The Loom-LAG for syntax analysis

Adding a language-independent level to LAG

Markus Schulze
University of Erlangen-Nürnberg
Department of Computational Linguistics
Bismarckstr. 6
91054 Erlangen
Germany
schulze@linguistik.uni-erlangen.de

Abstract

The left-associative grammar model (LAG) has been applied successfully to the morphologic and syntactic analysis of various european and asian languages. The algebraic definition of the LAG is very well suited for the application to natural language processing as it inherently obeys de Saussure's second law (de Saussure, 1913, p. 103) on the linear nature of language, which phrase-structure grammar (PSG) and categorial grammar (CG) do not.
This paper describes the so-called Loom-LAGs (LLAG) —a specialisation of LAGs for the analysis of natural language. Whereas the only means of language-independent abstraction in ordinary LAG is the principle of possible continuations, LLAGs introduce a set of more detailed language-independent generalisations that form the so-called loom of a Loom-LAG. Every LLAG uses the very same loom and adds the language-specific information in the form of a declarative description of the language —much like an ancient mechanised Jacquard-loom would take a program-card providing the specific pattern for the cloth to be woven. The linguistic information is formulated declaratively in so-called syntax plans that describe the sequential structure of clauses and phrases.
This approach introduces the explicit notion of phrases and sentence structure to LAG without violating de Saussure's second law and without leaving the ground of the original algebraic definition of LAG. LLAGs can in fact be shown to be just a notational variant of LAG —but one that is much better suited for the manual development of syntax grammars for the robust analysis of free texts.

1 Background – The Left-Associative Grammar Model (LAG)

As the reader may not be familiar with the model of left-associative grammar (LAG), this section starts with a brief introduction to LAG[1], which the reader familiar with LAG may well skip.

Most attempts at grammar acquisition and grammar coding are based on variants of PS-grammar. For reasons of complexity, these PS-grammars must be context-free (CF). It is widely agreed, however, that the class of context-free languages is not sufficient for the description of natural language. Hence these approaches suffer from an inherent empirical restriction.

The approach presented in this paper is based on the formalism of Left Associative Grammar (LAG, (Hausser, 1989)). While PS-grammar is based on the principle of possible substitutions, LA-grammar uses the principle of possible continuations. This difference results in a complexity hierarchy which is orthogonal to the well-known Chomsky hierarchy.

[1]For an in-depth discussion see (Hausser, 1989), and (Hausser, 2001).

In LA-grammars, the degree of complexity depends on the degree of ambiguity.[2] Of particular interest in the LA-hierarchy is the class of C1-languages, which parses in linear time and intersects with the class of CF- and CS-languages of the Chomsky hierarchy.[3]

The principle of possible continuations can be seen in the rule scheme of LAG:

$$\mathsf{r_i : cat_{ss} \times cat_{nw} \longrightarrow cat_{ss'}, rp_i}$$

An LAG-rule consists of a *categorial operation* and a *rule package*. The former checks the categories of the *sentence start* $\mathsf{cat_{ss}}$ and the *next word* $\mathsf{cat_{nw}}$ and derives the result category for the new sentence start $\mathsf{cat_{ss'}}$. The surface of the new sentence start is formed by the concatenation of the surface of the existing sentence start and the surface of the next word form. The rule package $\mathsf{rp_i}$ lists all rules applicable after the successful categorial operation of $\mathsf{r_i}$.

A complete LAG comprises combination rules as described above as well as a set of initial states and a set of final states. An initial state consists of an initial category and an initial rule package. A final state consists of a category(pattern) and rule package. The former of which specifies how a legal final category may look like if it has been produced by the rule which has the rule package listed as the latter.

Figure 1 presents a simplified analysis of the english sentence John gave Mary a book. The derivation order shown is strictly time-linear and deals with the traditional constituent structures implicitly.[4] The analysis is based on testing agreement and filling valence slots.

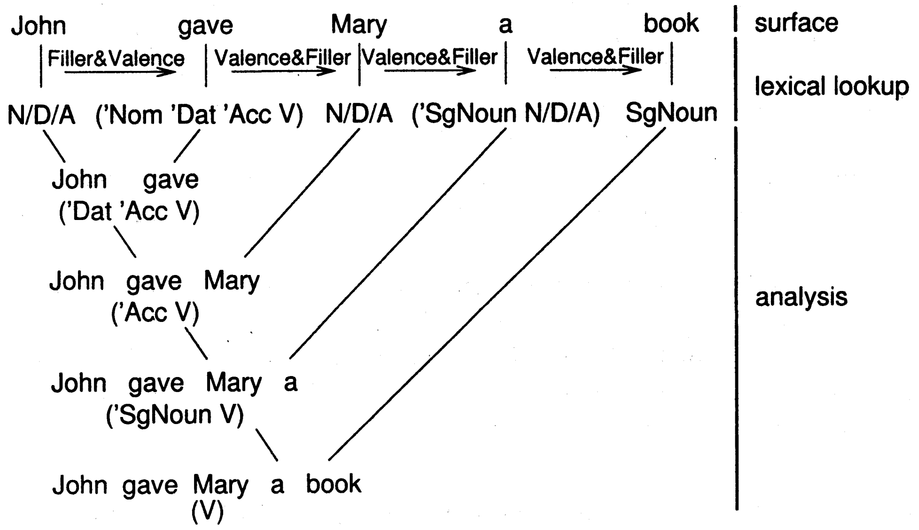

Figure 1: Simplified LA-analysis of an English sentence. The categories are simplified for the sake of clarity. Current implementations use categories with complex feature-value pairs. In addition, a structural representation of the sentence is being built in real implementations instead of just cancelling valence slots. In this figure, symbols with a preceding apostrophe represent valences, while symbols without apostrophes represent fillers or result categories.

Much like in Categorial Grammar (CG), the categories of an LAG contain an essential part of its combinatorial properties; e.g. information on valences or potential fillers or information

[2]For further details on complexity of LAGs, see (Hausser, 1992).

[3]Prominent examples of formal languages that can be parsed with C1-LAGs in linear time are $a^nb^nc^n$ or $a^{i!}$. Because C1-LAGs parse most context-free and many context-sensitive languages in linear time, it may be argued that the natural languages belong to this class.

[4]As a consequence the traditional linguistic notion of *phrases* (e.g. noun phrase, prepositional phrase etc.) is not directly represented in traditional LAG; a fact that has been criticized e.g. by (Sampson, 1999). The variant of LLAGs introduces the explicit modelling of phrases into LAG.

which attributes to check for agreement. While CGs contain only the standard canceling-rules, LAGs have a much more complex rule-system. Neither the number of rules nor their categorial operations are predefined in any way.

An LAG-parser acts as a simple *motor* of the grammar, merely applying the formal grammar mechanisms in the analysis of the input, as was originally intended in PS-grammar, (Berwick and Weinberg, 1984, p. 39). Consequently LAG achieves ***absolute type transparency*** as defined in (Berwick and Weinberg, 1984, p. 41).

In other words: The parser applies the rules of the grammar directly and in the same order as the grammatical derivation. Furthermore the input and output expressions of the parser and the grammar are identical in each rule application.

The LAG-framework has been applied to the morphologic, syntactic and semantic analysis of various languages (mainly German, Korean and English), e.g. in: (Leidner, 1998), (Lorenz, 1997) (Schulze, 1998), (Hong and Lee, 1999), (Lee, 1999) and (Chang et al., 2000).

2 New language-independent means of abstraction

The basic model of LAG has the principle of possible continuations as the only means of language-independent abstraction. Everything else is free to be handled in any way by any specific LAG.

There are, however, more principles that natural languages have in common. The approach of Loom-LAG (LLAG) aims at modelling language-independent properties within its loom-grammar that builds the groundwork of every LLAG.[5] The properties modelled by the loom of an LLAG are as follows:

Two syntactic levels: The approach of LLAG separates the modelling of syntax into two levels: The modelling of phrases and the modelling of sentence structure.

Basic structure of sentences: On the level of sentence structure there are so-called *fields*[6] and *delimiters*. Fields are spaces in which a specified number of phrases of specified types can occur —freely or in a given word order. The English *Object Field* is an example of a field: It begins after the finite verb or verbal complex of a main clause and can contain objects and adverbials in a given word order.

In contrast to fields, delimiters are always realised by exactly one token (word form or punctuation mark) and make up the structural part on the level of sentences. Examples for delimiters are punctuation marks, verbs or conjunctions.

Nested clauses: Beside the the top-level clause, there may be nested clauses occurring in fields.

Basic structure of phrases: Phrases may either be atomic —that is: consisting of one word form only— or they may be complex. The latter consist of an opening word form, optionally one or more continuing word forms and a closing word form.

Phrase-nested verbs: Verbs or other words with dependent phrases (complements and adverbials) may occur nested inside phrases. Examples for this are Korean adnominal verb forms and German participle groups.

3 The Loom-Grammar common to all LLAGs

The language-independent syntactic principles described in the preceding section are modelled in the loom-grammar, which is used for all LLAGs. The loom models the set of language-independent syntactic principles and is "linguistically empty" otherwise. Its rules don't provide

[5]The loom is only written once and then reused.

[6]The term *field* is coined after the German terms *Vorfeld, Mittelfeld* and *Nachfeld* which name examples for fields in German.

categorial operations themselves. Instead they import them from the plans, which form the language specific, declarative part of a LLAG. In a given LLAG, there are plans modelling the structure of its sentences and clauses and plans modelling phrases.

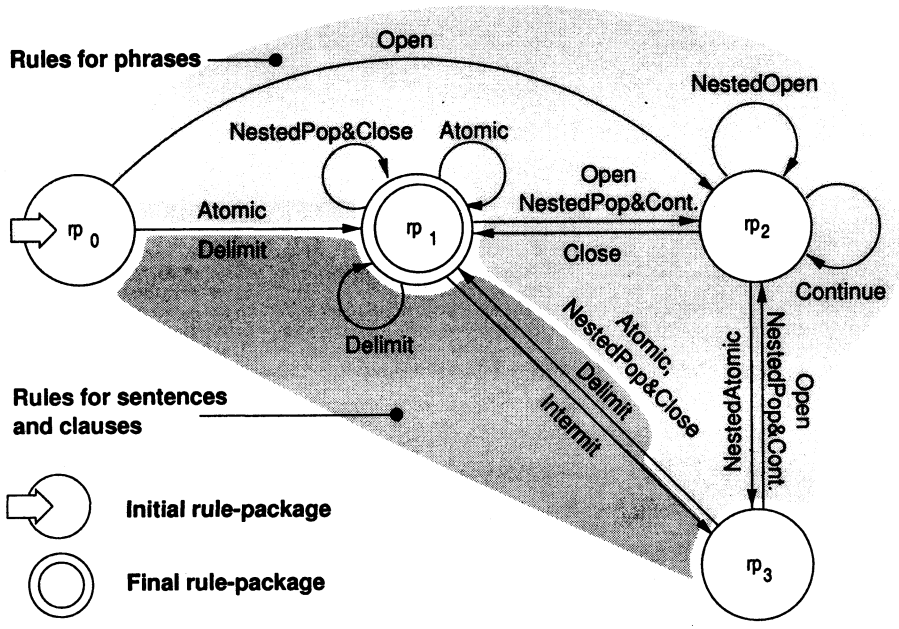

Figure 2: The loom-grammar common to all Loom-LAGs (LLAGs).

Apart from not containing categorial operations and importing them from plans instead, the language-independent loom-grammar is a conventional LAG. Figure 2 gives a graphic representation of the loom: The nodes represent its rule packages, while the edges represent its rules.

4 Syntax plans

In LLAG, the central means for the modelling of syntax are plans describing the sequential structure of sentences and subclauses as well as that of complex phrases.[7]

Plans are based on the LAG-principle of possible continuations. They specify how a sentence (or subclause or phrase) can start and how a given start can be continued and finally completed.

4.1 Sentence plans

Sentence plans are made up of a sequence of so-called *descriptors* for delimiters. Delimiters are atomic positions: They contain exactly one token (word form or punctuation mark) and make up the structure of a clause. Prominent examples of delimiters are verbs and punctuation marks. A descriptor contains the the categorial operation imported by the loom-rule `Delimit`. It models categorial restrictions that a word form has to fullfill in order to be read as the respective delimiter. A descriptor also names potentially following delimiters and describes the following field, if there is one. Fields model spaces in a sentence where phrases can occur. Number, type and order of phrases a field may contain are modelled within the descriptor of the delimiter that is preceding this field.

In addition to being associated to a preceding delimiter (as described above), fields may be part of the initial category (see section 5.2) of a sentence plan. The network constituted by the

[7]There are also atomic plans. Because they describe phrases described which consist of one word form only, these plans act more like a filter converting the morphosyntactic category of the input word form into the category of a phrase on the level of syntax.

initial states of a main-clause plan and its descriptors can be represented graphically, as shown in figure 3. The plan shown there models an English declarative main clause.

The initial node in the graph represents the initial subject field. The pointed brackets indicate that it is a field with fixed word order: it can take exactly one (superscript "1") noun phrase (NP) in nominative case (Nom). The only edge leading away from this node represents the delimiter holding the finite verb of the sentence, which in turn is followed by the object field. The pointed brackets indicate that this may take an optional (superscript "?") dative NP followed by another optional accusative NP.[8] After the object field the final delimiter holding a punctuation mark must follow.

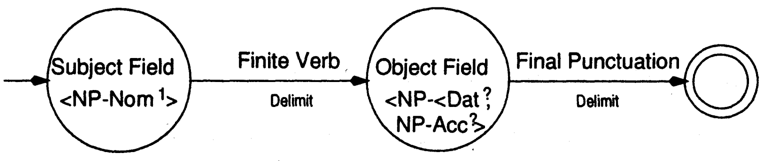

Figure 3: Schematic representation of a sample plan for an English declarative main clause: Nodes in the graph represent fields, edges represent delimiters.

Beside the sentence plans described above, there are subclause plans. These are structured the same way as sentence plans are, with one difference: They do not contain initial fields. They can only be started in a field of another plan (sentence plans and subclause plans) and only if this field specifies that it may contain a subclause of this type.[9]

The plans for phrase-nested clauses do not contain initial fields either. They can only be started inside a phrase plans when it specifies, that a phrase-nested clause of this type may occur there. For the sake of brevity neither subclause plans nor phrase-nested plans are explained in detail.

4.2 Phrase plans

Structured much like sentence plans, phrase plans are used to describe the sequential structure of complex phrases (e.g. noun phrases or prepositional phrases). They have three types of position descriptors: Opening, continuing and closing positions. An opening and closing position are mandatory for a phrase plan, while continuing positions are optional. A position descriptor specifies a set of word-classes it may hold. Descriptors also specify which attributes of the phrase start and the next word to check for agreement. Phrase plan descriptors are linked by specifying possibly succeeding positions.

Figure 4 shows the graphic representation of a sample plan for simple English noun phrases. The edges of the graph represent the position descriptors mentioned above. In this example, they specify the word-class of the word forms that may fill the respective positions as well as the attribute *number* that is to be checked for agreement on appending adjectives or the closing noun.[10]

Beside complex phrase plans, there are plans for so-called atomic phrases. These contain a single categorial operation that is imported by the loom-rule `Atomic` to read a single word form as a phrase.

[8]As the NPs that fill these places are actually underspecified in English, this specification of word order will result in a syntactic disambiguation of the case of the NPs in this field.

[9]From the perspective of the containing field and the respective sentence plan, subclauses are treated like phrases: They can be either be adverbial or a complement to the verb or verbal complex of the sentence or they can be an attached adjunct to a preceding phrase.

[10]In English, articles can carry a specified number. While adnominal adjectives do not carry inflectional markings for number, as substantives do, they can have a lexicalised semantic number attribute as in single or various.

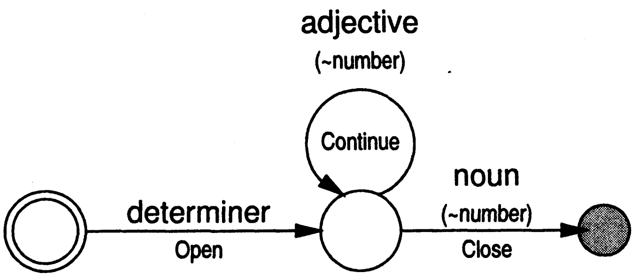

Figure 4: Schematic representation of a sample phrase plan for simple English noun phrases. The edges are labelled with the name of the loom-rule they are imported by.

5 Outline of the algebraic definition of LLAGs

(Hausser, 1999, sec. 10.2.1) defines LAGs as 7-tuples $(W, CLX, CO, RP, ST_S, ST_F)$, of word surfaces, category segments (symbols), lexicon, categorial operations, rule packages, initial states and final states.

The definitions of the first three components —set of word surfaces W, set of category symbols C and lexicon $LX \subset (W \times C^+)$— are used in the LLAG without any change.

The set of categorial operations CO and the set of rule packages RP ist structured differently in LLAGs. The rule packages of the language independent loom-grammar are not attached to fixed categorial operations. Instead the rules import the language-specific operations during the analysis as explained below.

The set of final states ST_S·and the set of final states ST_F are both defined by the the set of sentence plans of an LLAG, namely by their final and initial states.

5.1 Importing categorial operations

As described in section 1, the rule scheme of traditional LAG is as follows:

$$\mathsf{r_i : cat_{ss} \times cat_{nw} \longrightarrow cat_{ss'}, rp_i}$$

The categorial operation of a rule $\mathsf{r_i}$ combines the categories of the *sentence start* $\mathsf{cat_{ss}}$ and the *next word* $\mathsf{cat_{nw}}$ and derives the result category for the new sentence start $\mathsf{cat_{ss'}}$.

The rules of the loom-grammar common to all LLAGs do not contain categorial operations. They import them from the syntax plans instead. This is shown in the transformed rule scheme of LLAGs:

$$\begin{array}{lcccccl}
r_i : & [\quad\ ] & \times & [\quad\ ] & \longrightarrow & [\quad\ ] & , rp_i \\
 & \uparrow & & \uparrow & & \uparrow & \\
p_x : & (cat_{SA}, pp_y) & & cat_{NW} & & (cat_{SA'}, pp_x) & \\
p_x \in pp_y & & & & & &
\end{array}$$

r_i is a loom-rule which holds three empty spaces for the parts of the categorial operations checking the category of the sentence start and the category of the next word form as well as constructing the category of the new sentence start. The rule also contains a rule package, that lists loom-rules that may be applied in the next step.

The three parts of the categorial operation are imported from a position p_x which is part of a syntax plan. p_x is a member of the so-called *position package* pp_y. A position package, which is represented in the category of the sentence start, lists all positions (for clauses and phrases) that may be imported by the next loom rule. Accordingly the first part of the categorial operation checks for the presence of this position package in the category of the sentence start. In the third part of the categorial operation, the position package pp_x of plan positions that may follow p_x is written to the new sentence start category.

Every position of a syntax plan has a type. The type corresponds to the loom-rule that imports this position during analysis. Table 1 shows which rules of the loom-grammar import categorial information from which plan positions.

	Loom rule ▶	**Open**	**Cont.**	**Close**	**Atomic**	**Delimit**	**Intermit**
Type of plan ▼	Type of position ▼						
Complex phrases	Opening	×					
	Continuing		×				
	Closing			×			
Atomic phrases					×		
Main clauses	all delimiters					×	
Nested clauses	initial delimiter						×
	other delimiters					×	

Table 1: Overview over the interaction of loom-rules and syntax plans: The crosses indicate which rules import their categorial information from which type of plan-position.

5.2 Algebraic definitions of syntax plans

This section gives an outline of the algebraic definition of syntax plans. Here only plans for main clauses and for complex phrases are given. The definitions of atomic plans, nested clause plans and phrase-nested clause plans work analogously.

5.2.1 Plans for main clauses

The set of main clause plans of an LLAG is defined as a quadruple $(P_M, CO_M, C_{IM}, C_{FM})$:

P_M: is a finite set of *main clause positions*, which are identified by a unique name in P_M.

CO_M: $P_M \mapsto (C \times C \mapsto C \cup \{\perp\})$ is a function that associates each main clause position p_x with a *categorial operation* co_x.[11]

$C_{IM} = \{(cat_{IM-1}), \dots\} \subseteq C$ is a finite set of *initial categories*. Together with the initial state of the loom, they form the initial states of the LLAG.

$C_{FM} = \{(cat_{FM-1}), \dots\} \subseteq C$ is a finite set of *final categories*. Together with the final state of the loom, they form the final states of the LLAG.

Together with the initial rule-package of the loom-grammar rp_0, each initial main clause category c_{IM} forms an initial state (c_{IM}, rp_0) of the LLAG. Analogously, each final category (pattern) c_{FM} forms a final state (c_{FM}, rp_1) of the LLAG where rp_1 is the final rule-package of the loom-grammar.

[11]Categorial operations in LLAG are functions exactly as in LAG (see (Hausser, 2001, Sec. 10.2.1)).

5.2.2 Plans for complex phrases

The set of complex phrase plans of an LLAG is defined as a quintuple $(P_{CP}, CO_{CP}, PI_{CP}, PM_{CP}, PF_{C}$

P_{CP}: is a finite set of ***complex phrase positions***, which are identified by a unique name. P_{CP} comprises the three disjunct subsets PI_{CP}, PM_{CP} und PF_{CP}:
$P_{CP} = PI_{CP} \cup PM_{CP} \cup PF_{CP}$.

CO_{CP}: $P_{CP} \mapsto (C \times C \mapsto C \cup \{\bot\})$ is a function that associates each complex phrase position p_y with a *categorial operation* co_y.

$PI_{CP} \subset P_{CP}$ is the set of initial complex phrase positions. These are imported by the rule `Open`[12].

$PM_{CP} \subset P_{CP}$ is the set of intermediate complex phrase positions. These are imported by the rule `Continue`[13].

$PF_{CP} \subset P_{CP}$ is the set of final complex phrase positions. These are imported by the rule `Close`[14].

The description of a field within a clause plan contains in its position package the names of those initial phrase positions, that can be opened in that particular field.

6 Results achieved with a German LLAG

The approach of LLAG has been applied to German syntax[15] in order to develop a system for the robust[16] analysis of free texts. The system for writing LLAGs was implemented with MALAGA[17].

The German syntax grammar uses the rule-based left-associative morphology DMM[18] (Lorenz, 1997), which produces feature-based morphosyntactic readings including heuristic weights. The DMM has a lexicon of about 50 000 base forms and reaches a coverage of 97.4% with an accuracy of 91.3% on the LIMAS-corpus[19] (Lorenz, 1999).

Table 2 gives an overview of coverage data derived by applying the German syntax to the LIMAS-corpus. The analysis rate indicates how many word forms of the input surface[20] could be analysed syntactically. A rate of 100% means all input word forms were analysed, while a rate of 90% means that every tenth input word could not be analysed and was punted.

A sentence of which at least 80% have been analysed is considered useful for further (e.g. semantic) analysis. Roughly 68% of the sentences analysed fall into this category. The average analysis produces 1.8 results per input sentence and builds an analysis tree with an average number of 154 states.

[12] and by the rule `NestedOpen`

[13] and by the rule `NestedPop&Continue`

[14] and by the rule `NestedPop&Close`

[15] An www-based demonstration of this grammar —as well as of further toy grammars for English, Korean and German— will be available by November 2001 (http://www.linguistik.uni-erlangen.de/ max/Tree/grammars.html).

[16] The actual implementation of the LLAG-system that was used also has mechanisms for the robust analysis. These include generation of hypotheses for unknown word forms and syntagms as well as mechanisms for weighting and pruning.

[17] see (Beutel, 1997)

[18] Also implemented in MALAGA.

[19] Corpus of the *Forschungsgruppe LIMAS* (Bonn, Regensburg), 1970/71, 500 pieces of texts of 2000 word forms each, available through the IDS in Mannheim.

[20] The term *surface* refers to the (unanalysed) graphematic representation of a word form or sentence.

[21] Ambiguities that cannot be resolved by syntactic means alone (e.g. many cases of PP-attachments) are left unresolved and do not contribute to the ambiguity rate.

Sentences completely analysed:	36.4%
Sentences analysed 90–99%:	12.9%
Sentences analysed 80–89%:	17.7%
Average analysis rate:	83.9%

Average rate of ambiguity[21]:	1.8
Average nr. of word forms/sentence	22.7
Average nr. of analysis states/sentence	154.4

Table 2: Coverage data for the current version of the German LLAG applied to the LIMAS-Corpus.

7 Conclusion

The approach of Loom-LAGs (LLAG) inherits all benefits of the algebraic properties of original LAG as LLAGs are just a specialised notational form of an LAG. The strength of LLAGs lies in using the loom-grammar common to all LLAGs to model syntactic principles common to natural languages. Thereby LLAG introduces the notion of explicit modelling of phrases and clauses as sequences —and thus in accordance with de Saussure's second law (de Saussure, 1913, p. 103).

The grammar developer can model different types of clauses and phrases separately as the loom-grammar provides the means neccessary to integrate these into one grammar during analysis. This makes LLAGs for syntax much easier to maintain and expand than their ordinary LAG counterparts, in which the descriptions of all syntactic structures lie intertwined in the whole grammar.

References

Berwick, Robert C. and Amy S. Weinberg. 1984. *The grammatical basis of linguistic performance: Language Use and Acquisition.* MIT-Press, Cambridge, Massachusetts.

Beutel, Björn, 1997. *MALAGA 4.3 - Online-Documentation.* Dept. of Computational Linguistics, University Erlangen-Nürnberg, Germany.

Chang, Suk-Jin, Key-Sun Choi, Jungha Hong, Junsik Hong, Jae Sung Lee, Juho Lee, Kiyong Lee, and Minhaeng Lee. 2000. Development of multilingual information retrieval and check system based on database semantics.

de Saussure, Ferdinand. 1913. *Cours de linguistique génerale,.*

Hausser, Roland. 1989. *Computation of Language.* Springer Verlag, Berlin.

Hausser, Roland. 1992. Complexity in left-associative grammar. *Theoretical Computer Science,* 106(2):283–308.

Hausser, Roland. 1999. *Foundations of Computational Linguistics.* Springer Verlag, Berlin.

Hausser, Roland. 2001. *Foundations of Computational Linguistics.* Springer Verlag, Berlin.

Hong, Jungha and Kiyong Lee. 1999. Processing korean relative adnominal clauses. In *Proceedings of the Eleventh Conference on Korean Characters and Korean Information Processing,* pages 265–271, Korea.

Lee, Kiyong. 1999. *Computational Morphology.* Korean University Press, Seoul.

Leidner, Jochen. 1998. *Linksassoziative morphologische Analyse des Englischen mit stochastischer Disambiguierung. (Master's Thesis).* Dept. of Computational Linguistics, University Erlangen-Nürnberg, Germany.

Lorenz, Oliver. 1997. *Automatische Wortformerkennung für das Deutsche im Rahmen von MALAGA (Master's thesis).* Dept. of Computational Linguistics, University Erlangen-Nürnberg, Germany.

Lorenz, Oliver. 1999. Deutsche Malaga-Morphologie (DMM). http://www.linguistik.uni-erlangen.de/~orlorenz/DMM/DMM.en.html.

Sampson, Geoffrey. 1999. Review of "Foundations of Computational Linguistics" (Hausser, 1999). http://www.linguistik.uni-erlangen.de/~rrh/Reviews_sampson.html.

Schulze, Markus. 1998. Morphologie, Syntax und Semantik im Rahmen der linksassoziativen Grammatik. In G. Heyer and C. Wolff, editors, *Linguistik und neue Medien*, Wiesbaden. Deutscher Universitäts-Verlag.