Natural Language Generation
Grammar Formalisms and their Processing

Dr. Günter Neumann
DFKI Saarbrücken,
neumann@dfki.de

August, 2002

Abstract This essay[1] provides an overview of the grammar formalisms currently in use and how they are processed in language production systems. The material in question includes constraint-based formalisms, systemic grammars and tree adjoining grammars. With the aid of selected studies we demonstrate how these formalisms are used in generation. In the final part, the problem of *grammar reversibility* is discussed, that is, the use of one and the same grammar for parsing and generation.

Introduction

One of the central aims of natural language generation is the development of computer systems which, on receipt of a given input (which corresponds to the system's communicative intentions), give a relevant and lucid reply in natural language. It has proven very helpful to model the linguistic production process as a complex series of decisions taking place on a number of levels. The necessary decisions can be roughly divided according to whether they are relevant to the determination or the realization of the content. When determining the content it is important to select and organize material which is relevant to the present conversational target (e.g. answering a question). Content realization focuses on transforming this information into a spoken or written utterance (i.e. the concrete answer). The main information sources needed for this consist of a lexicon and a grammar. Therefore the decision processes include selecting lexical elements and constructing the grammatical structure of the planned utterance. This latter point is the subject of the present study.

In order to be in a position to judge the various formalisms it is useful to take a look at how form and function of linguistic objects relate to each other. For one, utterances can be classified on grounds of their syntactic structure (for example, as a question or as an answer), regardless of the actual intention of the utterance. The task of answering such questions is allocated to the domain of pragmatics. Here linguistic elements are judged

according to their function in various communication situations. As an example, asking someone to do something can be formulated as a request or as an order, depending upon the situation, and on the receiving end it can be understood as a question or a demand. If language is to be understood thoroughly, these two aspects cannot, of course, be studied separately. In the context of generation, this means that the relationship between function and form also has to be formulated.

In natural language generation systems NLG a series of different grammar formalisms is used for the production of the syntactic structure. Up until the beginning of the eighties grammar formalisms for generation systems were mainly developed and implemented by researchers working in artificial intelligence. During that period the integration of the grammar into the complete system (and hence also the relationship between function and form) was stressed more than a comprehensive and linguistically sound description of the form of language.

As the importance of constraint-based grammar formalisms grew, the interest in generation in the field of computational linguistics increased considerably, and this led to a strong systemization of syntax oriented generation problems. In contrast to the more AI oriented approaches, which concentrate on questions specific to application and discourse, the main point of discussion in CL is the relationship between semantic representation and the possibilities of syntactic extension on the basis of linguistically founded methods. Pragmatic questions are, however, not currently being considered. Thus viewed, computer linguistic work in the field of generation is primarily concerned with the task of content realization.

Starting Point for Content Realization The input specification for the generation part responsible for content realization (in short: grammar generator) comprises semantic and pragmatic information. Ideally only a small amount of syntactic information should be specified (e.g. the period), for precisely this is meant to be constructed. Furthermore, a certain amount of flexibility should be allowed for concerning how exact the input has to be, since all semantic and pragmatic features do not always have to be available. The grammar generator should, in principle, be able to deliver adequate results in spite of lacking information.

Essentially, a grammar generator has to solve the two following partial tasks:

1. Selection of the lexical elements.

[1] This paper is a translation of *G. Neumann. Grammatikformalismen in der Generierung und ihre Verarbeitung*., which appeared in the German Journal on Artificial Intelligence "Künstliche Intelligenz", Special Issue on Natural Language Generation, W. Hoeppner (Ed.), Germany, 1993.

2. Construction of the syntactic structure of these elements.

Both tasks can be modeled as decision processes which underlie certain restrictions. An essential part of these restrictions is formulated in the lexicon and in the employed grammar. Below is a selection of the decisions that have to be made in order to solve the above mentioned partial tasks, although many of the decisions can only be adequately solved if the partial tasks interact.

- Selection of the content defining words (*house, building, skyscraper*)
- Selection of definite features (*the ball versus a ball*)
- Selection of pro-forms (*there, he, we*)
- Selection of connectors (*but, however, then*)
- Realization of negation (*I haven't got any money; I'm not going home*)
- Realization of modifiers (e.g. as an adjective in *the green apple*, or as a relative clause in *the apple which is green*)
- Realization of focus (e.g. via passivization or topicalization)
- Agreement of subject and verb (*Peter drives*; *the children walk*)
- Correct word order (e.g. not *apple the green*)
- Selection of morphological inflection (*des grünen Apfels (the green apple's)*)
- Elision of clauses (*Peter besitzt rote, Maria grüne Klötze (Peter owns red blocks, Mary green ones)*)
- Construction of infinitives (*I beg you to come*)

It is very important to consider how specific the input must be to be accepted by the grammar and the lexicon, and how to react in cases of insufficient information. In many systems it is assumed that the input is necessarily and sufficiently related to the decision factors, that is, it is presumed that all the information necessary for the best realization is present.

Some systems allow for the input to be under-specified in this respect. The missing information can either be determined by default values, or via interaction with other modules (e.g. dialogue components, text planners). A description of an interaction-based system can be found in [Hovy, 1987].

Below we will introduce some of the most well known formalisms: constraint-based formalisms (in particular FUG), systemic grammars and tree adjoining grammars. We will use examples from selected works to demonstrate how these are used in generation. Finally we will deal with the problem of reversibility.

Constraint-Based Grammars

Grammar oriented research is currently focused on constraint-based formalisms. The central idea of these formalisms is the description of linguistic categories via complex feature structures, and the realization of the information flow during linguistic processing via unification of such feature structures. Among the most well-known advocates of this class of formalism are Functional Unification Grammar [Kay, 1984], PATR II [Shieber *et al*., 1983], and Definite Clause Grammars [Pereira and Warren, 1980], as well as Lexical Functional Grammar [Bresnan, 1982], and Head-Driven Phrase Structure Grammar (HPSG. [Pollard and Sag, 1987]).

In the next section we will describe the formalism of Functional Unification Grammar, which is representative of the modeling and processing of linguistic knowledge in constraint-based formalisms. In the final section we will examine special processing aspects of some other formalisms.

Functional Unification Grammar

Functional Unification Grammar FUG [Kay, 1975; Kay, 1984] has been employed for the processing of grammatical structure in numerous very prominent generation systems (e.g. In [Appelt, 1985], [McKeown, 1985] and [McKeown *et al.,* 1990]).

Representation In FUG all linguistic entities (that is words, phrases and sentences) are represented by functional descriptions (FDs). An FD is a matrix consisting of attribute/value pairs, called features. The input (which is calculated by the part responsible for content determination) as well as the grammar and the lexicon are represented as FDs. The only operation allowed in FUG is the unification of FDs. Informally, the unification of two FDs leads to the construction of a new FD which contains the information on the input structures and is compatible with these.

Fig. 1 shows an FD for a (strongly simplified) example grammar of German in which sentences are defined as subject-verb-object or subject-verb sequences.

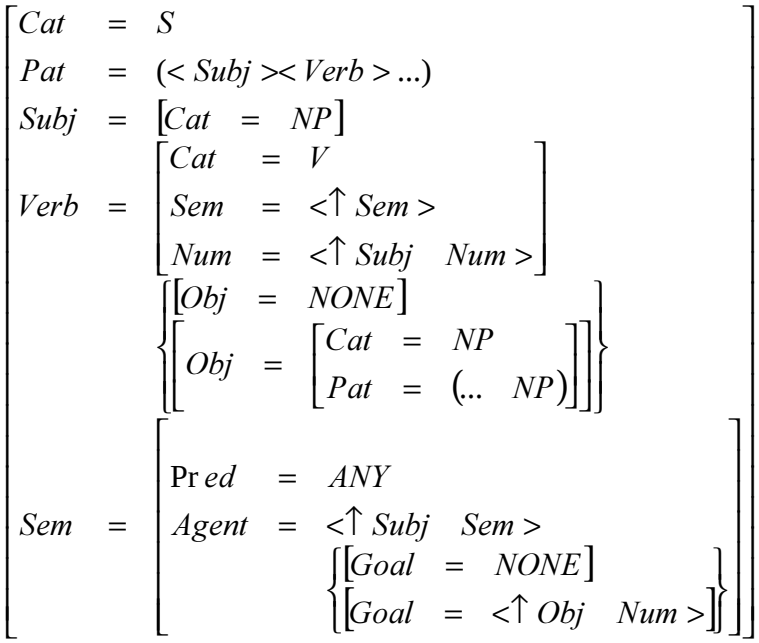

Fig. 1: A simple example grammar in FUG.

Features are noted in the form *a = v*; *a* marks an attribute and *v* its value. Attributes are any words without internal structure, whereas values can be described by various types. Essentially, nuclear and complex values are differentiated. In the above diagram **Cat** has nuclear values and the attributes **Subj** and **Pat** have complex values (in the case of Subj the value itself is an FD).

As the example demonstrates, an FD can also contain disjunctions of groups of features. The disjunction in the example (marked by braces) expresses that the object is optional and hence sentences can consist of two or three constituents. The *special* value **NONE** means that this alternative is only chosen when no object is specified in the input structure. The attribute **Sem** codes the semantics of the sentence. It is identical to that of the verb (expressed via the path ↑ *Sem* in the verb's FD). The value **ANY** has the status of a variable with the additional condition that it has to be substituted by a current value by the end of the unification. The value of the attribute **Num** describes a path. Paths are used to refer to the values of other attributes. This expresses that two attributes (which can pertain to various constituents) can have a common value or, in other words, that their values have to be identical.[2] Consequently, the example establishes that the number of the verb has to be identical with the number of the subject. In this way phenomena like, for example, congruence or projection (i.e. the relationship between a phrase and its head element; e.g.

[2] The symbol ↑ expresses that the root node of the path is situated in the next FD up. When two attributes share the same value it is also referred to as *structure sharing*.

the semantics of the sentence is identical to that of the verb) can be formulated very elegantly. In previous formalisms only procedural solutions existed for this (compare for example, [Meteer *et al.*, 1987]).

Information on the succession of the constituents is formulated by the attribute **Pat.** The value of this attribute is a list of paths to the immediate constituents of the FD. This list formulates position restrictions for the order of partial expressions of the constituents of the sentence. Formally, the list describes a regular expression about an alphabet consisting of the attributes of the constituents in which the feature **Pat** occurs. In the exemplary grammar the attribute **Pat** of the category *S* indicates that the surface string of a sentence begins with the subject followed by the verb and possibly further constituents. The possibility of options is expressed by ... In the exemplary grammar this position can be filled by the object, which means that the object has to stand at the end of a sentence. In order to identify the constituents of an FD exactly, FUG also makes use of the attribute **Cset**, the value of which is made up of a number of paths to the immediate constituents.[3]

Unification Unification is the only operation which is employed for processing; this is also true for sentence generation. In FUG sentences are produced by unifying the grammar with the input which, like the grammar, must also be formulated as an FD. The input for the unifacator is a semantic representation of what is meant to be expressed. The following diagram shows a possible (again highly simplified) input structure for the exemplary grammar described above.

The output is then a fully specified FD of the sentence "Peter liebt Maria" (Peter loves Mary), in which the linear sequence is described by the attribute **Pat** of the sentence constituent. If our exemplary grammar in Fig. 1 also included passive constructions, then it too could generate the sentence "Maria wird von Peter geliebt" (Mary is loved by Peter), if in Fig. 2, the attribute **voice** were occupied by the current value **passive**.

$$\begin{bmatrix} & & \begin{bmatrix} \mathrm{Pr}\,ed & = & [Sem & = & lieben] \\ Agent & = & [Sem & = & peter] \\ Goal & = & [Sem & = & maria] \end{bmatrix} \\ Sem & = & & & \\ Vocie & = & active & & \end{bmatrix}$$

Fig. 2: Input structure for the sentence "Peter liebt Maria":

[3] Hence FUG can be allocated to the ID/LP formalisms (compare [Gazdar *et al.*, 1985]) in which an explicit division of the constituent structure and linear sequence is expressed.

Unification can informally be interpreted as 'combination of information'. When two FDs are unified, the resulting FD should contain all the information of the input structures. There are two basic cases here, depending upon whether the value of an attribute is nuclear or complex. Nuclear values only unify when they are identical. The unification of complex values can be understood as recursive merging. The unification of values of the attribute **Pat** gives the average of the values. During unification checks are carried out to make sure that the structures to be unified are compatible. This is not the case, for example, when the values of an attribute which appears on the same level in both FDs do not unify (for example, when nuclear values are not equal or when the values have different types).[4] Essential aspects of the unification operation are (1) Order invariance i.e. that the order of the features is of no importance in the input structures (2) Commutivity (or bi-directionality) (3) Monotony and (4) Declaritivity - hence a grammar can be summed up as a mass of feature restrictions which can be added together to form an input, or checked, in which only which restrictions are pertinent is expressed, not the order of their processing.

FUG and Generation The advantage of FUG, particularly for generation, is related to the way that syntactic knowledge is coded. For the analysis of linguistic structures a hierarchical tree structure method has proven itself to be suitable. In contrast, FUG allows the formulation of a flatter structure of order restrictions. So a stronger stress of the functional roles of the constituents can be represented, which is of great importance for generation. In generation the key point of interest is the paradigmatic relationship between linguistic objects, that is, the choice between alternatives and their influence on the choice of the following elements. On the comprehension side, on the other hand, the syntagmatic relation, that is, the restrictions regarding the combination of objects, is the key point of interest.

Unification is frequently (at least in the way it is formulated in FUG for generation) realized as a non-deterministic top-down/depth-first control flow. However, this means that processing is far from efficient. To combat this problem, the formalism FUF [Elhadad, 1989], which is used in the generation system COMET [McKeown *et al*., 1990], made it possible to specify control information in the grammar. Thus, e.g. preferences can be specified for the selection of alternatives or whether the features of an FD should be processed via a depth or a width oriented strategy can be set.

Special Processing Aspects Strict top-down processing is not suitable for strongly lexicon oriented linguistic theories like, for example, HPSG [Pollard and Sag, 1987]. In such theories the majority of grammatical information is stored in the lexical entries. In contrast, the rules are described by schemata in a very abstract manner. For example, the rules

[4] A formal description of unification in constraint-based grammar formalisms can be found e.g. in [Shieber, 1986].

contain an abstraction from the verbs' special sub-categorization information, as the specific information is entered by the respective verbs. In principle, a rule schemata of the form *M* ➔ *HC** would be sufficient to produce verb-complement structures, for example (*M* is the mother node, *H* the head and *C** any, possibly empty set of complements. Let us also presume that this schema does not express anything about the correct word order). If we apply this schema to the verb *lieben* (to love) (which categorizes two nominal phrases), the following rule instance could be achieved: *S* ➔ *V NP NP*, in which *S* is interpreted as the projection of *V*.

A top-down driven process would therefore have to begin with an unspecified sub-categorization. [Shieber *et al.*, 1991] demonstrate that termination problems are caused by this and propose a bottom-up method as a solution. In this method the lexical entries corresponding to the (semantic) input are set first. Primarily lexical access is carried out for the semantic head (which is usually the predicate), so that their method can make restrictive use of the structural information of the input. Then the semantic arguments of this element are fixed via its grammatical structure and the whole process is continued recursively on the found arguments.

Systemic Grammars

The main point of focus in the development of systemic grammars [Halliday, 1985; Winograd, 1983] is the use of language in concrete communication systems, i.e. the question of the social function of language. The central idea in systemic grammars is the classification of linguistic utterances according to the function which they will have in conversational situations, i.e. if the utterance is to be realized as an instruction or a statement. A central target of systemic grammar scholars is to shape this kind of classification as delicately and specifically as possible. In the process, the complexity of the structure of natural languages is examined essentially according to the following three dimensions (see also [Winograd, 1983]): (a) classification (e.g. sentence types, word classes), (b) grouping (e.g. sequences of words as constituents), and (c) function (that is, the functional relationship between elements).

It is precisely the stress on paradigmatic and functional relationships between linguistic objects which makes systemic grammars particularly attractive to generation, as these describe which linguistic means can be allocated to which communicative situations. Not the form, but the function of linguistic objects, is in the foreground of the questions.

In a systemic grammar linguistic knowledge is represented by a network of related systems. On the lowest level the more specific decisions have to be made (e.g. choice of words) and

on the higher levels the less specific ones (e.g. sentence type). A single system is represented by a number of features and their possible values; conjunctive and disjunctive links are possible. Fig. 3 shows an excerpt from the system for English pronouns (conjunctive-linked decisions are represented by { and disjunctive ones by [).

For example, the pronoun *I* is selected on the basis of the information *Personal Singular First Subjective*, *she* by *Personal Singular Third Feminine Subjective* and *those* by *Demonstrative Far Plural*.

The generation of sentences in systemic grammars results in a series of functional decisions which are carried out parallel along the various functional areas. With its systemic grammar NIGEL, the system PENMAN is the most well known one to be built up on this theory [Matthieson, 1985]. Here a very extensive English grammar has been developed. In NIGEL a sentence is produced by a top/down driven traversal of the systemic networks. The values of the individual features are determined by the environment. If, for example, the value of the feature *Modus* is to be set (it is either imperative or indicative), then the semantic representation is asked whether or not it is an order. For this purpose, selection predicates are defined with each system; these carry out tests in the background program so as to be able to determine the current values of a feature. The choice of the follow-up systems depends upon the results of such questions and the current state of the system. This type of feature setting is called 'inquiry semantics' and allows the grammar to communicate with the background program's knowledge sources. In the system PENMAN world knowledge and semantic representation are represented in a KL-ONE-based formalism. Inquiry semantics serves as a mediator between grammar and knowledge representation. However, a disadvantage of this system is that communication is realized in a strongly procedural fashion and is hence dependant on a concrete application.

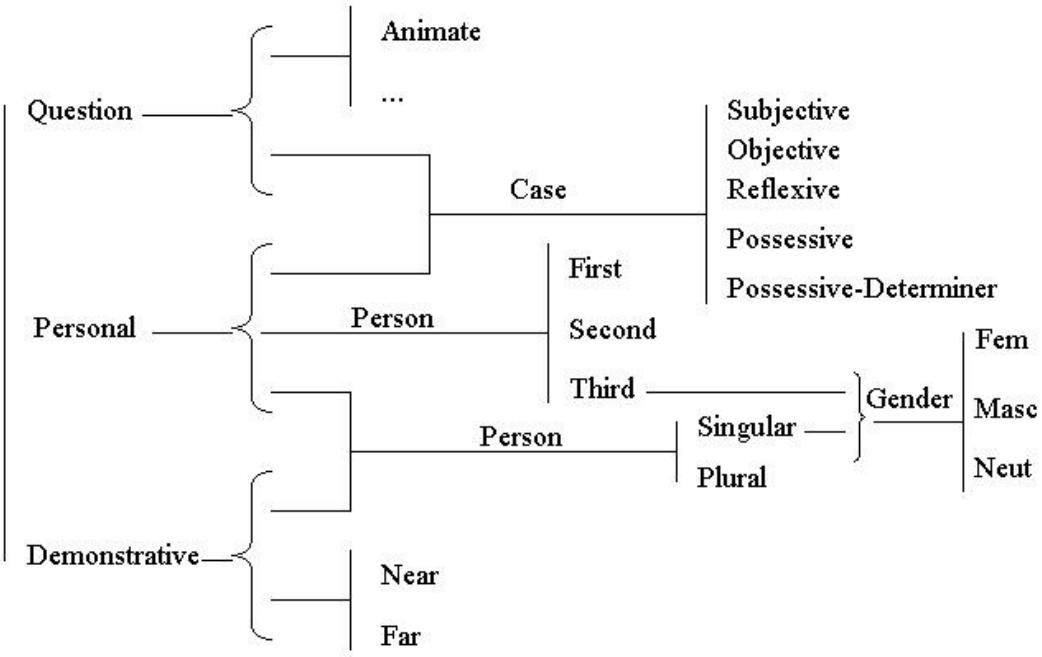

Fig. 3: The English pronoun system (according to [Winnograd 83]).

Tree Adjoining Grammars

Tree Adjoining Grammars (TAGs) were first introduced by [Joshi *et al.*, 1975] and have been examined intensively both formally and linguistically since then. A series of TAG variants was developed in which lexicalized TAGs [Abeille, 1988] and synchronized TAGs [Shieber and Schabes, 1990] are especially relevant to generation.

In the original definition a TAG consists of a finite number of elementary trees which are subdivided into a number of initial and auxiliary trees. The number of elementary trees constitutes the structural basis of a TAG. Initial trees represent minimal sentence structures. The root of an initial tree describes the sentence symbol S and the leaf nodes are described by terminal elements. All branch nodes describe non-terminal elements. The left-hand diagram in fig. 4 graphically represents the structure of initial trees. In the case of auxiliary trees, the root can describe any non-terminal symbol. With the exception of one element x, all leaf nodes describe terminal nodes. x describes a non-terminal element of the same category as the root (see the right hand diagram in fig. 4).

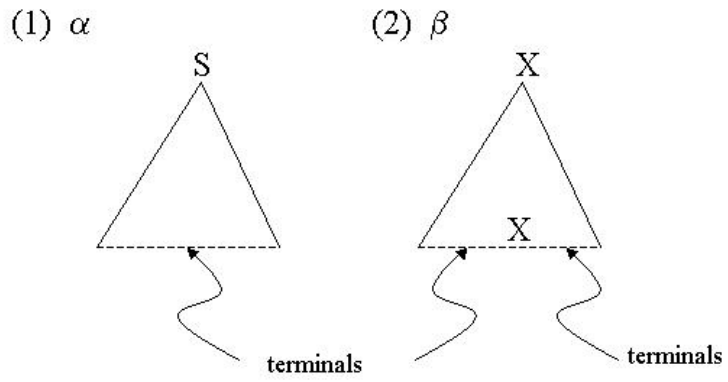

Fig. 4: The structure of initial (1) and auxiliary (2) trees (according to [Joshi 87]).

In contrast to context free grammars (CFG), where rules can be represented as trees with a depth of one, elementary trees represent a larger locality area than CFGs.

The central operation for constructing complex tree structures is adjunction. Via adjunction a new tree γ can be determined from an auxiliary tree β and tree α. Let α be a tree containing the node x and β an auxiliary tree the root of which is described by x. Fig. 5 is a graphic representation of how the new tree γ can be derived from α via adjunction of the auxiliary tree β and the node x (w_i and v_i being partial strings).

An advantage of adjunction is that it can be used to factorize the recursion of local dependencies (as is the case with CFGs). It could be shown that TAGS are more expressive than CFGs, as they can be used to describe languages which are not strongly context sensitive [Weir, 1988].

A disadvantage of the original formalism was the extremely redundant formulation of partial trees. For this reason [Abeille, 1988] introduced substitution as an additional operation allowing partial trees in elementary trees to be replaced by corresponding non-terminals. Hence non terminals marked by the substitution symbol $\downarrow$ are also accepted as leaf nodes.

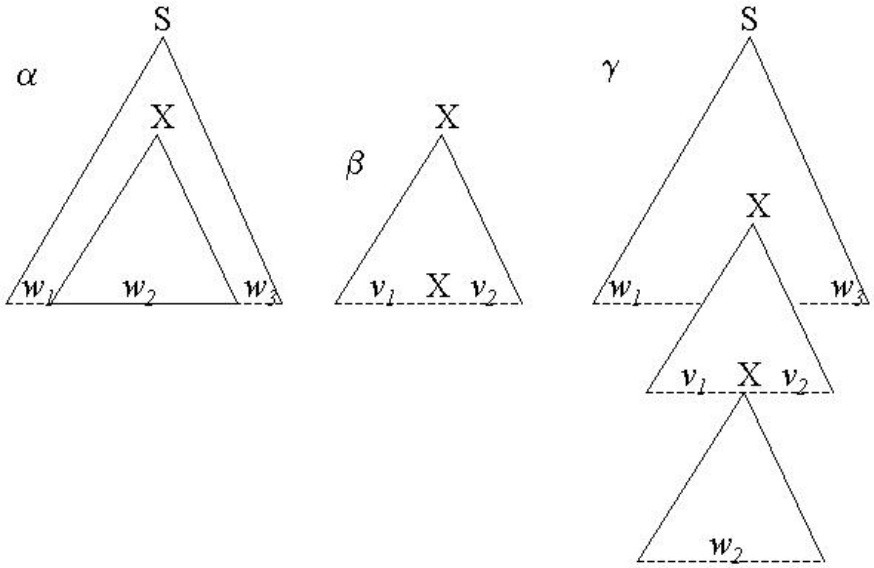

Fig. 5: Adjunction in TAG.

The only difference between the extracted partial trees and the initial trees is that their root node can describe any non-terminal. It could be shown that the addition of substitution does not influence the formal features of TAGs, but that it leads to a more elegant description of linguistic objects with less redundancies. Fig. 6 shows an example for a more compact initial tree and an extracted partial tree. Substitution is an obligatory operation, i.e. a derivative tree is only complete when all substitutions have been carried out.

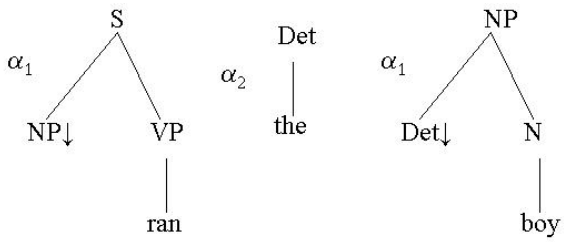

Fig 6: Initial trees in lexicalized TAGs.

Relevance of TAGs to Generation TAGs were primarily developed to formulate syntactic structures. They were already used in generation systems relatively early, namely in the system MUMBLE-86 [McDonald and Pustejovsky, 1985]. [Joshi, 1987] pays particular attention to the relevance of TAGs to generation. The combination of the following characteristics of TAGs is of significance for generation (1) larger locality area than in context free grammars, (2) the possibility to split dominance and linear sequences, (3) use of unification, (4) suitable for incremental processing.

We will now briefly introduce two more recent works that model some of these aspects for generation.

TAGs and Systemic Grammars The central idea of the approach described in [McCoy *et al.*, 1992] lies in the combination of systemic grammars and TAGs, in which the systemic grammar serves as a means to describe the functional relationship between linguistic objects and the TAG to describe the form of the objects. Additionally, the syntactic structures are represented in the TAG formalism. Elementary syntactic trees are classified in accordance with functional categories. The complete structure is described as a TAG network. A systemic grammar is employed to determine the number of functional features from an input structure, so that the relevant number of elementary trees can be determined with this information by traversing the TAG network. The trees which have been selected in this manner are then joined on grounds of their functional relationships. The above mentioned steps really are integrated in the process. The starting point for an integrated process is the head/modifier structure of the elementary trees. To represent these relationships a lexicalized TAG is used in which predicates are lexicalized and their allocated arguments represented by non-terminal elements (compare Fig. 6). In total, this results in a recursive generation process of the systemic grammar and the TAG network which is oriented on the head/modifier structure of the input.

Let, for example, the feature structure in Fig. 7 be the starting point for the generation process (following [McCoy *et al.*, 1992]). This information is the starting point for the traversal of the systemic grammar. The target is (a) to determine the head/modifier structure, (b) to extract the set of functional features which are needed for the traversal of the TAG network. The first step involves the calculation of the head element (*think* in our example) and under consideration of its structure the process is then continued recursively over the arguments (*you* and *hit*). For each tree determined in this way the functional information employed is recorded. The overall result is a structure of elementary trees oriented on the functional information; in the final step these are all joined together to form an overall structure in accordance with adjunction and substitution.

An essential advantage of the described approach is that the architecture allows functional aspects of generation to blend elegantly with syntactic ones. Both aspects are represented separately by specific formalisms, but their combination during the process is mutually restrictive.

$$\begin{bmatrix} s-act & wh-question & \\ wh-it & \mathbf{1} & \\ tense & past & \\ proc & think & \\ actor & you & \\ phen & \begin{bmatrix} proc & hit & \\ tense & past & \\ actee & john & \\ actor & 1\begin{bmatrix} type & person \\ id & quest \end{bmatrix} & \end{bmatrix} & \end{bmatrix}$$

Fig. 7: Input for *Who did you think hit John?*

Incremental Generation with TAGs In [Harbusch *et al.*, 1991] the TAG formalism is employed for incremental generation. In general, incremental generation is regarded as a prerequisite for the production of natural language. Based on numerous tests, incremental generation is psychologically motivated, and a psychologically motivated incremental generation system was first introduced by [DeSmedt and Kempen, 1987], although here only the content realization was modeled exactly. [Reithinger, 1991] describes an incremental system in which incremental processing on the level of content determination is also realized. Reithinger does not, however, claim that his model is in any way psychologically adequate.

In [Kempen, 1987] and [Neumann and Finkler, 1990] the following points are listed as requirements for incremental content realization: (1) use of lexicon centered grammars, (2) explicit separation of dominance and linear sequence, (3) the possibility of reformulation, (4) local decision criteria, and (5) operations for insertion as well as top-down and bottom-up expansion of syntactic structures.

In the incremental generation system introduced by [Harbush et al,. 1991], a lexicalized LD/LP-UTAG is used, in which LD/LP stands for 'Local Dominance/Linear Precedence', and U for unification. In this formalism elementary trees are interpreted as mobile ones, in which the number of possible permutations is restricted by linear precedence rules (e.g. the rule *det* < *adj* only allows pre-terminal sequences like *det adj n* or *n det adj*, but not a sequence

like *adj det n*). The unification in TAGs allows elementary trees to be annotated with feature restrictions. Thus, semantic restrictions as well as details about congruence can be associated with syntactically oriented trees.

The use of a lexicalized TAG is motivated by the fact that, at least in an incremental model, the syntactic processing was supposed to be best lexically driven [Levelt, 1989]. It is assumed that the content determining part sets a semantically oriented functor/argument structure of the utterance to be generated as the input structure for the realization part. Now appropriate lexical entries must be found for this structure by word selection. Each individual entry is then the starting point for a search for its 'own' allocated elementary trees. In the system by [Harbusch *et al.*, 1991], lemmata and functional relationships between them are expected as input for the realization using TAGs. Let us take the lexical trees from Fig. 6 as an example for lexicalized trees; then *Junge* (boy) and *lief* (ran) would be possible lemmata, and *agens(lief)* = *Junge* a functional relationship. The individual trees can be combined via the functional relationship, in which realization of further lemmata can be inserted via adjuncts. Structures which have been combined in this way are linearized and inflected in a further step.

What is exceptional about an incremental generation system is that partial structures are available for verbalization before the complete structure has been calculated. To allow spontaneous generation, an attempt is made to produce a locally complete syntactic structure for this partial information (or *segments*), so that this can be inflected in the following morphological component and uttered immediately. However, whether the structures can really be uttered immediately depends upon when the lemmata arrive. Let us presume that *Junge* arrives before *lief* in the above example. The problem with this sequence is that *Junge* cannot be uttered spontaneously as the case information is necessary for its inflexion, and that is determined by the verb. In principle, there are three possibilities to approach the problem: (a) the use of default values [DeSmedt and Kempen, 1987], (b) to demand explicit information [Finkler and Neumann, 1989], or (c) to wait until the verb is known [Harbusch *et al.*, 1991]. The advantage of (a) is that spontaneous generation is realized. A serious disadvantage is, however, that it is possible that a series of corrections will be necessary during the remaining generation process, since the defaults employed have to be revised by new information. In (b) this is avoided, but the disadvantage here is that only a delayed spontaneous utterance is possible. The advantage that (b) has over (c) is that in (b) there is no unnecessary waiting time as the missing information is demanded immediately.

Which of the described approaches is to be chosen depends upon the modeling of the segment sizes, so for example, if each individual adjective in a complex nominal phrase has to be uttered before the noun is known or if the entire nominal phrase has to be uttered.

Short Summary Before we introduce the aspect of reversibility in the next part, a short summary. Constraint-based formalisms (CG) and TAGs are partially suited to the representation of syntactic knowledge, while in CGs the relation between syntax and semantics is stressed additionally. In [McCoy *et al.*, 1992] an approach is introduced demonstrating how systemic grammars can be related to TAGs. An example of how systemic grammars can be combined with CGs can be found in [Bateman et al., 1992]. It is impossible to say which of the formalisms mentioned here is the most suited to generation. Maybe a combination of these approaches, as described above, comes the closest to the practical conditions. The aspect of reversibility may also provide further assessment criteria.

Reversibility

The idea of a reversible grammar, that is, the use of one and the same grammar for the analysis and production of sentences has frequently been outlined as a goal worth achieving (see [Neumann, 1994; Neumann 1998] for an overview). However, reversible grammars have only been properly investigated in the last few years, almost exclusively in the domain of Computational Linguistics. The starting point for the development of reversible grammars is constraint-based formalisms. As these require a declarative formulation of grammatical knowledge, they should, in principle, be able to be used bi-directionally. But it became apparent relatively quickly that grammars or formalisms which have been used exclusively in one direction up until now, cannot be employed in the other direction without problems. We will use the Lexical Functional Grammar (LFG, [Bresnan, 1982]) to demonstrate this. In LFG natural language sentences are described by a context free grammar (c-structure), the categories of which are annotated by a number of equations. These define a diagram on a functional level (f-structure) which explains the functor arguments. In the original derivation concept it was assumed that the c-structure of a sentence has to be set before the f-structure can be calculated by evaluating the functional equations. However, this process is not adequate for generation as the generation was meant to begin with an f-structure, but a direct relationship between the f-structure and the c-structure is not formulated. Therefore the c-structure would have to be constructed before its f-structure can be compared with the input. For these reasons LFG seemed to be better suited to parsing than to generation [Block, 1987]. But in [Wedekind, 1988; Wedekind, 1991], Wedekind showed that a re-wording of the derivation term is possible, so that parsing and generation can both be carried out effectively without influencing any essential elements of the theory. The central idea here is the simultaneous description of the c-structure and the f-structure. In this way,

whether the f-structure of the annotated equations cohere with the relevant description in the input can be checked in each step of the derivation. This procedure requires a top/down processing strategy.

Types of reversible systems Current work in the field of reversible grammars can be roughly divided into three types.

Type A Compilation of specific parsing and generation grammars from one grammar

Type B Use of one grammar, but different processes

Type C Use of one grammar and one uniform process

In systems of Type A, the linguistic knowledge for parsing and generation is formulated in one common grammar, which is compiled in a special parsing and generation grammar for the run time of the system. The advantage of this method is that the source grammar of each respective use can be tuned individually. A great disadvantage lies in the fact that during the runtime the grammatical knowledge is redundant in the complete system. Approaches which follow this method are, e.g. [Block, 1991] and [Dymetman *et al.*, 1990].

This disadvantage is avoided in systems of Type B, as parsing and generation operate on the same grammar. A further advantage is that the grammar can be tested and altered more easily. A disadvantage of this method is that the present processes are too inefficient.

Systems of Type C pursue the most radical approach to reversibility: not only is a common grammar employed, but also the same fundamental process. One of the first uniform architectures described in [Shieber, 1988] follows the paradigm 'Linguistic Processing as Deduction', Shieber uses a variant of the Early algorithm as the communal fundamental process. In [Emele an Zajac, 1990] a uniform architecture is introduced which is founded on the paradigm 'Linguistic Processing as Type Inference'. The only processing strategy it uses is complete type expansion via unification. The greatest disadvantage of both approaches is that they are seriously inefficient. Therefore they are not presently suitable for realistic use in NLG.

However, recently a series of suggestions has been made as to improving the efficiency of the two latter types of reversible systems, including efficient indexing techniques for the lexicon, or the use of delayed type expansion. [Uszkoreit, 1991] suggests the annotation of control information in the form of preferences and feature structures. This control information can be used to drive the sequence of conjuncts or disjuncts or to fade out

information. In principle, it would be possible to establish various preference systems for parsing and generation so as to be able to formulate specific control aspects.

Integration of Parsing and Generation In most reversible approaches grammatical processing is carried out independently of the discourse context of an utterance. In [Appelt, 1989] and [Neumann, 1991b] it is shown that a strict use of reversible grammars has a considerable influence on the design of an NLG, in particular the problem of chosing paraphrases is eradicated. In general it cannot be controlled if and to what extent the generated surface string is ambiguous. For example, the sentence 'Lösche den Ordner mit den Systemfiles' (delete the folder with the system files) is ambiguous, as it is unclear whether the folder is to be deleted with the help of the system files, or if the folder containing the system files is to be erased. Therefore NLG is faced with the risk of misunderstandings due to double meanings. In [Neumann and van Noord, 1992], a method for the solution of the problem is introduced which is based on a strict integration of parsing and generation with the use of a reversible grammar.
This tight mesh makes it possible to use the parser to support generation and vice versa. For example, parsing can be used during generation to pin-point relevant sources of ambiguity which can then be solved in one step. In Neumann (1994, 1998) we present a new model of natural language processing in which natural language parsing and generation are strongly interleaved tasks. Interleaving of parsing and generation is important if we assume that natural language understanding and production are not only performed in isolation but also work together to obtain subsentential interactions in text revision or dialog systems.
The core of the model is a new uniform agenda-driven chart algorithm, called UTA. Although uniformly defined, UTA is able to configure itself dynamically for either parsing or generation, because it is fully driven by the structure of the actual input - a string for parsing and a semantic expression for generation. Efficient interleaving of parsing and generation is obtained through *item sharing* between parsing and generation. This novel processing strategy facilitates the automatic exchange of items (i.e., partial results) computed in one direction to the other direction as well.

The advantage of UTA in combination with the item sharing method is that we are able to extend the use of memorization techniques to the case of an interleaved approach. In order to demonstrate UTA's utility for developing high-level performance methods, we present a new algorithm for *incremental self-monitoring* during natural language production.

Final Comments

In this essay an overview of the current grammar formalisms has been given. Selected examples were used to illustrate how systemic grammars (SGs), constraint-based formalisms (CGs) and tree adjunct grammars (TAGs) can be used in generation.

This essay can, of course, only supply an insight into the aspect of content realization, in particular such exciting themes as choice of words or the use of grammar formalisms in machine translation could not be touched upon due to space restrictions. Nonetheless, I hope to have demonstrated that 'Natural Language Generation Grammar Formalisms and their Processing' is still a hot theme, especially when the aspect of reversibility comes into the scene.

Literatur

[Abeille, 1988] A. Abeille. Parsing french with tree adjoining grammar: some linguistic accounts. In Proceedings of the 12th International Conference on Computational Linguistics (COLING), Budapest, 1988.

[Appelt, 1985] D. E. Appelt. Planning English Sentences. Cambridge University Press, Cambridge, 1985.

[Appelt, 1989] D. E. Appelt. Bidirectional grammars and the design of natural language generation systems. In Y. Wilks, editor, Theoretical Issues in Natural Language Processing 3, pages 206-212. Hillsdale, N.J.: Erlbaum, 1989.

[Bateman et al., 1992] J. A .Bateman, M. Emele, und S. Momma. The nondirectional representation of systemic functional grammar and semantics as typed feature structure. In Proceedings of the 14th International Conference on Computational Linguistics (COLING), Nantes, 1992.

[Block, 1987] R. Block. Can a `parsing grammar' be used for natural language generation? the negative example of LFG. In Proceedings of the first European Natural Language Generation Workshop, Abbey de Royaumont, 1987.

[Block, 1991] H. U. Block. Compiling trace & unification grammar for parsing and generation. In Proceedings of the ACL Workshop Reversible Grammars in Natural Language Processing, Berkeley, 1991.

[Bresnan, 1982] J. Bresnan, editor. The Mental Representation of Grammatical Relations. MIT Press, 1982.

[DeSmedt und Kempen, 1987] K. DeSmedt und G. Kempen. Incremental sentence production, self--correction and coordination. In G. Kempen, editor, Natural Language Generation, pages 365--376. Martinus Nijhoff, Dordrecht, 1987.

[Dymetman et al., 1990] M. Dymetman, P. Isabelle, und F. Perrault. A symmetrical approach to parsing and generation. In Proceedings of the 13th International Conference on Computational Linguistics (COLING), pages 90-96, Helsinki, 1990.

[Elhadad, 1989] M. Elhadad. Extended functional unification programmers. Technical Report Technical Report No. cuc-420-89, Department of Computer Science, Columbia University, 1989.
[Emele und Zajac, 1990] M. C. Emele und R. Zajac. Typed unification grammars. In Proceedings of the 13th International Conference on Computational Linguistics (COLING), pages 293--298, Helsinki, 1990.
[Finkler und Neumann, 1989] W. Finkler und G. Neumann. Popel-how: A distributed parallel model for incremental natural language production with feedback. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pages 1518-1523, Detroit, 1989.
[Gazdar et al., 1985] G. Gazdar, E. Klein, G. Pullum, und I. Sag. Generalized Phrase Structure Grammar. Blackwell, 1985.
[Grosz et al., 1986] B. Grosz, K. Sparck Jones, und B. L. Webber, editors. Readings in Natural Language Processing. Morgan Kaufmann, 1986.
[Halliday, 1985] M. A. K. Halliday. An Introduction to Functional Grammar. London: Edward Arnold, 1985.
[Harbusch et al., 1991] K. Harbusch, W. Finkler, und A. Schauder. Incremental syntax generation with tree adjoining grammars. Technical report, DFKI Saarbr¨ucken, 1991. RR-91-25.
[Hovy, 1987] E. H. Hovy. Generating Natural Language under Pragmatic Constraints. PhD thesis, Yale University, 1987.
[Joshi et al., 1975] A.K. Joshi, L.S. Levy, und M. Takahashi. Tree adjunct grammars. Journal Computer Systems Science, 10(1), 1975.
[Joshi, 1987] A. K. Joshi. The relevance of tree adjoining grammar to generation. In G. Kempen, editor, Natural Language Generation, pages 233--252. Martinus Nijhoff, Dordrecht, 1987.
[Kay, 1975] M. Kay. Syntactic processing and functional sentence perpective. In R. Schank und B.L. Nash-Webber, editors, Theoretical Issues in Natural Language Processing, 1975.
[Kay, 1984] M. Kay. Functional unification grammar: A formalism for machine translation. In Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics (COLING), pages 75--78, Standford, 1984.
[Kempen, 1987] G. Kempen. A framework for incremental syntactic tree formation. In Tenth IJCAI, pages 655--660, Mailand, 1987.
[Levelt, 1989] W. J. M. Levelt. Speaking: From Intention to Articulation. MIT Press, Cambridge, Massachusetts, 1989.
[Matthiesen, 1985] C. Matthiesen. The systemic framework in text generation: Nigel. In J. Benson und W. Greaves, editors, Systemic Perspectives on Discourse, volume 1. Ablex, Norwood NJ, 1985.

[McCoy et al., 1992] K. F. McCoy, K. Vijay-Shnaker, und G. Yang. A functional approach to generation with tag. In 30th Annual Meeting of the Association for Computational Linguistics, Newark, Delaware, 1992.

[McDonald und Pustejovsky, 1985] D. D. McDonald und J. D. Pustejovsky. Description-directed natural language generation. In Ninth IJCAI, pages 799-805, Los Angeles, 1985.

[McKeown et al., 1990] K. R. McKeown, M. Elhadad, Y. Fukomoto, J. Lim, C. Lombardi, J. Robin, und F. Smadja. Natural language generation in comet. In Robert Dale, Chris Mellish, und Michael Zock, editors, Current Research in Natural Language Generation, pages 103-139. Academic Press, London, 1990.

[McKeown, 1985] K. R. McKeown. Text Generation: Using Discourse Strategies und Focus Constraints to Generate Natural Language Text. Cambridge University Press, Cambridge, 1985.

[Meteer et al., 1987] M. W. Meteer, D. D. McDonald, S. D. Anderson, D. Forster, L. S. Gay, A. K. Huettner, und P. Silbun. Mumble-86: Design and implementation. Technical Report COINS Technical Report 87-87a, University of Massachusetts at Amherst, 1987.

[Neumann und Finkler, 1990] G. Neumann und W. Finkler. A head-driven approach to incremental and parallel generation of syntactic structures. In Proceedings of the 13th International Conference on Computational Linguistics (COLING), pages 288-293, Helsinki, 1990.

[Neumann und van Noord, 1992] Günter Neumann und Gertjan van Noord. Self-monitoring with reversible grammars. In Proceedings of the 14th International Conference on Computational Linguistics (COLING), Nantes, 1992.

[Neumann, 1991a] G. Neumann. A bidirectional model for natural language processing. In Fifth Conference of the European Chapter of the Association for Computational Linguistics, pages 245-250, Berlin, 1991.

[Neumann, 1991b] G. Neumann. Reversibility and modularity in natural language generation. In Proceedings of the ACL Workshop on Reversible Grammar in Natural Language Processing, pages 31-39, Berkeley, 1991.

[Neumann, 1994] G. Neumann: A Uniform Computational Model for Natural Language Parsing and Generation. PhD thesis, University of the Saarland, Saarbrücken, 1994.

[Neumann, 1998] G. Neumann: Interleaving Natural Language Parsing and Generation Through Uniform Processing. Artificial Intelligence 99, (1998) pp. 121-163.

[Pereira und Warren, 1980] F. C.N. Pereira und D. Warren. Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. Artificial Intelligence, 13, 1980. reprinted in [Grosz et al., 1986] .

[Pollard und Sag, 1987] C. Pollard und I. A. Sag. Information Based Syntax and Semantics, Volume 1. Center for the Study of Language and Information Stanford, 1987.

[Reithinger, 1991] N. Reithinger. Popel: A parallel and incremental natural language generation system. In C. L. Paris et al., editor, Natural Language Generation in Artificial Intelligence and Computational Linguistics, pages 179-199. Kluwer, 1991.

[Shieber und Schabes, 1990] S. M. Shieber und Y. Schabes. Synchronous tree-adjoining grammars. In Proceedings of the 13th International Conference on Computational Linguistics (COLING), Helsinki, 1990.

[Shieber et al., 1983] S. M. Shieber, H. Uszkoreit, F. C.N. Pereira, J. Robinson, und M. Tyson. The formalism and implementation of PATR-II. In B. J. Grosz und M. E. Stickel, editors, Research on Interactive Acquisition and Use of Knowledge. SRI report, 1983.

[Shieber et al., 1991] S. M. Shieber, F. C. N. Pereira, G. van Noord, und R. C. Moore. Semantic-head-driven generation. Computational Linguistics, 16:30-42, 1991.

[Shieber, 1986] S. M. Shieber. Introduction to Unification-Based Approaches to Grammar. Center for the Study of Language and Information Stanford, 1986.

[Shieber, 1988] S. M. Shieber. A uniform architecture for parsing and generation. In Proceedings of the 12th International Conference on Computational Linguistics (COLING), Budapest, 1988.

[Uszkoreit, 1991] H. Uszkoreit. Strategies for adding control information to declarative grammars. In 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, 1991.

[Wedekind, 1988] J. Wedekind. Generation as structure driven derivation. In Proceedings of the 12th International Conference on Computational Linguistics (COLING), Budapest, 1988.

[Wedekind, 1991] J. Wedekind. Unifikationsgrammatiken und ihre logik. Technical report, Arbeitspapiere des Sonderforschungsbereichs 340, 1991. Bericht Nr. 8.

[Weir, 1988] D. Weir. Characterizing mildly context-sensitive grammar formalisms. PhD thesis, University of Pennsylvania, 1988.

[Winograd, 1983] T. Winograd. Language as a Cognitive Process. Reading, Mass.: Addision-Wesley, 1983.