Proceedings 5. TaCoS, Saarbrücken (Germany), May 25–28, 1995; pp. 11–16
http://coli.uni-sb.de/~brawer/articles/cfg-german/

Treating German with a Provable Context-Free Grammar

Coping with Subcategorization, Unbounded Dependencies and Partially Free Word-Order

Sascha Brawer · University of the Saarland
FR 8.7, Computational Linguistics · 66041 Saarbrücken
e-Mail: brawer@coli.uni-sb.de

Abstract

Several applications of Natural Language Processing need short processing times, while retaining a certain degree of linguistic adequacy. This article describes some techniques employed in a text-to-speech synthesis system for German which allow for the parsing of sentences containing linguistic phenomena such as unbounded dependencies, subcategorization and free word order in the German Mittelfeld. The grammar is restricted to the unification of atomic terms only. It is shown that every grammar using such a formalism is equivalent to a context-free grammar.
Although the article may be of some theoretical interest — there is fairly widespread agreement in Computational Linguistics that complex linguistic phenomena cannot be handled with context-free grammars[1] —, some of the presented techniques could be useful for implementing fast language processing without too much loss of linguistic adequacy.
It has to be mentioned explicitly that the article does not state any linguistic theory, it simply describes a method for coping with certain phenomena in way which enables fast processing.

Introduction

For three months in 1993, I had the opportunity to work at the German language text-to-speech system SVOX. This system, developped by the speech processing group in the Electrical Engineering Department of the

[1] For an example, see [Haugeneder/Trost 1993].

Federal Institute of Technology at Zurich, Switzerland, is described in detail in [Russi 1990], [Traber 1993] and [Traber 1995]. A previous article, [Brawer 1994], describes the reasons leading to the choice of formalism; in addition, it presents the techniques shown below. This paper is principally an abbreviated version of the German original.

Context-Freeness of the Formalism

For the description of both syntactic and morphological knowledge, the SVOX system uses a grammar formalism which corresponds, apart from slight syntactic differences, to Prolog definite-clause grammars (DCGs).
It is indeed possible to represent non-context-free languages with definite-clause grammars. For example, [König/Seiffert 1989], p. 112[2] gives a DCG for the language $a^n b^n c^n$ which can be shown to be not context-free (see [Lewis/Papadimitriou 1981], p. 126).
Although the parser of SVOX is able to unify even complex terms (so having the same expressive power as the DCG formalism), a restriction to atomic values took place in the grammar development. The reason is increased efficiency: unification of complex terms is somewhat more complicated than atomic unification, which can basically be performed by a single comparision.
However, this restriction has rather severe theoretical consequences: It is a restriction to the class of context-free languages. The proof is sketched below.
Let A be the (finite) set of all discernable atoms contained in the grammar. When a variable is instantiated, its value becomes a member of A due to the restriction to atomic values. Therefore, we can replace any rule R containing a variable v by the new rules R [v/a] for every a A. This is repeated until none of the rules in the grammar contain any variables.
The result of such a replacement step is equivalent to the original rule

[2] This grammar "remembers" the number of ***a***'s, ***b***'s and ***c***'s as numbers and unifies them; this ensures that all items appear the same number of times. Embedded Prolog code increments the value of an attribute by one for each rule application. However, one could imagine a treatment which encodes a number as the length of a list, so avoiding explicit Prolog commands.

since it is not possible to determine by unification if there is a free variable $\boldsymbol{v}$ in a specific term of a rule, or if instead there are several rules, each one containing a possible instantiation for $\boldsymbol{v}$**.** Whatever $\boldsymbol{v}$ will be unified with, the result of the unification will be the same with or without this replacement. Since the original grammar is equivalent to itself and every step preverves equivalency, the new grammar is equivalent to the original one by induction.

After these replacement steps, we have a completely variable-free grammar; all terms are of the form $\boldsymbol{L}\,(\boldsymbol{a}_1, \ldots, \boldsymbol{a_n})$ with $\boldsymbol{a_i}$ $\quad \boldsymbol{A}$. Renaming to $\boldsymbol{L}\$\boldsymbol{a}_1\$\ldots\$\boldsymbol{a_n}$ (let \$ be a new symbol) leads to an equivalent context-free grammar.

Extensions to the DCG Formalism

Besides a slightly modified syntax, two extensions to the standard DCG formalism were developped by T. Russi and C. Traber (cf. [Traber 1995]). These extensions allow some of the mechanisms decribed below, but do not have any impact on the class of handled languages.

For a number of reasons, principally ambiguity discernation, some constructions should be considered as more probable than others; a very simple way to accomplish this is to assign a rule-specific penalty to each rule application. The penalties are summed over the whole derivation; the derivation with the lowest penalty is considered most probable and passed to subsequent components.

A second extension consists of rules marked to be “invisible”. Although they are processed in the normal way, their application does not appear in the structure passed to other modules.

Below, we will demonstrate some concrete applications of these extensions allowing a simple and fast treatment of subcategorization and free word order in the German Mittelfeld with a provable context-free grammar. Nevertheless, we would like to stress explicitly that the following are possible methods rather than adequate treatments of linguistic phenomena.

Pseudo Operators

The introduction of special operators or of functions, as in HPSG, would have made a change in the (Modula-II) source code of the parser necessary for every new function. This would not have been elegant, and further it would involve a mix of the levels of knowledge representation and processing.

By employing the described possibility to "hide" a rule application in the resulting trees, it is possible to introduce something we call ***pseudo*** operators. They allow the semantics of an operator to be stated in a declarative way without the need to extend the formalism. We would like to show the principle with the logical ***or.***

To determine intonation of the generated speech, it is important to know if a sentence is a question or a statement. If a constituent appears which through particular words or word order signals a question, the whole sentence should be treated as a question. To implement this, every constituent is given an attribute "interrogative" whose value is true if one of its daughters is a question. It is straightforward to implement this behaviour with an operator which calculates Boolean ***or:***[3]

```
or(f, f, f). penalty: 0 · invisible
or(t, _, t). penalty: 0 · invisible
or(_, t, t). penalty: 0 · invisible
```

An example for using this operator could be:

```
s(IsInterrg) -->
    np(NP_IsInterrg), vp(VP_IsInterrg),
    or(NP_IsInterrg, VP_IsInterrg, IsInterrg).
    penalty: 1 · visible
```

Such a declarative definition of functions is not new at all, but it is more common in the field of Logic Programming than in grammar development. In a "traditional" definite-clause grammar it would be necessary to integrate the "function call" directly into the rules by partial evaluation (in the example, this would lead to three rules for s). Alternativeley, the DCG mechanism of embedded Prolog code can be used — something not feasible in a framework separating knowledge representation and knowledge processing. One of the aims of this

[3] The SVOX grammars use a slightly different notation; I have aligned them to DCG for easier understanding.

separation is to allow a completely different “grammar processing engine”. For example, for efficieny reasons, a Modula-II program is used to parse in SVOX, disallowing completely any Prolog-specific constructs. Another interesting application of invisible rules is something called ***weak unification,*** a mechanism extensively used in the SVOX grammars.

Weak Unification

By means very similar to those mentioned above, it is possible to define an operator whose use is not penalized if its arguments match; otherwise, a penalty of e. g. 50 is applied.

```
WeakUnif50(A, A). penalty:  0 · invisible
WeakUnif50(_, _). penalty: 50 · invisible
```

Of course it is necessary to take certain precautions in the parser; otherwise, such a “mis-use of the formalism” would reduce efficiency.[4]

Treating Subcategorization and Free Word-Order

Let us now turn to the main subject of this article: how to treat certain linguistic phenomena using only the unification of atomic values. As has been shown above, this leads to a restriction to context-free languages. The task in hand, we would like to repeat that a linguistically adequate treatment of the phenomena was not the aim of the grammar. For example, the following mechanisms are linguistically rather unconventional, but they lead to usable results with a small amount of calculation.
A well-known problem in developping German grammars is the rather free word-order of complements and adjuncts in the so-called Mittel-

[4] SVOX employs chart parsing algorithms for both syntactical and morphological analysis (cf. [Traber 1995]). If the atoms in our example unify, both edges (with high and low penalty) will be inserted into the chart. Before inserting an empty inactive edge, the parser would have to check if a similar edge was not already present; if this was indeed the case, the penalty should only be minimized. However, this has not yet been implemented.

feld. In this part of German sentences, there seem to exist rather little restrictions on word order.[5]

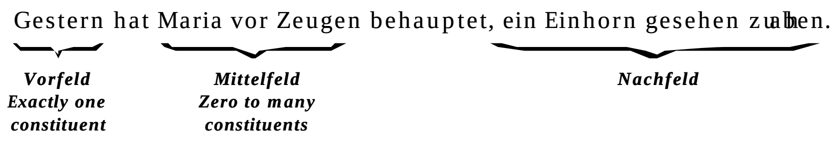
Gestern hat Maria vor Zeugen behauptet, ein Einhorn gesehen zu haben.
Vorfeld
Exactly one constituent
Mittelfeld
Zero to many constituents
Nachfeld

Fig. 1: Parts of a German declarative sentence

There now follows the description of a mechanism for treating subcategorization and free word-order in the Mittelfeld of German declarative sentences. The techniques have been modified (and complicated) slightly by including a treatment of the Nachfeld, but these modifications are not shown here. Further, the SVOX grammars currently do not cope with subcategorizing nouns (“die Tatsache, daß”), nor adjectives (“begierig, ihn zu sehen”). An implementation for the treatment of these phenomena would depend on methods similar to those shown below.

A formalism mixing rules for “hierarchical” dependencies ***(immediate dominance)*** and rules for “horizontal” word-order restrictions ***(linear precedence)*** would have some difficulties processing languages with (at least partial) free word-order. The basically context-free grammars described here belong to this class just as for example Lexical-Functional Grammar (LFG). In contrast, HPSG separates rules for ***immediate dominance*** and ***linear precedence*** explicitly (cf. [Pollard/Sag 1987], especially chapter 7). But there are several possibilities to treat the problem. For example, one could introduce special kinds of rules whose right-hand sides can be permuted freely; for LFG, a permutation operator has been proposed.

In our grammar, verbs can subcategorize among other things for subjects (***Ich*** gehe), accusative objects (***Mich*** friert. Ich nehme ***den Schal***), dative objects (Ich kaufte ***der Marktfrau*** einen Apfel ab), “zu” phrases

[5] In fact, there are at least some weak restrictions which cause certain readings to be preferred to others. This is a topic of current syntactic research, cf. e. g. [Pechmann et al. 1994]. For the purpose of text-to-speech synthesis, overgeneration of the grammar does not matter seriously. Thus, the Mittelfeld word-order can be assumed to be unrestricted with no great loss.

(Ich scheine ***immer alles zu vergessen***), infinitive constructions (Ich will ***einen Salat essen***) and "daß" phrases (ich verspreche, ***daß ich gehe***).

The lexicon entry of a verb holds the information as to which constituents it can subcategorize for. Because of free word-order in the Mittelfeld, it is common to use a set, or — if the formalism is not aware of sets — a list as a simulation of a set. However, we have stated that non-atomic feature values would be avoided for reasons of efficiency. Instead of using sets directly, we simulate them by assigning a Boolean feature for every potential element of the subcategorization set. Its value is ***true*** if the verb subcategorizes for the corresponding constituent. Using this mechanism, we can simulate (finite) sets: Each object can be a set member at most once, and in addition, the elements can be accessed directly without the need to traverse a list.

Optional complements can be represented by underspecification in a very efficient way: Instead of two entries for "geben" 'to give' — one with and one without dative object — the lexicon contains a single entry whose corresponding attribute value is not specified, respectively there is the Anonymous Variable which unifies both with ***t*** (complement allowed) and with ***f*** (complement not allowed).

If SVOX did not perform an elaborate morphological analysis, lexical entries would look like the following:

```
v(t,f,_,f,f,f,f,…) --> [geben].     »Ich gab (dir) alles«
v(t,f,f,f,f,t,f,…) --> [wollen].    »Ich will gehen«
v(t,f,f,f,f,f,t,…) --> [wollen].    »Ich will, daß Du gehst«
v(t,f,_,f,t,f,f,…) --> [scheinen].  »Anna scheint (mir)
                                    zu wachsen«
```

Fig. 2: Simplified lexical entries for selected words. The features stand for subject, accusative object, dative object, genitive object, "zu" phrase, infinitive construction and "daß" phrase

The word grammar ensures that infinitive verbs cannot subcategorize for subjects. Thus, it is not necessary to treat control constructions (raising and equi verbs) separately. Now, it is known which categories have to be present for a sentence to be complete; the next question is how a verb can bind these complements.

The main work of our sentence grammar is to "collect" the complements and adjuncts in the Mittelfeld and to set them in relation to Vorfeld and subcategorization frame in such a way that only sentences will be accepted which neither lack a complement nor do they have a superfluous one.

In accordance to the HPSG treatment of unbounded dependencies, our grammar uses ***two*** sets — respectively, as discussed above, simulations of sets — to solve this problem. Each Mittelfeld constituent bears the information as to which unbound complements it contains and which complements are needed to form a completed sentence. The former is implemented by a set which we call ***HAVE*** and which is comparable to ***TO-BIND*** in HPSG**,** the latter by a set called ***NEED.*** Different from the HPSG treatment of unbounded dependencies, our grammar works without traces.

The processing will be explained using the sentence "Diesen Ball wollte ich Maria gestern geben"; the resulting tree after parsing is given below.

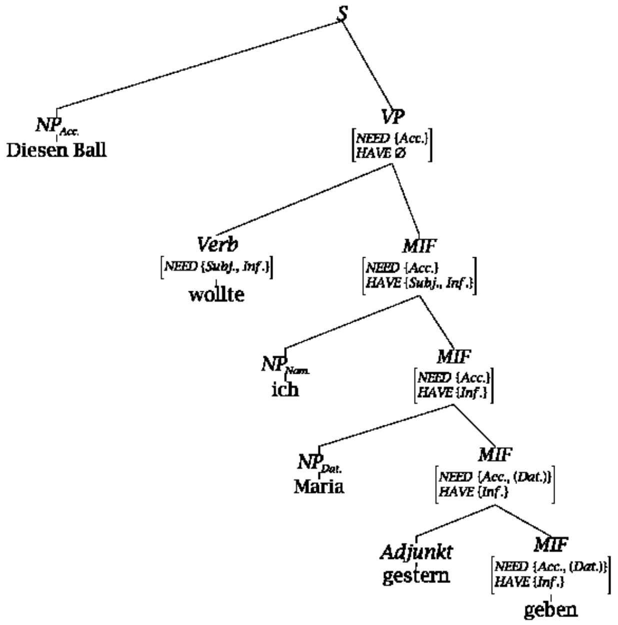

Fig. 3: The result of parsing the sentence "Diesen Ball wollte ich Maria gestern geben" ('I wanted to give this ball to Mary yesterday.'). ***MIF*** stands for Mittelfeld; see discussion below

The Mittelfeld is “constructed” in a recursive way: The verb ***geben*** subcategorizes for an accusative and optionally for a dative object; as mentioned above, the word grammar ensures that infinitives never subcategorize for a subject, thus allowing control constructions. So, the Mittelfeld which consists only of “geben” (at the bottom right of the tree) ***has*** an infinitive, viz. geben itself, and ***needs*** an accusative and optionally a dative.

An adjunct such as “gestern” (‘yesterday’) does not change the two “sets”; the attribute values are passed without changes.

The dative NP “Maria” combines with the Mittelfeld “gestern geben” to a larger Mittelfeld “Maria gestern geben” whose ***HAVE*** set still contains the infinitive and whose ***NEED*** indicates that an accusative object is missing.

The next constituent is the subject NP “ich” — but “Maria gestern geben” does not ***NEED*** a subject. Therefore, the subject is added to the ***HAVE*** set of yet unbound elements of the larger Mittelfeld.

In this recursive way, the Mittelfeld is constructed step by step. The needed grammar rules are explained at the example of the dative object; most other complements are treated similarly. Only subjects are somewhat more complicated because of their agreement in person and number with the finite verb.

– A dative object forms, together with a Mittelfeld not yet ***NEED***-ing a dative, a new Mittelfeld, now ***HAVE***-ing a “superfluous” (= unbound) dative. All other features are passed without change.

– A dative object forms, together with a Mittelfeld ***NEED***-ing a dative, a new Mittelfeld which does not ***NEED*** a dative and does not ***HAVE*** a dative to bind.

Because we simulate sets by attributes with Boolean values, it is possible to combine these two rules to a single one, using a negation operator: The new Mittelfeld has a dative in its ***HAVE*** set if and only if the old, smaller Mittelfeld did not ***NEED*** one. So the grammar contains one grammar rule for each of the possible complements.

We assume that no dative will be combined with a Mittelfeld containing an unbound, “superfluous” dative. Actually, this is not correct, but it has not led to wrong analyses yet. The reason is that it is rather seldom for two similar constituents to be bound non-locally.

When the Mittelfeld is constructed completely, it will be combined with the verb to the “classical” verb phrase. This is performed by a special binding operator:

verb	Mittelfeld		verb phrase	
NEED	NEED	HAVE	NEED	HAVE
t	f	t	f	f
t	f	f	t	f
f	t	f	t	f
f	f	t	f	t
f	f	f	f	f

Table: Semantics of the ***BIND*** operator. Read the first line as: If a verb needs an accusative and the Mittelfeld has one (then, of course, the Mittelfeld cannot need one), the resulting VP neither needs an accusative nor does it have one. The other lines are to be read in a similar way.

Some of the possible combinations are missing in the table; for example is it not possible for a Mittelfeld to both have and need an accusative — in this case, the dative would already have been bound (and removed from the sets) during Mittelfeld construction.
The application of ***BIND*** is repeated for each of the possible complements, thus calculating the sets of the VP.
As a last step, the verb phrase has to be connected with the Vorfeld. In our example, the sentence consists of an accusative object together with a VP needing an accusative. Other complements are treated in the same way. Additionally, in the Vorfeld can be adjuncts (“Gestern brannte es”) and expletives (“Es brennt ein Feuer im Garten”); in these cases, both ***NEED*** and ***HAVE*** must be empty.

Acknowledgments

I would like to express thanks to the group for speech processing of the Electrical Engineering Department of the Federal Institute of Technology, Zurich. Especially I would like to thank Christof Traber and Schamai Safra who held many discussions with me from which I greatly profited.

References

[Brawer 1994] Brawer, Sascha: Mechanismen einer kontextfreien Grammatik für das Deutsche. In: Sehn, Arthur/Autexier, Serge [ed.]: Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94. DFKI Document D-94-12. ISSN 0946-0098.

[Haugeneder/Trost 1993] Haugeneder, Hans/Trost, Harald: Beschreibungsformalismen für sprachliches Wissen. In: Görz, Günther [ed.]: Einführung in die künstliche Intelligenz. Bonn [etc.]: Addison Wesley, 1993. ISBN 3-89319-507-6. pp. 372 – 424.

[König/Seiffert 1989] König, Esther/Seiffert, Roland: Grundkurs Prolog für Linguisten. Tübingen: Francke, 1989. ISBN 3-7720-1749-5.

[Lewis/Papadimitriou 1981] Lewis, Harry R./Papadimitriou, Christos H.: Elements of the theory of computation. Englewood Cliffs (New Jersey): Prentice-Hall, 1981. ISBN 0-13-273426-5.

[Pechmann et al. 1994] Pechmann, Thomas/Uszkoreit, Hans/Engelkamp, Johannes/Zerbst, Dieter: Word Order in the German Middle Field. Linguistic Theory and Psycholinguistic Evidence. CLAUS [Computational Linguistics at the University of the Saarland] Report No. 43, August 1994. ISSN 0941-083X.

[Pereira/Warren 1980] Pereira, F. C. N./Warren, D. H. D.: Definite Clause Grammars for Language Analysis — A Survey of the Formalism and a Comparison with Augmented Transition Networks. Artificial Intelligence, Vol. 13, 1980. Seite 231 – 278.

[Pollard/Sag 1987] Pollard, Carl J./Sag, Ivan A.: Information-based syntax and semantics. Vol. 1: Fundamentals. CSLI Lecture Note No. 13. Stanford: CSLI, 1987. ISBN 0-93707-24-5 (paperback) and 0-937073-23-7 (cloth).

[Russi 1990] Russi, Thomas: A Framework for Syntactic and Morphological Analysis and its Application in a Text-to-Speech System. Diss. Federal Institute of Technology Zürich Nr. 9328, 1990.

[Traber 1993] Traber, Christof: Syntactic Processing and Prosody Control in the SVOX TTS System for German. In: Proc. Eurospeech '93, vol. 3, pp. 2099 – 2102, 1993.

[Traber 1995] Traber, Christof: SVOX: The Implementation of a Text-to-Speech System for German. Diss. ETH Zürich. Zürich: vdf Hochschulverlag, 1995. ISBN 3-7281-2239-4.