The ALEP platform provides a TH component which allows "pre-processing" of inputs. It converts a number of formats among them 'Latex'. Then the ASCII text first goes through SGML-based tagging: Convertion to an EDIF (Eurotra Document Interchange Format) format, then paragraph recognition, sentence recognition and word recognition. The output of these processes consists in the tagging of the recognized elements: 'P' for paragraphs, 'S' for sentences, 'W' for words (in case of morphological analysis, the tag 'M' is provided for morphemes) and 'PT' for punctuation signs as exemplified in (1).

```
(1)  <P>
       <S>
        <W>John</W>
        <W>sleeps</W>
        <PT>.</PT>
       </S>
     </P>
```

In the default case, this is the information which is input to the TH-LS component (Text-Handling to Linguistic Structure) component. ALEP provides a facility (tsls-rules) which allows the grammar writer to identify information which is to flow from the TH to the linguistic processes. We will show how this facility can be used for an efficient and consistent treatment of all kinds of 'messy details'.

The TH component of the ALEP platform also foresees the integration of user-defined tags. The tag (USR) is used if the text is tagged by a user-defined tagger. An example of an integration of a user-defined tagger between the sentence recognition level and the word recognition level of the TH tool is given below.

The tagger for 'messy details' has been integrated into the German grammar and has been adapted for the following patterns:

1 'Quantities' (a cardinal number followed by an amount and a currency name (e.g. "16,7 Millionen Dollar"))

2 Percentages (12%, 12 Prozent, zwölf Prozent)

3 Dates (20. Januar 1996)

4 Acronyms and abbreviations ('Dasa', 'CDU', 'GmbH', etc.).

5 Prepositional contractions (zum, zur, am etc.)

- Appositions: Prof. Dr. Robin Cooper

We will examplify the technique for 'quantities'. Recursive patterns are described in the programming language 'awk'. (2) defines cardinal numbers (in letters).

```
(2)  two-to-ten = "(([Zz]wei)|([Dd]rei)|
         ([Vv]ier)|([Ff]|nf)|
         ([Ss]echs)|([Ss]ieben)|
         ([Aa]cht)|([Nn]eun)|([Zz]ehn))"
     eleven-to-nineteen = ....
     twenty-to-ninety
     etc.
     card = "(([Ee]in|"two-to-ten")
         (und"twenty-to-ninety")?|"....)
```

On the basis of these variables other variables can
be defined such as in (3).

```
(3)  range = "("number"|"card")"
     amount ="("Millionen"|"Milliarden")
     currency="("Mark","DM","Dollar")"
     curmeasure="("amount"??"currency"?)"
     quantity ="("range" "curmeasure")"
```

The following inputs are automatically recognized
as 'quantities':

"Zweiundzwanzig Dollar"
"Sechsundzwanzig Milliarden"
"Dreiundvierzig Milliarden DM"

This treatment of regular expressions also means a
significant improvement of efficiency because there
is only one string whereas the original input con-
sisted of five items ("vierzig bis fünfzig Milliarden
Dollar"): "vierzig_bis_fuenfzig_Milliarden_Dollar".

(4) gives an example for information flow from TH
to linguistic structure:

```
(4)  ld:{
       spec => spec:{
              lu => TYPE},
       sign => sign:{
           string => string:{
                first => [ STRING | REST],
                rest => REST},
           synsem => synsem:{
               syn => SYN => syn:{
                   constype => morphol:{
                       lemma => VAL,
                       min => yes } } } } },
       'USR',['TYPE' => TYPE,
              'VAL' => VAL], STRING ).
```

The feature 'TYPE' bears the variable TYPE (in
our case: "quantities"). The feature 'VAL' repre-
sents the original input (e.g.: "fünfzig Milliarden
Dollar") and the variable STRING represents the
output string of the tagged input (in this case:
"fuenfzig_Milliarden_Dollar"). This value is co-
shared with the value of the "string" feature of the
lexicon entry. The definition of such generic entries
in the lexicon allows to keep the lexicon smaller
but also to deal with a potentially infinite number
of words.

These strategies were extended to the other phe-
nomena. The TH component represents a pre-
processing component which covers a substantial
part of what occurs in real texts.

4 The Linguistic Modules

4.1 Two Level Morphology (TLM)

The TLM component deals with most major mor-
phographemic variations occurring in German, such
as umlautung, ss-ß alternation, schwa-instability.

We will introduce this component by way of ex-
emplification. (5) represents the treatment of 'e'-'i'
umlautung as occurring in German verbs like 'gebe',
'gibst', referring to (Trost90).

An ALEP TL rule comes as a four to five place
PROLOG term, the first argument being the rule
name, the second a TL description, the third (rep-
resented by the anonymous variable) a specifier
feature structure, the fourth a typed feature struc-
ture constraining the application of the rule and
a fifth allowing for linking variables to predefined
character sets.

```
(5)  tlm_rule(
     umlautE_no,
     [] [e] [] <=> [] ['E'] [],
     _,
     syn:{
        constype => stem:{
            phonol => pho:{
                umlaut => no } } } ).

     tlm_rule(
     umlautE_i_yes,
     [] [i] [C] <=> [] ['E'] [],
     _,
     syn:{
        constype => stem:{
            phonol => pho:{
                umlaut => (yes&i)} } },
     [C in consonants] ).
```

In order to understand the treatment one has to
bear in mind that there exists the lexical entry in
(6) to which the TL rules above map to.

$$(6) \quad \left[syn \mid cons \begin{bmatrix} \text{lemma gEb} \\ \text{phonol [umlaut none\&i]} \end{bmatrix} \right]$$

The morphologically relevant information is en-
coded in 'cons'. It contains two features, 'lemma'
which encodes the abstract morpheme with a cap-
ital 'E' (this is the basis for a treatment according
to ((Trost90)) and the feature 'phonol' which en-
codes phonologically relevant information, in this
case whether umlautung is available or not.

The values for 'phonol' is a boolean conjunction
from two sets: s1 ={none, no, yes} and s2 = {e,
i}.

The treatment consists of mapping a surface 'e' to a
lexical 'E' in case the constraint which is expressed

```

as a feature structure in the fourth argument holds. It says that for the first rule 'e' is mapped on 'E' if the feature 'umlaut' has a value which is 'no'. This applies to (6). This handles cases such as 'geb-e'. The second rule maps 'i' 'E' if 'umlaut' has the value 'yes & i'. This also holds in cases like 'gib-st'. One would expect according to (Trost90) that only two values 'no' and 'yes' are used. The change has been done for merely esthetic reasons. The '2nd pers sing' morpheme 'st' e.g. requires an 'umlaut = yes' stem, if the stem is capable of umlautung, at all which is the case for 'gibst'. In case the stem cannot have umlautung (as for 'kommst') 'st' also attaches. This makes that non-umlautung stems have to be left unspecified for umlautung, as otherwise 'st' could not attach. 'st' can now be encoded for 'umlaut = no'.

## 4.2 Lexicon

Lexical information is distributed over three lexicons:

- A TL lexicon which contains information relevant for segmentation, exclusively.
- A syntax lexicon.
- A semantic lexicon.

The distribution of information over three lexicons has a simple reason, namely avoiding lexical ambiguities at places where they cannot be resolved or where they have impact on efficiency. So, e.g. the verbal suffix 't' has lots of interpretations: '3rd pers sing', 2nd pers pl', preterite and more. These ambiguities are NOT introduced in the TLM lexicon as the only effect would be that a great number of segmentations would go into syntactic analysis. Only then these ambiguities could be resolved on the basis of morphotactic information. A similar situation holds on syntactic level. There is no point in multiplying syntactic entries by their semantic ambiguities and make all of these entries available for analysis. It would result in a desaster for efficiency. Semantic reading distinctions thus are put into the (semantic) refinement lexicon. We would like to introduce lexical information for the preposition 'in' by way of illustration.

**TL-Entry for 'in':**

$$
(7) \quad \begin{bmatrix} \text{string} & [\text{in} \mid \_] \\ \text{synsem} & \begin{bmatrix} \text{syn} & \begin{bmatrix} \text{cons} & \begin{bmatrix} \text{lemma in} \\ \text{umlaut no} \end{bmatrix} \\ \text{word} \end{bmatrix} \end{bmatrix} \end{bmatrix}
$$

The morphological information is encoded in the 'cons' feature.

**Analysis-Entries for 'in':**

Prepositions may occur in 'strongly bound PPs' where they are functional elements (semantically empty, but syntactically relevant). This is encoded in (8). A PP headed by a functor cannot be an adjunct (mod=none). The head-dtr chosen by 'in' is an NPacc. The mother of such a construction also comes out as NPacc which is encoded in 'projects'. The major reason for such a treatment lies in the fact that it allows for a unified treatment of all functional elements like inflectional affixes, complementizers, auxiliaries, infinitival zu, functional prepositions etc..).

$$
(8) \quad \begin{bmatrix} \text{string} & [\text{in} \mid \_] \\ .. \mid \text{syn} \mid \text{cat} & \begin{bmatrix} \text{head}_n & [\text{mod none}] \\ \text{subcat}_{\text{func}} & \begin{bmatrix} \text{selects NPacc} \\ \text{projects NPacc} \end{bmatrix} \end{bmatrix} \end{bmatrix}
$$

(9) is the entry for 'in' as a head of a PP subcategorizing for an NPacc.

$$
(9) \quad \begin{bmatrix} \text{string} & [\text{in} \mid \_] \\ .. \mid \text{syn} \mid \text{cat} & \begin{bmatrix} \text{head}_p & [\text{mod } ] \\ \text{subcat}_{\text{subst}} & [\text{comps } \langle \text{NPacc} \rangle] \end{bmatrix} \end{bmatrix}
$$

**Semantic entries for 'in':**

Prepositions need (semantically) different entries depending on whether the p heads a PP which is a complement or an adjunct.

**'in' as complement:**

$$
\begin{bmatrix} \text{syn} \mid \text{cat} \mid \text{subcat} & \begin{bmatrix} \text{subj} <> \\ \text{compls} \langle [...\mid \text{cont } \boxed{1}] \rangle \end{bmatrix} \\ \text{sem} & \begin{bmatrix} \text{cont} & \begin{bmatrix} \text{psoa} & \begin{bmatrix} \text{rel in} \\ \text{arg1} \boxed{1} \end{bmatrix} \end{bmatrix} \\ \text{r\_psoa} \end{bmatrix} \end{bmatrix}
$$

The content of a PP is a relational psoa.

**'in' as Adjunct:**

$$
\begin{bmatrix} \text{syn} \mid \text{cat} & \begin{bmatrix} .. \mid \text{mod} \mid ... \mid \text{cont} & \begin{bmatrix} \text{quants} [1] \\ \text{rd\_cont} & \begin{bmatrix} \text{psoa} [2] \\ \text{restr}[3] \end{bmatrix}_{\text{r\_psoa}} \end{bmatrix} \\ \text{subcat} \mid \text{comps} < .. \mid \text{sem} \mid \text{cont} [4] > \end{bmatrix} \\ .. \text{cont} & \begin{bmatrix} \text{quants} [1] \\ \text{rd\_cont} & \begin{bmatrix} \text{psoa} [2] \\ \text{restr} < \begin{bmatrix} \text{rel in} \\ \text{arg1} [4] \end{bmatrix} \mid [3] > \end{bmatrix}_{\text{r\_psoa}} \end{bmatrix} \end{bmatrix}
$$

The preposition puts its content on a restriction list. It composes the restriction list with the restriction list of the modified item. Quants and the psoa are copied.

## 4.3 Word Structure and Phrase Structure (PS)

Both the word structure and the phrase structure component are based on the same small set of binary schemata closely related to HPSG. In the system described here they exist as macros and they are spelt out in category-specific word and phrase structure rules. (Efficiency is the major reason, as underspecified syntax rules are very inefficient).

Such a schema is e.g. the following head-comp-schema.

Mother:

$$\left[ \text{synsem} \left[ \text{syn} \left[ \text{cat} \left[ \begin{array}{l} \text{head } \boxed{1} \\ \text{subcat} \left[ \begin{array}{l} \text{compls } [5] \\ \text{subj}[2] \end{array} \right] \end{array} \right] \right] \text{sem } [3] \right] \right]$$

Head-dtr:

$$\left[ \text{synsem} \left[ \text{syn} \left[ \text{cat} \left[ \begin{array}{l} \text{head } \boxed{1} \\ \text{subcat} \left[ \begin{array}{l} \text{compls } < [4]|[5] > \\ \text{subj } [2] \end{array} \right] \end{array} \right] \right] \text{sem } [3] \right] \right]$$

Comp-dtr:
$$\left[ \text{synsem } \boxed{4} \right]$$

Head information is propagated from head-dtr to mother, so is semantic information. 'subact' information is structured slightly differently as in HPSG to allow for one innovation wrt HPSG which is our treatment of functional elements.

Mother:
$$\left[ \begin{array}{l} \text{SYN } \boxed{3} \mid \text{CONSTR funct\_att} \\ \text{CAT } \left[ \text{HEAD } \boxed{1} \left[ \begin{array}{l} \text{BASE } \boxed{5} \\ \text{SUBCAT } \boxed{6} \end{array} \right] \right] \\ \text{SEM } \boxed{4} \end{array} \right]$$

Head-dtr:

$$\boxed{2} \left[ \text{SYN} \mid \text{CAT} \left[ \begin{array}{l} \text{HEAD} \mid \text{BASE } \boxed{5} \\ \text{SUBCAT } \boxed{6} \end{array} \right] \right]$$

Functor:
$$\left[ \begin{array}{l} \text{SYN} \mid \text{CAT} \left[ \begin{array}{l} \text{HEAD } \boxed{1} \\ \text{SUBCAT} \left[ \begin{array}{l} \text{SELECTS } \boxed{2} \\ \text{PROJECTS } \boxed{3} \end{array} \right] \\ \textit{funct} \end{array} \right] \\ \text{SEM } \boxed{4} \end{array} \right]$$

The functor macro is highly general. It shows the functor treatment applied in the German grammar, namely that the functor selects its head-dtr,

combines itself with the head-dtr and projects the mother. More specifically: The functor-dtr, indicated by the value 'funct' of the attribute 'subcat' shares the value of the attribute 'selects' with the 'synsem' value of the head-dtr and its 'projects' value with the 'syn' attribute of the mother. The 'head' value is shared between head-dtr and mother, the 'base' value in addition between head-dtr and functor. The subcategorization is shared between head-dtr and mother.

The difference to the head-comp schema is that head information comes from the functor, also the semantics. 'subcat' is inherited from the head-dtr. The powerful mechanism comes in by the subcat-feature of the functor which allows for a very detailed specification of the information to be projected.

The PS component covers all categories, especially all clausal categories (main clauses, subordinate clauses, relatives), NPs, APs, ADVPs, PPs.

## 5 'Efficiency' and Performance

In this section we would like to address the topic of efficiency. A number of points contributing specifically to efficiency should be summarized here.

- ALEP is designed to support efficiency as far as the formalism ('lean' approach) is concerned. Formal constructs known to be computationally expensive are not available[3].

- Refinement (mentioned already) is a monotonic application of phrase structure rules and lexical entries to further featurize (flesh-out with features) a linguistic structure, established in analysis.

  If $Q_1$ is the linguistic output structure of the analysis, then $Q_2$ is the output structure of 're-finement' if $Q_1$ subsumes $Q_2$, i.e. every local tree in $Q_2$ and every lexical entry in $Q_2$ is subsumed by a corresponding local tree and a corresponding lexical entry in $Q_1$.

  Any non-deterministic backtracking algorithm (depth-first) is badly effected by ambiguities as it has to redo repeatedly large amounts of work. In terms of lingware development this means that lexical ambiguities have to be avoided for analysis. As on the other hand lexicalist theories result in an abundance of lexical ambiguities, 'refinement' is a relief. Optimal distribution of information over analysis and refinement results in a gain of efficiency by several factors of magnitude.

- Head selections: ALEP allows for user-defined parsing head declarations as "the most appro-

---

[3] It should have been shown in the precious sections that felicitous descriptions are possible anyway.

priate choice of head relations is grammar dependent" (Alsh91), p.318. On the basis of the user-defined head relations the reflexive transitive closure over head relations is calculated. It has to be made sure that the derived relations are as compact as possible. Optimal choice of head relations pays off in a gain in efficiency by several factors of magnitude.

- Keys: Keys are values of attributes within linguistic descriptions defined by path declarations. Keys allow for indexation and efficient retrieval of rules and lexical entries. This becomes extremely relevant for larger-scale resources. A key declaration which the grammar developer may do identifies the atomic value which is to serve as a key. Optimal keys again result in a substantial gain in efficiency.

- Last not least tuning the grammars with a view on efficiency has contributed to the current performance of the system.

In the following we would like to give some actual figures which may illustrate performance. These figures are not meant to be an exact measurement as exact measurements are not available. In order to give an indication it may be said that ALL the phenomena which increase indeterminism in a grammar of German are covered: All forms of the articles ('die', 'der') and homomorphous relative pronouns, all readings of verbs ( all frames, all syntactic realizations of complements), semantic readings, prepositions and homomorphous prefixes, PPs as nominal adjuncts, as preadjectival complements, as adjuncts to adverbs, as VP adjuncts, valent nouns (with optional complementation), all readings of German 'sein', coordination, N + N combinations, relatives, Nachfeld.

One result of the corpus investigation was that 95% of the sentences in the corpus have between 5 and 40 words. The grammar is able to parse sentences with up to 40 words in 120 secs. The following are corpus examples containing time-consuming parse problems.

```
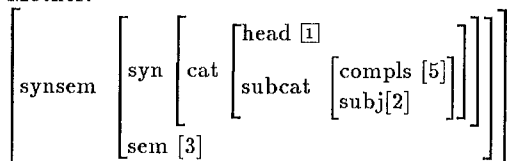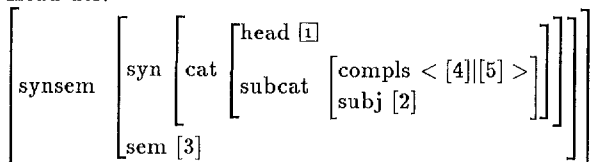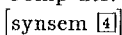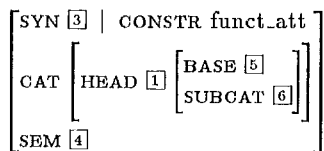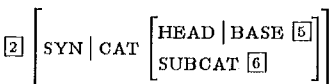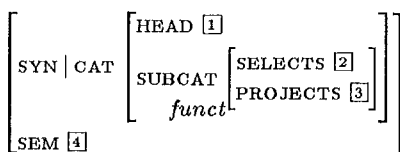Input: In den Wochen vor Weihnachten konnte der
stolze Vorsitzende der zu Daimler-Benz
gehoerenden Deutsche Aerospace AG ein
Jahresergebnis, das alle Erwartungen
uebertraf, verkuenden.
(Comment: In the weeks before X-mas the proud
head of the Deutsche Aerospace AG which belongs
to Daimler-Benz could announce an annual
statement of accounts which exceeds all
expectations.)
           total   RWordSeg RLiftAna Refine
sol : 1   34.450   0.380   34.070   0.000

Input: Dieser Erfolg ueberrascht in zwei
Hinsichten.
(Comment: This success is surprising in two
respects.)
           total   RWordSeg RLiftAna Refine
```

```
sol : 1    1.910    0.130     1.780     0.000
```

For industrial purposes this may still be too slow, but we think that the figures show that the system is not so far away from reality.

6 Conclusions

This paper was about the following aspects of lingware development:

- Linguistic felicity and leanness.
- Leanness and efficiency.
- Methods for large-scale grammar development.
- 'Holistic' approach.

We can summarize briefly:

- ALEP provides all modules and tools from text handling to discourse processing (the latter not introduced here). The lingware created is especially interesting in that it provides an integrated design for all the modules.

- The formalism for lingware development is lean, but it provides sufficient means to support mainstream felicitous linguistic descriptions.

- Efficiency is good compared to other unification-based systems. It is not yet ready for immediate commercial applications, but it is neither very far away.

- The corpus-based approach to grammar development is the only realistic way to get closer to a coverage that is interesting from an application point of view.

ALEP is a promising platform for development of large-scale application-oriented grammars.

References

Hiyan Alshawi, Arnold D J, Backofen R, Carter D M, Lindop J, Netter K, Pulman S G, Tsujii J and Uszkoreit H, (1991), *Eurotra ET6/1: Rule Formalism and Virtual Machine Design Study (Final Report)*, CEC 1991.

H. Trost: The application of two-level morphology to non-concatenative German morphology. *Proceedings of COLING-90*, Helsinki, 1990, vol.2, 371-376.