An Interface between Text Structure and Linguistic Description – a DRAFT VERSION of the later published contribution

Thierry Declerck*

February 1, 1999

Abstract

This paper describes various uses of the Text Handling to Linguistic Structure (TH-LS) component of the Advanced Linguistic Engineering Platform (ALEP). Basically, the Text Handling (TH) subsystem of ALEP performs a conversion of the input string to a SGML-marked text. The TH-LS component consists in a set of so-called *tsls* rules defining a mapping between textual structures (TS) and (partial) linguistic descriptions (PLS). The instantiated PLS are the input for the linguistic parser, dealing with linguistic structures (LS). I show how an adequate use of the TH-LS interface permits the modularization of the lingware and the definition of subgrammars, from morpheme level to the whole text, taking into consideration both processing steps and languages. The TH subsystem of ALEP also foresees the `<USR>`tag for user-supplied markup. An intensive use of this possibility and also the integration of information delivered by a PoS tagger into the TH component allowed both a substantial extension of the coverage and a significant improvement of the efficiency of the ALEP-based grammars, the parser getting as an input linguistically enriched PLS.

Introduction

The starting point of this work has been the development of ALEP-based grammars within the LS-GRAM project[1]. This project was concerned with the investigation of methodologies for the development of large-scale grammar and linguistic resources have been developed for 9 European languages. These resources have been defined within the ALEP-formalism, which is a so-called lean unification-based formalism. The overall architecture of LS-GRAM grammars is outlined in figure 1.

*DFKI GmbH (German Research Center for Artificial Intelligence).

[1]The LS-GRAM (Large-Scale GRAMmars for EC languages) project, was funded by the CEC, number LRE 61029. See (BIM/SEMA 1993), (Simpkins et al. 1993) and (Alshawi et al. 1991) for the ALEP platform, (Schmidt et al. 1996) for a short description of the LS-GRAM project and (Block and Humphreys 1996) for the final review of the project. The LS-GRAM grammars and the documentations are available at: http://www.iai.uni-sb.de/LS-GRAM/home.html

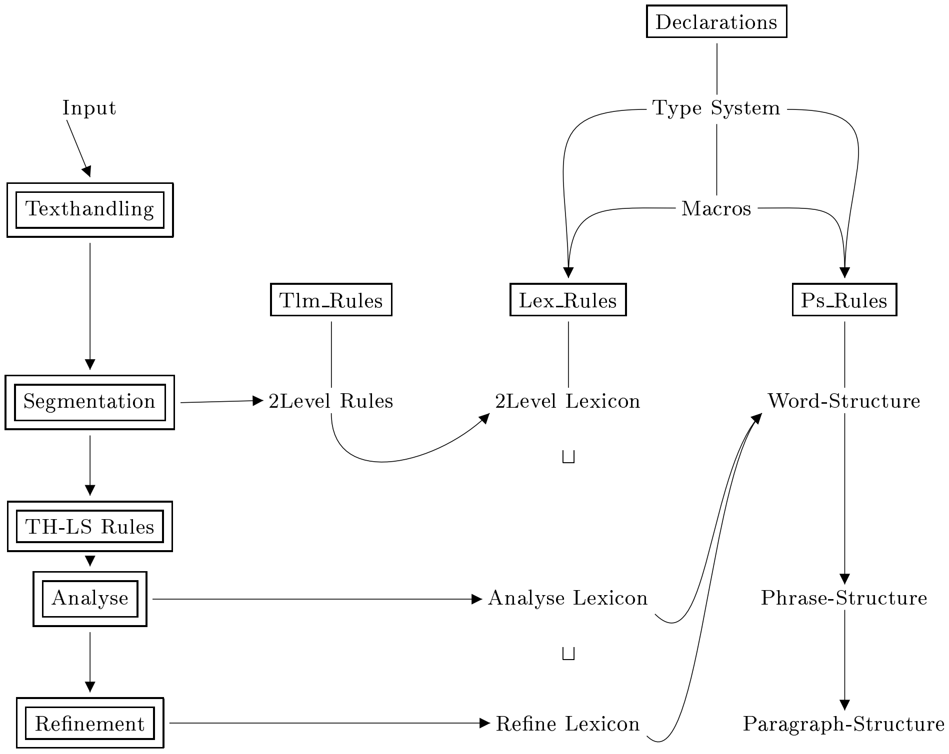

Figure 1: *The overall architecture of LS-GRAM grammars*

In this figure, the reader can see how subgrammars have been defined in dependency of the processing steps (listed in the left column) provided by the ALEP-system (where the sequence of steps can be modified or extended). The subgrammars have also been designed according to the distinct levels of linguistic description. Sections 1.2 and 1.3 describe the role played by the TH-LS component in the modularization of the grammar.

During the last phase of the project, much attention has been given to the Text Handling (TH) subsystem of the ALEP-platform. The initial motivation for this was the fact that LS-GRAM was committed with the treatment of so-called *messy details* – constructions such as dates, numbers, measures, proper names, etc. –, which are word constructs occuring quite often in 'real life' texts. The strategy followed in order to deal with those phenomena within the TH subsystem will be described in section 2, where the task performed by the TH-LS component will be explained in more details.

A third use of the TH-LS interface has been concerned with the integration of part-of-speech (PoS) information delivered by a tagger. In section 3, we will see how this strategy has been rendered possible with a minimal adaptation of the

TH-LS component.

1 The TH-LS component of ALEP

1.1 The TH subsystem

The ALEP-platform provides the user with a Text Handling (TH) component which allows a "preprocessing" of inputs. Input texts will first go through a processing chain consisting in a SGML-based tagging of their elements. The default setup of the system defines the following processing chain: the text is first converted to an EDIF (Eurotra Document Interchange Format) format. Then a subsequent recognition process is provided: paragraph recognition, sentence recognition and word recognition. The output of those processes consists in the tagging of the recognized text 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. Some specialized features are also provided for the tagged words, allowing to characterize them more precisely, so for example `<ACRO>` for acronyms or `<D>` for digits and so on.

The simple ASCII input string "John loves Mary.", after going through the recognition process of the default TH component, will take the form:

```
<P><S><W>John</W><W>loves</W><W>Mary</W><PT>.</PT></S></P>
```

`<P>` and `</P>` mark the beginning and the respective ending of the recognized paragraph structure. The same remark is valid for the other tags and their corresponding structures. The complete output of the TH subsystem for the word level is as follows:

```
<W TYPE="WORD" CASE="FIRST">john</W>
```

Here we know that we deal with a word beginning with uppercase and having as data content the string "john".

The point being here that the starting point for the linguistic processing will never be an ASCII input, but a SGML structured string. And since, as will be shown in the next section, a correspondence can be established between text elements and linguistic objects, there will be no problem in extending the domain of applications of syntactic rules to sentence-external text structures. This is not only important if one wants to process units bigger than sentences, where for example one would like to deal with cross-sentential phenomena like anaphora resolution (see Declerck (1996)), but also for applications like Grammar Checking, where the checking at sentence-level can be influenced by text-structure information (see Adriaens (1996)).

1.2 The tsls-rules

In the default case, it is the kind of text-structured string described above which provides the input to the TH-LS component of the system. Within this subsystem one has to specify so-called *tsls* (text structure to linguistic structure) rules, which map the TH output into partial linguistic descriptions[2]. The syntax of these rules is as follows:

```
tsls_rule( <ld>, <tag_name>, [<features>], <tag_content> )
```

where:
<ld> is a (partial) Linguistic Description (LD). <tag_name> is the name of a SGML tag (e.g. <S>, <W>). <features> is a list of feature-value descriptions of the tag's features. <tag_content> is the atomic content of the string within the tag (optional). As shown in figure 2 – a tsls rule for the word level – this flow of information is in case of the default use of the TH subsystem just a mapping between the text structure tag (with an empty list of feature-value descriptions[3]) and some information described within the linguistic type *ld*. The list of feature-value descriptions of the textual tag will be the place where it will be possible to integrate richer linguistic information provided by PoS taggers (or other tools), as can be seen in section 3.

$$
\begin{array}{l}
\text{ts\_ls\_rule(} \\
{}_{ld}\left[\begin{array}{ll}
\text{SPEC} \mid \text{LEVEL\_SPEC} \mid \text{M\_W} & \text{yes} \\
\text{SIGN} \mid \ldots \mid \text{SYN} \mid \text{CONSTYPE} & {}_{phrasal}\left[\begin{array}{ll} \text{MIN} & \text{yes} \\ \text{CONSTR} & \text{(w\_form;lexical)} \end{array}\right]
\end{array}\right], \\
\text{W, [ ]).}
\end{array}
$$

Figure 2: *Correspondence established between the partial linguistic description and the* W *markup of text*

1.3 Modularizing the lingware

The appropriate design of the tsls rules allows the definition of subgrammars, thus describing a modularization of the grammar ressources at a very early stage of processing. In fact, at the level of TH a lot of information can be dealt with, also due to the commonality of the representation format: SGML.

1.3.1 The specifier features

One of the specificity of ALEP consists in the possible encoding of procedural information as the value of so-called *specifier feature* described within the linguistic description – and also in the same format – but without weakening the declarativeness of the linguistic description. In figure 2, the reader can see the instantiation of such an object within the *ld* feature structure: the *spec* feature. Specifier features are in ALEP designated feature structures which serve to encode the memberschip of rules or other linguistic objects. This encoding of membership realizes at the

[2]This partial linguistic description will then provide the direct input to the linguistic parser.
[3]The values of the features TYPE and CASE are not playing a role here.

same time an internal partition of the grammar. In figure 2 only the rules belonging to the class of rules concerned with objects having already undergone the process of word building (*m_w = morph_to_word*) will be accessed. So all the rules or entries concerned with morphemes will not be accessed once a word is mapped onto a PLS. The reader can imagine that this issue permits an improvement of the performances of the systems (the application of rules being strongly controlled) besides the desired elegance of achieved modularity.

1.3.2 Processing levels

The partitioning described just above has been called the *horizontal* one (see Declerck et al. (1995) and Theofilidis (1997a)) in the sense that certain rules may be specified to be accessed only when parsing morphemes to words or words to sentences etc. But a *vertical* partitioning is also possible along the line of the sequence of processing levels, as can be seen in figure 1. So the lexicon, for example, can be subdivided in dependency of the distinct processing steps. This plays a role in order to reduce the numbers of ambiguities at processing levels for which the ambiguities are actually not relevant. The processing level of *refinement* being here an ALEP specific construction: once an analysis tree has been processed by the *analysis* processing step, refinement can come into play and in a monotonous way apply rules or entries to already consolidated structures, just further decorating the obtained trees. This strategy has been applied, among others, in order to avoid the production of multiple analysis trees where the ambiguities are semantic in nature. The gain of efficiency is here that the multiple production of parse trees is very expensive, where the subsequent decoration of structures is not time consuming. Therefore certain lexicon entries or structure rules are applied only at the refinement processing level, allowing an efficient filtering or enriching of already instantiated syntactic structures.

A lexicon entry which should be accessed only at the processing level of refinement can have following (partial) *spec* feature structure (figure 3):

$$
{}_{ld}\left[\begin{array}{ll}
\text{SPEC} & {}_{spec}\left[\begin{array}{ll} \text{LEVEL\_SPEC} \mid \text{M\_W} & \text{yes} \\ \text{OPER\_SPEC} \mid \text{REF} & \text{yes} \end{array}\right] \\
\text{SIGN} \mid \ldots \mid \text{SYN} \mid \text{CONSTYPE} & {}_{phrasal}\left[\begin{array}{ll} \text{MIN} & \text{yes} \\ \text{CONSTR} & (\text{w\_form;lexical}) \end{array}\right] \\
\ldots &
\end{array}\right],
$$

Figure 3: *A word level lexicon entry which will be accessed only at the refinement processing level of sc ALEP*

1.3.3 Multilingual input

The designated specifier feature can also be used for the purpose of the processing of multilingual input. The overall language of the text to be analyzed is declared in the default setting of the TH subsystem. But very often, portion of a text are written in a distinct language. Suppose this information on alternate language is available, the place where this information will be encoded is also the TH component, where the default value of the *language* feature will be overwritten. The TH-LS component can than ensure that from the distinct lingwares loaded in the system, the relevant language will be chosen. We have still very little experience with this kind of task, but small scale exercises seemed promising.

The alternance of the languages in a text can be directly encoded in a SGML manner or also via the use of a specialized editor, designed for the production of multilingual texts. For this second case we are thinking for example at the use of *MtScript*, a multilingual text editor delivered by the Multext project[4]. The detection of alterning language could also be (semi)-automatic, as we experienced with the `mpro` tool[5]. So having processed following input: "Boris Becker machte die *big points*", the `mpro` system was able to add following features to the two english words: {flang = en}.

In any case: the TH-LS component provides the system with a flexible interface for the processing of multilingual input. What is still missing is the description of appropriate syntactic rules within the linguistic component of the system.

We have decribed the use we made of the TH-LS component in order to subdivide the grammar in modules, this on the base of the default mapping between text structures and PLS and with the user-defined instantiation of *specifier* features. This step already allowed some achievements in the sense of the development of efficient medium-scale grammars, within the framework of a unification-based formalism. Also the maintainability of the LS-GRAM grammars has been improved and a high degree of modularity has been reached. The appropriate encoding of the TH-LS component proved to be very important and, even if this instantiation was quite simple (only four tsls rules were necessary), it contributed in a crucial way to the achieved results.

2 The <USR> tag for user-supplied markup

2.1 User-defined tags

The TH component of the ALEP-platform also foresees the use of a `<USR>` tag, which can be activated if the text is marked-up by user-defined programs, which in the context of LS-GRAM grammars have been implemented in both `awk` and `Perl` and have been named `tagit`. We applied this technique in order to cope with *messy details*.[6] The central idea has been to deal with those word constructs with the help of small user-defined tokenizers which describe patterns of regular expressions, against which the input text is matched. Recognized expressions are then marked

[4]The Multext project has been funded by various institutions. See for more information: http://www.lpl.univ-aix.fr/projects/multext/index.html. The MtScript can be downloaded at: http://www.lpl.univ-aix.fr/projects/multext/MtScript/.

[5]The `mpro` tool has been developed by Heinz Dieter Maas at the IAI in Saarbrücken, see Maas (1996) and Maas and Hirschfeld (1996).

[6]This strategy has been first described by Music (1995) and then extended to other LS-GRAM grammars, as has been briefly documented in Bredenkamp et al. (1996).

with the <USR> tag and get an alternative treatment by the parser which allow to avoid the definition of (potentially infinite) additional lexicon entries and parse-time consuming detailed syntax rules. This marking step has been plugged in the processing chain of the TH subsystem of ALEP between the levels of sentence and word recognition. Every sequence of words which has been marked by the <USR> tag will then be skipped by the subsequent process of word recognition.

2.2 Messy details

Messy details are text constructs which do not lend themselves well to treatment by traditional techniques for linguistic analysis, whence their 'messiness'. Typical examples are numbers, codes, proper names or other (sequences of) word-forms which can occur in many variations, making difficult a comprehensive treatment by the means of unification-based formalisms (ALEP being just an example): one would have to code them individually in some lexicon and/or to implement sets of grammar rules for parsing them syntactically.

To give just one example of messy details we have been dealing with, take the case of currency expressions. Strings like "dreiundvierzig_milliarden_dollar" are recognized by the `tagit` program as a currency expression and enclosed within <USR> tags:

```
<USR LEVEL=W TYPE=CURRENCY_MEASURE STR="Dreiundvierzig Milliarden
  Dollar">Dreiundvierzig_Milliarden_Dollar</USR>
```

The value of the feature TYPE gives the name of the type of word construct we are dealing with. And the three items of the recognized string are collected together (by means of underscores) in order to form just one unit to be processed by the parser later. Thus the whole string will be considered as being just one generic lexicon entry (see section 2.3). Multiple words units are handled in exactly the same way.

2.3 Generic lexicon entries

Unification-based grammar development platforms commonly foresee the definition of generic lexicon entries, in order to allow the reduction of the number of entries sharing some common linguistic properties. In our case we extended this approach to strings which have been concatenated to one string during the processing by the `tagit` program. The shared property in the generic lexicon entries consists in the value of the text feature TYPE (see above). In ALEP such an entry has the form showed in figure 4:
All expressions markep up with the <USR> tag during the extended TH processing of ALEP will be mapped onto such an entry in the internal lexicon of ALEP. How this works is described in section 2.4. Important to note here, is the fact that the definition of such generic entries in the lexicon allows not only to keep the lexicon smaller but also to deal with a potentially infinite number of (sequence of) words.

Such entries are distinguished from the other by the specifier feature *usr_type* having non-nil has a value. We have thus another criteria for defining sub-grammars

$$
{}_{ld}\left[\begin{array}{ll}
\textsc{spec}\,|\,\textsc{usr\_type} & \text{CURRENCY\_MEASURE} \\
\textsc{sign}\,|\ldots|\,\textsc{cat} & {}_{cat}\left[\begin{array}{ll}
\textsc{head} & {}_{n}\left[\textsc{agr} \quad (\text{pl\&p3})\right] \\
\textsc{subcat} & \langle\rangle \\
\ldots &
\end{array}\right]
\end{array}\right]
$$

Figure 4: *Generic lexical entry for currency expressions*

within the overall ALEP-based grammar: linguistic object which are accessed on the base of their user-defined preprocessing. This is important in case of desired reusability of the resources, since user-defined marked entries or rules must be delivered with the involved preprocessing tools.

2.4 The tsls-rules for the <USR> tag

As has been said above, the system still needs to know when the generic entries will be accessed, since for this purpose no string information is available. Here again the TH-LS will do the job. Once an input has been marked with the `<USR>` tag during TH processing level, the mapping onto a PLS is ensured by a new tsls rule, especially designed for this goal, as can be seen in figure 5. The reader

ts_ls_rule(

$$
{}_{ld}\left[\begin{array}{ll}
\textsc{spec}\,|\,\textsc{usr\_type} & \boxed{1} \\
\textsc{sign}\,|\,\textsc{string}\,|\,\textsc{first} & \left\langle \boxed{2} \mid \_ \right\rangle
\end{array}\right],
$$

USR, [TYPE=$\boxed{1}$] , $\boxed{2}$).

Figure 5: *The general tsls rule doing the conversion for all* USR *tags*

shoud here notice that the list of feature-value descriptions associated with the `<USR>` tag is not empty, in contrast to the corresponding list associated with the `<W>` tag in case of the default application of tsls rules (see figure 2). And we can structure-share information described in this list with the value of some features of the PLS. So the value of the text feature TYPE will be co-shared with the one of feature USR_TYPE of the linguistic feature *ld*. This is the way the system is getting informed that the corresponding generic entry should be accessed. The generic entry in the linguistic entry will also receive more “flesh” in the sense that the STRING value of the textual representation will be co-shared with the value of the feature FIRST, which is underspecified in the ALEP generic entry.

2.5 Coverage and efficiency

The integration of user-defined tags during preprocessing and the mapping of this information onto partial linguistic structures via the TH-LS interface had remarkabale effects on both coverage and efficiency of LS-GRAM grammars. In case of coverage, we could just deal with a large set of phenomena which are normally not covered by competence (unification-based) grammars, which also mostly are not directly concerned with this kind of expressions. The efficiency could also be improved, since multiple words have been concatenated into one string, the processing of which is evidently faster as the processing of the original string. This was particularily true in case of the processing of some appositive structures. So for example in the case of apposition of proper names, the system has to process only one string and can also get rid of complex apposition rules defined in the grammar and which are very expensive in term of parse time.

3 Integration of external taggers

Once (positive) experiences have been made with the integration of small user-defined taggers, it seemed to be straightforward to try to integrate information resulting from the processing of the input texts by an independent (and real) tagger. We decided to base this experiment on the `mpro` tool, which is briefly presented in the next section[7]. For our work, we used this tool as a black box, being here interested only in its output. So we have not been involved in the development of the tool and were not directly interested in the way this tool works. We just assumed that the results provided are correct – for the sake of the experiment. For a real application, an evaluation of the performances of the tool is necessary, to see to which extent the tool is working properly and is delivering valuable results. At this place we are only interested in seeing how the results of a PoS tagger can be integrated into a grammar development platform having has a kernel a unification-based processing engine (ALEP), thus extending the field of application of the TH-LS interface to syntactic information.

3.1 The Part-of-Speech tagger – MPRO

The `mpro` tool can deal with free input texts. Looks up in lexicons, which are continually updated, a complete morphology and distinct heuristics allow to provide with an accurate annotation of all items of the input string. For the text tagging, morphological analysis is combined with a look-up in a word-form dictionary especially designed for this purpose. Nouns, adjectives, verbs and derived adverbs are recognized by morphological analysis, and all other words must be listed in the tagging dictionary. The morpheme lexicon of the morphological analyzer contains approx. 30 000 lexemes (36 500 allomorphs). After distinct steps of disambiguations a rough syntactic analysis is provided, which also contributes to the disambiguation of the input. The output of `mpro` is done in the form of lists of feature structure, a fact which facilitates the conversion into the format required by ALEP for the input of the TH-LS component. As an example, the first lines of the output of the `mpro` analysis for the following sentence:

```
Grosse Bereiche der Dasa leiden unter dem R\"uckgang des einst
```

[7] For a detailed description of the tool, see Maas (1996) and Maas and Hirschfeld (1996)

```
lukrativen R\"ustungsgesch\"afts.
(Large areas of the DASA are suffering from the decline of the once
lucrative arms trade.):
```

```
<1> to <14>:
c=satz,r=339}
  {c=np,r=302,tr=genattr}
     {c=np,r=380}
        {ori=Grosse,wnra=1,wnrr=1,snr=1,pctr=no,pctl=no,last=no,gra=cap,c=adj,lu=gross,
         endung=e,deg=base,t=gross,cs=a,ts=gross,ds=gross,ss=a,w=1}
        {ori=Bereiche,wnra=2,wnrr=2,snr=1,pctr=no,pctl=no,last=no,gra=cap,c=noun,nb=plu,
         case=nom;gen;acc,lu=bereich,g=m,s=domain,t=bereich,cs=n,ts=bereich,ds=bereich,
         ss=domain,w=1}
        {c=np,r=212,tr=d_n,nb=sg,g=f,case=gen}
           {ori=der,wnra=3,wnrr=3,snr=1,string=der,lu=d-,c=w,sc=art,fu=def;np,p=dE:r,
            s_flex=weak,spec=def,pctr=no,last=no,pctl=no,gra=small,case=gen,nb=sg,g=f}
           {ori=DASA,wnra=4,wnrr=4,snr=1,pctr=no,pctl=no,last=no,gra=caps,c=noun,lu=dasa,
            s=coll&isto,t=dasa,cs=n,ts=dasa,ds=dasa,ss=coll&isto,w=1,case=gen,nb=sg,g=f}
...
```

From this output we just automatically extract the information which is relevant for our purpose:

```
STRING,CAT,STEM
Grosse,a,gross
Bereiche,n,bereich
...
```

And this information is added to the default output of the word recognizer of the TH component:

```
<W TYPE="WORD" CASE="FIRST" POS="a">grosse</W>
<W TYPE="WORD" CASE="FIRST" POS="n">bereiche</W>
...
```

The output of the input preprocessing is thus enriched with linguistic information.

3.2 Extending the tsls-rule with PoS information

In order to also map this information onto the PLS, an extension of the tsl rule for the word level items is necessary. In fact, the format remains identical and we just make use of the possibility to add elements into the list of feature-value descriptions associated with the `<W>` tag (remember: this list was empty in the case of the default mapping). The adapted tsls rule is presented in figure 6.

3.3 Efficiency

Sharing the value of both the `POS` feature of the text structure and of the feature CAT of the linguistic description is enough in order to allow the 'lifting' (= mapping from ts to ls) of the information delivered by the PoS tagger. The advantage of

$$
\text{ts\_ls\_rule}(\\
\left[\begin{array}{ll}
\textsc{spec}\,|\,\textsc{level\_spec}\,|\,\textsc{m\_w} & \text{yes} \\
\textsc{sign}\,|\,\ldots\,|\,\textsc{syn} & \left[\begin{array}{ll}
\textsc{constype} & \left[\begin{array}{ll}\textsc{min} & \text{yes} \\ \textsc{constr} & (\text{w\_form;lexical})\end{array}\right]_{\textit{phrasal}} \\
\textsc{category}\,|\,\textsc{cat} & \boxed{1}
\end{array}\right]_{\textit{syn}}
\end{array}\right]_{\textit{ld}}, \\
\text{W, [POS=}\boxed{1}\text{]).}
$$

Figure 6: *Value-sharing of a feature of the text structure tag* W *and the* CAT *feature of the linguistic description*

this strategy: before starting, the parser will process items already containing PoS information, which are typically a contextual information and normally not available to the parser.

This way, it was possible to considerably reduce the parsing time of the system. It should suffice to consider the following figures, comparing the parse time for the German sentence we introduced above: "Grosse Bereiche der Dasa leiden unter dem Rueckgang des einst lukrativen Ruestungsgeschaefts.". Two distinct parsers have been used, the "basic" and the "record" parser[8]. The results displayed are in term of CPU time:

```
without PoS information
basic parser:  40.550   record parser: 6.260

with PoS information:
basic parser:  13.880   record parser: 4.860
```

The improvement of performance is partly due to the fact that after the segmentation of the input words has been achieved, the grammar is concerned first with the reconstruction of the words, enriched with linguistic information contained in the morpheme lexicon of the grammar. For this process affixes have been described as the parsing heads. And affixes (in German) being highly ambiguous (for example the affix *en* of the word *leiden* above), there are several entries for some of them in the morpheme lexicon, implying that the process of word construction will run several times, also when not necessary. But once the PoS is known before the process of word construction is started, the homograph affixes not corresponding to the PoS won't be considered any longer, thus reducing the search space for the parser. This remark being also valid for ambiguities at the word level.

So our strategy will be profitable everywhere where the input words are built with ambiguous affixes. In any case, the basic parser will be (considerably) faster, whereas the record parser (which already shows very good performances) will be improved only in case we have really strong ambiguities at both morpheme and

[8]Both parsers are non-deterministic bottom-up head-out parsers, but the record is more efficient since it subsumes the common information of possible alternatives.

word level.

Similar results have been obtained for other sentences: so for example (1) "In den Wochen vor Weihnachten konnte der stolze Vorsitzende der DASA ein gutes Jahresergebnis verkünden." and (2) "Das Luft- und Raumfahrtunternehmen hat das Jahr 1991 mit einem Gewinn von vierhundert Millionen Mark wiederholt." (this sentence gets more than one syntactic analysis, due to pp attachment, which resolution is postponed to the refinement processing step.). Both sentences have been tested on a slower machine and the results are:

```
without PoS information
basic parser:  91.201 (1)   record parser: 17.384  (1)
              106.583 (2)                  22.049  (2)
with PoS information:
basic parser:  43066  (1)   record parser: 17.384  (1)
               65.733 (2)                  21.936  (2)
```

We can observe here again the substantial improvement of the performance of the basic parser. We observe also that the record parser remains, for those examples, insensitive to the integration of PoS. We still have to investigate more precisely in which cases the performances of the record parser can be improved.

3.4 Extending the strategy

The strategy described above is getting extented to the integration of the full information delivered by external morphological tools and taggers. In the case of the integration of morphological information (both flexional and derivational) we will be able to plug out the Two-Level-Morphology (TLM) component of ALEP (which is not very efficient[9]) and to replace it with more performant and more informative tools. In case of the integration of `mpro` the system will also have other benefits, because we can get rid of the whole word grammar described in the actual. There we expect another substantial gain of efficiency.

3.5 Remaining problems

At the time being it is still not possible to directly map every kind of information from the text structure to the linguistic structure. This is due to the fact that value-sharing between those two levels of representation is allowed only for atom-valued features of the LS. This problem has been described in Declerck and Heyd (1997), where the alternatives of a direct mapping are described. But we also advocate for a modification of the TH-LS implementation in order to keep this interface as flexible and transparent as it is for the case of atom-valued linguistic features.

[9]On the limitations of the TLM algorithm instantiated in ALEP, the reader can consult Badia and Tuells (1997) and Theofilidis (1997b)

4 Conclusions

We showed how various instantiations of the TH-LS interface of the ALEP-platform contributed to both extension of coverage and improvement of performances of grammars written within the framework of an unification-based formalism. We also showed that the TH processing level of ALEP is the adequate place in order to integrate various kind of information delivered by Language Engineering (LE) tools. We agree on this with Cunningham et al. (1997), that commonality of information – necessary in order to ensure a smooth integration – is best reached at a lower level of processing, in our case at the text preprocessing level of ALEP, the TH subsystem (taking also in consideration that the text representation is based on the SGML standard).

I also understand our work as an implicit contribution to a discussion on hybrid approach to NLP. The flexible and transparent TH-LS interface of ALEP seems to provide an adequate tool for the task of integration of various LE tools into a so-called high-level unification-based lingusitic description. Work will be pursued in order to investigate the relationship between text structure and linguistic structure in the opposite direction, and thus making available high-level linguistic descriptions for the purposes of low-level text processing.

References

Adriaens, G. (1996). SECC: Using text structure information to improve checker quality and coverage. In *Proceedings of the 1st International Workshop on Controlled Language Applications*, Leuven, pp. 226–232.

Alshawi, H., D. J. Arnold, R. Backofen, D. M. Carter, J. Lindop, K. Netter, S. Pulman, J. Tsuji, and H. Uszkoreit (1991). Eurotra et6/1: Rule formalism and virtual machine design study. Technical report, CEC.

Badia, T. and A. Tuells (1997). SEGMORPH: An extension of the Alep morphographemic segmentation formalism. In *Proceedings of the 3rd ALEP User Group Workshop*, Saarbrücken, pp. 1–8.

BIM/SEMA (1993). *ALEP System Documentation.* Luxembourg: CEC.

Block, H. U. and L. Humphreys (1996). LS-GRAM final review report. Technical report, CEC, Luxembourg. Available at http://www.iai.uni-sb.de/LS-GRAM/papers.html#grammar-doc.

Bredenkamp, A., T. Declerck, F. Fouvry, and B. Music (1996). Efficient integrated tagging of word constructs. In *Proceedings of the 16th International Conference on Computational Linguistics, COLING-96*, Volume 2, Copenhagen, pp. 1028–1031.

Cunningham, H., K. Humphreys, R. Gaizauskas, and Y. Wilks (1997). Software infrastructure for natural language processing. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington.

Declerck, T. (1996). Dealing with cross-sentential anaphora resolution in ALEP. In *Proceedings of the 16th International Conference on Computational Linguistics, COLING-96*, Volume 1, Copenhagen, pp. 280–286.

Declerck, T. and S. Heyd (1997). Modifying the internal representation of the th-ls component of ALEP? In *Proceedings of the 3rd ALEP User Group Workshop*, Saarbrücken, pp. 14–18.

Declerck, T., S. Rieder, P. Schmidt, and A. Theofilidis (1995). Integration grammatischer Komponenten und Organisation des Lexikons in ALEP. In *Proceedings of the DGfS/CL*, Düsseldorf.

Maas, H. D. (1996). MPRO – Ein System zur Analyse und Synthese deutscher Wörter. In R. Hausser (Ed.), *Linguistische Verifikation, Sprache und Information.* Tübingen: Max Niemeyer Verlag.

Maas, H. D. and D. Hirschfeld (1996). Improving the functionality of a text-to-speech system by adding morphological knowledge. In G. Görz and S. Hoelldoblerund (Eds.), *KI-96: Advances in Artificial Intelligence*, Number 1137 in Lecture Notes in Artificial Intelligence. Springer.

Music, B. (1995). Tagging messy details in ALEP. Technical report, Center for Sprogteknologi, Copenhagen. LS-GRAM (LRE 61029) Supplement to Deliverable E-D5-DK.

Schmidt, P., A. Theofilidis, S. Rieder, and T. Declerck (1996). Lean formalisms, linguistic theory and applications. Grammar development in alep. In *Proceedings of the 16th International Conference on Computational Linguistics, COLING-96*, Volume 1, Copenhagen, pp. 286–291.

Simpkins, N., M. Groenendijk, and G. Cruickshank (1993). *ALEP–1 User Guide*. Luxembourg: CEC.

Theofilidis, A. (1997a). ALEP-based distributed grammar engineering. manuscript.

Theofilidis, A. (1997b). Efficiency of morphological analysis in ALEP. In *Proceedings of the 3rd ALEP User Group Workshop*, Saarbrücken, pp. 93–96.