Investigation on the Reusability of LFG-based Grammar Resources

Thierry Declerck

German Research Center for Artificial Intelligence (DFKI)

Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany

Proceedings of the LFG97 Conference

University of California, San Diego

Miriam Butt and Tracy Holloway King (Editors)

1997

CSLI Publications

http://www-csli.stanford.edu/publications

1 Background

Within the context of the LS-GRAM project[1] an investigation has been done on the reusability of LFG-based grammar resources for the ALEP framework. The LFG-grammar for German taken as the starting point of this investigation has been developed at the *Institut für maschinelle Sprachverarbeitung* (IMS) at the University of Stuttgart (see [Berman 1995]) and implemented within the Grammar Work Bench of Xerox (GWB)[2]. Our work was also based on the results of a study on the (non-)equivalences of the distinct formal devices implemented in ALEP and other unification-based formalisms (see [ET-10/52-93]). But our investigation was rather concerned with a real experiment, achieving an output equivalence between the source LFG-grammar and the target ALEP-grammar.

In this paper I will show how it has been taken advantage of some structural identities of LFG and ALEP (both being a lexical-driven formalism combined with an obligatory context-free backbone) in order to migrate the lexicon and structure rules. More work was needed in order to migrate the feature descriptions of the LFG grammar into ALEP (being a typed formalism allowing a simple hierarchy). In this case, one has to consider also the *completeness* and *coherence conditions* put on the feature structures of LFG and the distinct kinds of (constraining) equations defined in the formalism. This information is enough for the definition of the type system of the ALEP grammar. ALEP being a so-called "lean" formalism[3], it was not possible to straightforward reproduce the compact LFG-formulation of rules and lexicon entries.

I will also present some work recently done within the ALEP platform which led us into the field of grammar engineering. This work is dedicated to the integration of an external morphology and part-of-speech information provided by preprocessing tools and taggers[4] into ALEP. These experiments have led to a reformulation of the linguistic resources defined within the ALEP grammars, allowing a very compact formulation (for example generic lexicon entries and reduction of the set of syntactic rules), which should facilitate the migration of ALEP resources to grammar formalisms supporting richer formal devices. The question which will remain: how can this approach been made available for other platforms supporting unification-based grammars, like for example the GWB or the XLE platform[5].

2 Migrating LFG Resources into the ALEP Formalism

The migration experiment has been carried on with the purpose of obtaining grammar resources for the development of a German grammar within the ALEP platform, avoiding thus to start writing the grammar from scratch. I considered only a subpart (the nominal syntax) of the German LFG grammar developed at the IMS, since other partners of the LS-GRAM project were working with a similar task, accessing other grammar resources from other platforms.

[1] The LS-GRAM (Large-Scale GRAMmars for EC languages) project was funded by the CEC under the number LRE 61029.

[2] See the documentation of this system, available by anonymous ftp: parcftp.xerox.com in the directory /pub/nl. This is a Postscript file, and can be sent directly to most printers. The file is: lfgmanual.ps.

[3] See also [Schmidt et al. (1996)]

[4] See [Bredenkamp et al. (1996)] and [Declerck and Maas (1997)]

[5] The XLE (Xerox Linguistic Environment) platform is a recent development of Xerox for the writing and processing of LFG grammars. This platform is more adequate for linguistic engineering as its predecessor, the GWB platform. See [Kaplan 1997].

2.1 The Source Grammar

I concentrated on the nominal syntax of the German LFG grammar developed at the IMS in Stuttgart. This syntax has been formulated within the paradigm of the DP analysis (for the motivation, see [Berman 1995]). The descriptions which have been migrated included full-form lexicon entries, structural rules and templates.

2.2 The Target Grammar

The target grammar was formulated in the ALEP formalism, a lean type-based unification formalism, allowing for a simple inheritance mechanism. The lexicon has been integrated into a TLM (Two-Level-Morphology) component, and we have structural rules, macros and typed feature structures.

2.3 Similarities between LFG and ALEP

The migration experiment was facilitated due to the fact that the formalisms share important common features. They are both unification-based and support a lexical approach combined with an obligatory context-free component.

2.4 Distinctions between LFG and ALEP

But ALEP being a so-called “lean” formalism, it is not possible to reproduce in a straightforward manner the compact formulation of LFG descriptions. And also the ALEP formalism is typed and includes an obligatory declaration component. ALEP allows a single inheritance between type descriptions.

2.5 The Type of Migration

The migration has been done manually, but a semi-automatically migration between LFG and ALEP seems to be feasible (see the algorithm described by Dieter Kohl and Stefan Momma in the ET 10/52 study, which was mainly based on the Charon-System developed at the IMS in Stuttgart, [ET-10/52-93]). The decision to make this migration in a manual fashion ha been dictated by two factors: the GWB platform was not transparent enough for this kind of work, and we also needed the migrated grammar in a very short time. So there was no time left for describing migration algorithms between the GWB and the ALEP platforms[6].

The migration has been done in two steps:

– porting of rules and migration of feature names onto ALEP type descriptions,

– precising the type system of ALEP, taking into account the filtering *conditions* of LFG and the kind of *constraints* used.

The work done so far has achieved an *output equivalence* between both grammars, having also an identical coverage of linguistic phenomena. This results have been obtained in a very short time (two days), taking into account that the person performing this migration had also to be introduced to the ALEP platform.

3 Migrated Resources

In the following sections, I just show some of the results of the migration. Due to limitations of space, I don't reproduce here the original LFG lexicon entries, rules and templates. The reader is

[6]This should however be done in connection with the XLE platform.

referred to [Berman 1995].

3.1 Lexicon Entries

Two examples are displayed here, one for a determiner and one for a substantive:

```
die ~
   sign:{
     syn => d_syntax:{
              cat => det,
              agr => ((nom;acc)&((fem&sg);pl)&weak),
              fcomp => [sign:{
                          syn => n_syntax:{
                                   cat => np},
                          sem => SEM} ],
              ehf => no},
     sem => d_semantic:{
              class => def,
              pred => def,
              arg => fcomp},
     string => [die|R], rest=>R }.
```

As in the LFG source grammar, the determiner is considered as a syntactical head (at least for the first step of this migration experiment, see below the concluding remarks). One can recognize this at the feature 'fcomp', which has a subcategorization list. New in comparison with the source grammar is the 'sign' organization, i.e. the use of a hierarchical type system. The type system, which has been build stepwise for the purpose of this experiment is displayed below. Also new is the treatment of agreement, having as value a boolean type ('&' = AND, ';' = OR, '∼' = NOT), that are defined within ALEP. The semantic features here are just collecting information which in the source grammar is not explicitly specified as such (there was not *s*-projection in the source LFG grammar).

In the case of substantives, the same kind of remarks is valid:

```
maerkte ~
   sign:{
     syn => n_syntax:{
              cat => noun,
              agr => (mas&pl&(~dat))},
     sem => n_semantic:{
              class => count,
              pred => markt},
      string => [maerkte|R], rest=>R }.
```

Once the migration into a type system has been done, there is no problem in migrating this kind of lexicon entries.

3.2 Structure Rules

In the next section, we can see the context free backbone of ALEP. The formulation of the rules is straightforward. But it was not possible to preserve the compact formulation of the rules of the source grammar, since optionality, for example, had to be spelled out in different rules. The number of rules in the target grammar increased a lot. A way to keep the rule system compact can be found with the use of macros (see the documentation of the final version of the core grammar [DC-WP6e], available under http://www.iai.uni-sb.de/LS-GRAM/papers.html). Or to use boolean feature types. In both cases, the migration has to be done in two steps: one step for the (straight-forward) porting of the rules, and one step for the transformation of LFG specific expressions (including the annotations) into ALEP constructs. In the next rules, the mother-daughter relation is represented through a "<". The daughter are collected in a list (see the square brackets).

As an example, the following rule (D' $\rightarrow$ D, NP) which seems to be complex, but this is due only to the immediate translation of the LFG rule into ALEP. For this reason, one should make an intensive use of macros for her grammar design in ALEP. With the use of the variables it is possible to reproduce the structure sharings formulated in the source grammar. The feature distribution corresponds to the one resulting from the 'ups' and 'downs' arrows in LFG.

```
sign:{
  syn => n_syntax:{
           cat => np,
           agr=>A},
  sem => N_SEM,
  string=>S,
  rest=>R } <
 [ sign:{
     syn => a_syntax:{
              cat => ap,
              agr=>A},
     sem => A_SEM,
     string=>S,
     rest=>S1 },
   sign:{
     syn => n_syntax:{
              cat => np,
              agr=>A},
     sem => N_SEM,
     string=>S1,
     rest=>R } ].
```

One can also see how the the names of the features of the source grammar have been introduced into a type system.

4 The resulting Type System

The resulting type system from the migration is here only partly displayed, and will not been commented. It should be enough to say, that I also recurred to the use of templates in the LFG source grammar in order to design the type system. The reader will see that features may have distinct types of values. This is also something which can complicate the migration, since in order to describe an efficient ALEP grammar, one has to make choices upon the various types of values allowed, in dependence of the phenomenon he/she wants to account for. This diversity of kind of types is not present in the LFG source grammar, which thus gives no hint for a decision concerning the type of values to be (optimally) chosen. The inheritance relations are encoded with the > sign. The inheritance relation is also something which is to be designed in a second step, once the LFG source grammar has been migrated:

```
type( sign:{    syn     => type({ syntax:{} }),
                sem     => type({ semantic:{} }),
                spec    => type({ specifier:{} }),
                refined => atom({ y,n }),
                string  => list,
                rest    => list
        },
        ''
).
syntax >
   {n_syntax,
    d_syntax,
    a_syntax > {st_a_syntax,
                a_nom_syntax},
    st_syntax,
    p_syntax > {p_obj_syntax,
                p_adj_syntax},
    t_syntax
   }.

type( syntax:{  agr     => boolean([{nom,acc,dat,gen},{fem,mas,neut},
                           {sg,pl},{p1,p2,p3},{weak,strong}]),
                gaps    => type({ gap_lists:{} })
               },
               ''
).


type( d_syntax:{ cat    => atom({ det, d1, dp, e}),
                 fcomp  => list(type({ sign:{} })),
                 ehf    => atom({ yes, no})
                },
               ''
).

type( n_syntax:{ cat => atom({ noun, n_app, n1, n2, np })
                },
               ''
).

type( p_syntax:{ cat => atom({ p, pp }),
                 pdet => atom({ yes, no })
                },
               ''
).

type( p_obj_syntax:{ obj => list(type({ sign:{} })),
                     pcase => atom
                },
               ''
).

....
```

5 Concluding Remarks concerning the Migration

We have seen that if one concentrates on the similarities between LFG and ALEP formalisms, a straightforward migration is partly possible. My goal was only to achieve an output equivalence between the two grammars, just trying to keep the grammatical descriptions in both formalisms as identical as possible. This was possible, because I didn't have to migrate a lot of metalinguistical descriptions or tools. The migration has mainly be done by just porting the feature names (modifying them only if needed for transparency) and transforming them into a type description. The migration of the rules can be very straightforward, but one has to think about the feature descriptions and the type hierarchy, which are, in this explicit form, not existent in LFG. Those remarks are also valid for the migration of the lexicon. We also saw the need for a consequent use of macros in ALEP in order to keep the grammar compact and readable. The resulting ALEP grammar had to be modified later, since the DP analysis was not the one adopted for the ALEP grammar, which avoids empty categories. But the type system resulting from the migration had to be just minimally modified, which is in the context of a type-based formalism very important, since a significant time of the grammar development is concerned with the definition of the declaration component.

6 A Migration between LE Platforms

6.1 A Reverse Migration?

The interest for a reverse migration should be placed not on a migration between formalisms, but on a migration taking into consideration linguistic engineering aspects. The fact is that the LFG community disposes of enough German LFG grammars, and so the situation we had for ALEP is not present: for the German ALEP grammar we wanted to reuse as much as possible already existent grammar resources. But on the other side, it would be interesting to see how some work on linguistic engineering done within the context of the LS-GRAM project could be reused within a LFG grammar development environment. This is actual since the step done from the GWB to the XLE platform definitely permits to place the task of further developing a LFG grammar into the context of the broader aspect of linguistic engineering. But for sure, I didn't have time to experiment anything for this. So I will just shortly present some results of the LS-GRAM project, wondering what can be done in order to achieve a general strategy for the integration of linguistic engineering tools into unification-based grammar development environments.

6.2 The Integration of LE Tools into the ALEP Platform

The ALEP platform provides for a Text Handling component, concerned with the SGML marking of input texts. After this pre-processing stage, the input for the parser is (simplified) as follows:

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

But before being processing by the parser an intermediate step is defined, consisting in specialized mapping rules which associate the SGML-marked texts with linguistic descriptions formulated by the grammar writers. The importance of this mechanism for an efficient processing of grammar descriptions in ALEP has been commented a.o. in [Declerck and Maas (1997)] and [Theofilidis 1997]. For the default case, this kind of rules – the so-called tsls-rules (Text Structures to Linguistic Structures) – can have the form displayed in figure 1, where a correspondence is established between

$$
\begin{array}{l}
\text{ts\_ls\_rule(} \\
{}_{ld}\left[\begin{array}{ll}
\textsc{spec} \mid \textsc{level\_spec} \mid \textsc{m\_w} & \text{yes} \\
\textsc{sign} \mid \ldots \mid \textsc{syn} \mid \textsc{constype} & {}_{\mathit{phrasal}}\left[\begin{array}{ll} \textsc{min} & \text{yes} \\ \textsc{constr} & \text{(w\_form;lexical)} \end{array}\right]
\end{array}\right], \\
\texttt{W}, [\ ]\ ).
\end{array}
$$

Figure 1: *Correspondence established between the partial linguistic description and the* `<W>` *markup of text*

a certain text structure (the words) and a class of linguistic descriptions (the type *ld*). It is also possible to enrich the list of features associated with the SGML tag (here the `<W>` markup), which in figure 1 is empty. So for example we wrote small taggers for the recognition of *messy details* and *fixed phrases*. On the base of the output of those taggers, some enriched tsls-rules can be described, so for example if the tagger recognizes and marks currency expressions:

```
<W TYPE="CURRENCY_MEASURE" ORIG="Dreiundvierzig Millionen Dollar">
  Dreiundvierzig_Millionen_Dollar</W>
```

The corresponding tsls-rule will be like shown in figure 2, where a value-sharing between a text

$$
\begin{array}{l}
\text{ts\_ls\_rule(} \\
{}_{ld}\left[\begin{array}{ll}
\textsc{spec} \mid \textsc{usr\_type} & \boxed{1} \\
\textsc{sign} \mid \textsc{string} \mid \textsc{first} & \left\langle \boxed{2} \mid \_ \right\rangle
\end{array}\right], \\
\texttt{W}, [\ \texttt{TYPE=}\boxed{1}\ ]\ , \boxed{2}\ ).
\end{array}
$$

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

feature and a feature of the linguistic description is defined. The lexicon entry referred to by the linguistic description has the particularity that it won't be accessed by its realization, but by the *class* of words it belongs too (see the value-sharings). This is possible because ALEP supports the definition of generic lexicon entries, as shown in figure 3. Generic entries allow to include in the ALEP lexicon in a very compact manner classes of expressions which usually are causing serious problems to the coverage of the grammar and to the performances of the parser. This strategy

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

Figure 3: *Generic lexical entry for currency expressions*

has been extended to the output of a PoS-tagger. Here again, the information delivered by an external tool has been integrated into the grammar processing of ALEP via some tsls-rules. So if, for example, following PoS information is delivered by a tagger[7],

[7]This information is currently extracted from the results of the analysis of the MPRO tool, see [Declerck and Maas (1997)]

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

this information can be integrated into the grammar processing via the tsls-rule displayed in figure 4.

$$
\begin{array}{l}
\text{ts\_ls\_rule(} \\
\left[\begin{array}{ll}
\text{SPEC} \mid \text{LEVEL\_SPEC} \mid \text{M\_W} & \text{yes} \\
\text{SIGN} \mid \ldots \mid \text{SYN} & \left[\begin{array}{ll}
\text{CONSTYPE} & \left[\begin{array}{ll} \text{MIN} & \text{yes} \\ \text{CONSTR} & \text{(w\_form;lexical)} \end{array}\right]_{\textit{phrasal}} \\
\text{CATEGORY} \mid \text{CAT} & \boxed{1}
\end{array}\right]_{\textit{syn}}
\end{array}\right]_{\textit{ld}}, \\
\texttt{W, [POS=}\boxed{1}\texttt{]).}
\end{array}
$$

Figure 4: *Value-sharing of a feature of the text structure tag* <W> *and the* CAT *feature of the linguistic description*

The importance of this addition of information can be showed at the improved processing times of the parsers of the ALEP platform. For the sentence "Große Bereiche der Dasa leiden unter dem Rückgang des einst lukrativen Rüstungsgeschäfts." (Large areas of the DASA are suffering from the decline of the once lucrative arms trade.) the parsing time was respectively 40.550 and 6.260 CPU for the basic and the record parser of ALEP without the integration of PoS information and 13.820 and 4.850 *with* the integration of such information, a lot of information being instantiated before the parsers start their job[8]. 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 word level.

We also expect another important improvement of performance, once a fast external morphological analysis will been integrated, delivering thus the parser of ALEP of the task of building word units out of the results of the actual Two-Level component of ALEP.

7 Concluding Remarks and Future Work

Insisting more on the linguistic engineering aspects of grammar development in the second half of this paper, I presented some works done within the context of the LS-GRAM project in connection with the ALEP platform. There I showed how the integration of external linguistic engineering tools

[8]For sure, one has to ensure correctness of the PoS-tagger, or at least some control of the output of the tagger, an area on which I am working for the time being.

(like document processing, morphological analysis, part-of-speech taggers) can have a significant influence on coverage and performance of ALEP grammars. Also the design of the grammars has to be reconsidered in the light of this kind of operation. From this we might be able to follow that generally the linguistic descriptions of unification-based grammars have to be adapted for the sake of efficiency (in the setting of a possible industrial application). The migration between formalisms could be easier, since it will apply only to this simplified linguistic descriptions (generic entries, reduced set of rules and templates, etc.). In a future work, it will be investigated if the design of a general interface between document processing and linguistic description is possible for other grammar development platforms working within the paradigm of unification-based formalisms. This particularly for the XLE environment platform. At the end of this experiment about the reusability of LFG-resources, we are concerned with the more general question about the reusability of linguistic engineering resources for unification-based grammars.

References

[Berman 1995] Judith Berman and Anette Franck, *Deutsche und Französiche Syntax im Formalismus der LFG*, Niemeyer,1995.

[Bredenkamp et al. (1996)] Bredenkamp, A., Declerck, T., Fouvry, F., Music, B. 1996. *Efficient Integrated Tagging of Word Constructs.* COLING 96. pp. 1028–1031.

[Bresnan 1982] Joan Bresnan. 1982. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge Massachusets.

[DC-WP6e] Schmidt P., Rieder S., Theofilidis A., Declerck T., 1996, *Final Documentation of the German LS-GRAM Lingware*, LRE 61029, Deliverable DC-WP6e, final report, CEC.

[Declerck (1996)] Declerck Thierry. 1996. *Dealing with Cross-Sentential Anaphora Resolution inALEP.* COLING 96. pp. 280–285.

[Declerck and Maas (1997)] Declerck Thierry, Maas Heinz-Dieter. 1997. *The Integration of a Part-of-Speech Tagger into the ALEP Platform.* In Proceedings of the 3rd AUG Workshop.

[ET-10/52-93] Douglas Arnold, Josef van Genabith, Dieter Kohl, Stella Markantonatou, Stefan Momma, Steven Pullman, Louisa Sadler and Paul Schmidt. 1993. *Reusability of Grammatical Resources and Grammar Development in ALEP*, ET-10/52 Final Report, CEC.

[Kaplan 1989] Kaplan, Ronald M. The Formal Architecture of Lexical-Functional Grammar. *Proceedings of ROCLING II, Taipei, Republic of China, 1989.* Reprinted in: *Journal of Information Science and Engineering* 5, 305-322, 1989.

[Kaplan 1997] Kaplan, Ronald M. & Paula S. Newman. Lexical Resource Reconciliation in the Xerox Linguistic Enironment. *Proceedings of the Workshop Computational Environments for Grammar Development and Linguistic Engineering, Madrid*, 54-61, 1997.

[Poll & Sag 1994] Carl Pollard & Ivan Sag, (1992). *Head Driven Phrase Structure Grammar*, Chicago University Press.

[Simpkins (1993)] N Simpkins, M Groenendijk, G Cruickshank 1993. *ALEP-1 Version 1.1 User Guide. Chapter 19: ET/-6/1 Based Linguistic Formalisms*, CEC 1993.

[Schmidt et al. (1996)] Schmidt Paul, Rieder Sibylle, Theofilidis Axel, Declerck Thierry. 1996. *Lean formalisms, linguistic Theory and Applications. Grammar Development in ALEP.* COLING 96. pp. 286–291.

[Theofilidis 1997] Theofilidis Axel & Paul Schmidt. Distributed Grammar Engineering in ALEP. *Proceedings of the Workshop Computational Environments for Grammar Development and Linguistic Engineering, Madrid*, 54-61, 1997.