A Reusable Syntactic Generator for Dutch

Erwin Marsi

University of Nijmegen, Department of Language and Speech

Abstract

The syntactic generator is the component of a Natural Language Generation (NLG) system that takes care of the syntactic and morphological form of the generated natural language expressions. We argue that, once developed, a syntactic generator can be reused to build NLG systems for new application domains with less effort. We describe our work in progress on a reusable syntactic generator for Dutch called SEM2SYN. We discuss its input (semantic structures) and the tasks it is responsible for, i.e. inserting function words, inflecting content words and determining the word order. Next, we outline its implementation as a Functional Unification Grammar (FUG). We focus on the part of the grammar that determines word order at the sentence level. We conclude with a provisional evaluation of SEM2SYN.

1 Introduction

The goal of Natural Language Generation (NLG) is to build computer systems that map non-linguistic input to natural language output (Bateman and Hovy 1991, De Smedt, Horacek and Zock 1996, Reiter and Dale 1998). The input can range from a simple data structure to some sort of structured knowledge representation like a semantic network, possibly accompanied by a communicative goal. The output is a natural language expression, which may again range from a single phrase to a completely formatted (hypertext) document. Consider for example an NLG system that serves as the front-end to a train table information system. The input to the generator is a knowledge representation for a train journey, which contains information about the place and time of departure, the place and time of arrival, the places to change trains, train types, train numbers, platform numbers, etc. The overall communicative goal given to the NLG system is to inform the user about how to get from one place to another by train. The resulting output is a text that is intended to transfer to the user the information that is required to satisfy this communicative goal.

1.1 Architecture of NLG systems

In spite of the fact that NLG is a relatively new discipline and agreement about its fundamental goals, problems and methods is often lacking, there appears to be a consensus on the overall architecture of a NLG system (Reiter 1994, Reiter and Dale 1998). Most systems employ a *modular pipe-line architecture*, which consists of three modules that perform separate generation tasks and have one-way communication without feedback:

1. The **Text Planner** performs content determination (determining *what* to say) and discourse planning (expressing the content in a coherent way).

2. The **Sentence Planner** is concerned with aggregation (determining which pieces of information will be combined and expressed as phrases or clauses), lexicalization and referring expression generation (among others, generating a pronominal or definite expression where appropriate).

3. The **Linguistic Realizer** takes care of the syntactic, morphological and orthographic realization. Among its tasks are inserting function words, inflecting content words, determining word order and adding punctuation.

The sentence planner outputs a sequence of *sentence plans*. A sentence plan is an abstract representation for a sentence, also known as a *deep structure* or a *semantic structure*, which latter term we will use here. The task of the third module is to map a semantic structure to a string of words constituting a grammatical sentence. Again, there are various names in use for this module: *syntactic realization component*, *surface generator*, *tactical generator* or – the term we prefer – *syntactic generator*. For small application domains, this mapping can be done conveniently by using templates. For larger domains, however, where the range of possible expressions is much wider, the template approach is no longer feasible, and some form of grammar is required (see (Reiter 1995) for a comparison of the two approaches). The input to a grammar-based realizer has the form of a feature structure. It usually contains some sort of case frame structure, enriched with semantic features and the lemmas of content words. We will return to this matter in Section 2.

1.2 Reusability

One of the advantages of the modular architecture sketched above is that it allows for reuse of modules in new systems for other application domains. This is especially true for the syntactic generator, since it is relatively domain-independent, which is in turn due to the fact that morphology and syntax are to a large extent domain-independent aspects of a language. For English, there are a number of reusable syntactic generators available. Among the best known are: NIGEL (Mann and Matthiesen 1983) (the syntactic module of the Penman NLG system), which is continued in the multilingual KOMET system (Bateman 1994); SPOKESMAN (Meteer 1989); and SURGE (Elhadad and Robin 1996, Elhadad and Robin to appear) (see (Reiter 1994) for more systems). The concept of a reusable syntactic generator has proven to be quite successful. SURGE, for example, has been used in at least eight different NLG systems, for applications ranging from stock market reports to the technical documentation of telephone networks (Elhadad and Robin to appear, Appendix A).

As might be expected, the situation for Dutch is less fortunate. Although a number of NLG systems for Dutch have been developed (Claassen 1992, Appelo, Leermakers and Rous 1993, Marsi 1995, Teunissen 1997), none of these has given rise to a reusable syntactic generator comparable to those existing for English. Probably the best known system is IPG (Incremental Procedural Grammar) by Kempen & Hoenkamp (Kempen and Hoenkamp 1987), which formed the point of departure for IPF (Incremental Parallel Formulator) (De Smedt 1990). However, both systems appear to be primarily oriented towards implementing a psycholinguistic model of human grammatical encoding, and are less concerned with comprehensiveness. As far as we are aware, the only work that may come close to a reusable syntactic generator is the systemic grammar for Dutch that is under development as a part of the multilingual NLG system KOMET (Degand 1993). Nevertheless, the conclusion seems justified that there is room for reusable syntactic generators for Dutch. The availability of such generators would greatly facilitate the construction of NLG systems for Dutch.

The syntactic generator we have been developing for Dutch goes by the name SEM2SYN and is part of a concept-to-speech system called CONPAS (Concepts to Prosodically Adequate Speech). So far, CONPAS has been used for two application domains: (1) generating spoken train table information; (2) generating spoken descriptions of the botanical properties of plants. CONPAS is primarily meant as a research tool to test and extend linguistic theory about how prosody is related to syntax and semantics. In addition, the goal is to extend the idea of a reusable syntactic generator to include speech output as well. Our goal is a realization component that takes a semantic structure as input, that will subsequently take care of its syntactic, morphological and prosodic realization, and that will ultimately output prosodically adequate synthetic speech.

However, we will not elaborate on concept-to-speech or the CONPAS system here (but see (Marsi to appear)). Instead, we will focus on the present state of the syntactic generator SEM2SYN. First, we will describe the form of its semantic input on the basis of an example. Second, we will give an overview of the specific tasks that it carries out in order to arrive at a grammatical expression. We will centre on the question *what* is done during syntactic generation. The range of syntactic phenomena that is discussed not only illustrates why a syntactic generator is necessary in an NLG system for larger domains, but also that implementing a syntactic generator is far from trivial, since it must incorporate detailed knowledge of the syntactic structure of the language. Third, we will outline SEM2SYN's implementation as a modular Functional Unification Grammar. Here, we will focus on *how* syntactic generation is done. Since it is impossible to fully describe all implementation aspects here, we will concentrate on how word order at the sentence level is determined. We will finish with a provisional evaluation of SEM2SYN.

2 Input to the generator

The input to SEM2SYN is a semantic structure for a natural language expression (a sentence, a phrase, or just a single word). An example is given in Figure 1.[1] The input has the form of a *feature description* (FD), which consists of attributes and their values. The feature [CAT S] at the top level tells SEM2SYN that it has to generate a sentence. The semantic structure of this sentence is encoded under SYNSEM and has three principle components:

[1]The corresponding output is not particularly useful for any practical domain, but is only meant to illustrate several aspects of SEM2SYN in a single example.

$$
\begin{bmatrix}
\text{CAT} & S \\
\text{SYNSEM} & \begin{bmatrix}
\text{LEMMA} & \textit{betalen} \\
\text{TMS} & \begin{bmatrix}
\text{PARTIC} & \begin{bmatrix}
\text{AGENT} & \begin{bmatrix}
\text{CO1} \mid \text{SYNSEM} & \begin{bmatrix} \text{LEMMA} & \textit{Hein} \\ \text{PROPER} & \textit{true} \end{bmatrix} \\
\text{CO2} \mid \text{SYNSEM} & \begin{bmatrix} \text{LEMMA} & \textit{Niels} \\ \text{PROPER} & \textit{true} \end{bmatrix} \\
\text{CONJUNCTION} \mid \text{LEX} & \textit{of}
\end{bmatrix} \\
\text{THEME} & \begin{bmatrix}
\text{SYNSEM} \mid \text{LEMMA} & \textit{auto} \\
\text{DESCRIBERS} & \begin{bmatrix} \text{CO1} \mid \text{SYNSEM} \mid \text{LEMMA} & \textit{groot} \\ \text{CO2} \mid \text{SYNSEM} \mid \text{LEMMA} & \textit{duur} \end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
\text{CIRCUM} & \begin{bmatrix}
\text{TIME} \mid \text{SYNSEM} & \begin{bmatrix}
\text{TOP} & \textit{true} \\
\text{HRS} & \begin{bmatrix} \text{N1} \mid \text{VALUE} & \textit{1} \\ \text{N0} \mid \text{VALUE} & 2 \end{bmatrix} \\
\text{MIN} & \begin{bmatrix} \text{N1} \mid \text{VALUE} & \textit{1} \\ \text{N0} \mid \text{VALUE} & \textit{6} \end{bmatrix}
\end{bmatrix} \\
\text{LOCATION} \mid \text{SYNSEM} \mid \text{DISTANCE} & \textit{near} \\
\text{INSTRUMENT} & \begin{bmatrix}
\text{SYNSEM} & \begin{bmatrix} \text{LEMMA} & \textit{spaarcent} \\ \text{NUMBER} & \textit{plural} \end{bmatrix} \\
\text{DETERMINER} \mid \text{SYNSEM} & \begin{bmatrix} \text{LEMMA} & \textit{jan} \\ \text{PROPER} & \textit{true} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
\text{EXP} \mid \text{TENSE} \mid \text{PERFECT} & \textit{true}
\end{bmatrix}
\end{bmatrix}
$$

Om twaalf uur zestien heeft Hein of Niels hier een grote dure auto
At twelve hour sixteen has Hein or Niels here a big expensive car
betaald met Jans spaarcenten.
paid with Jan's savings.

Figure 1: Example of SEM2SYN's input and output. The feature description at the top is the semantic structure that serves as input. The Dutch sentence at the bottom is the string of words that is produced as output.

1. The *thematic structure* (TMS), which describes a situation (an event, state, belief, etc.) in terms of a predicate (LEMMA) and a number of thematic roles. Thematic roles are divided into obligatory *participants* (PARTIC) and optional *circumstances* (CIRCUM).[2]

[2]There are many proposals regarding the set of thematic roles that ought to be distinguished, which all employ slightly different labels and definitions (see (Winograd 1983, Section 6.5) for an overview). In fact, it is not even very clear what the criteria for deciding on thematic roles should be. Rather than

2. Other aspects of the meaning of a sentence that are systematically expressed by the grammar, which include the tense, mood and voice of a sentence. For lack of a better term, we will call this the *expressive structure* (EXP) of the sentence.

3. The *information structure*, which is a component of the semantic structure that is not localized in the value of a single attribute (as for TMS and EXP), but distributed throughout the input structure in the form of a Boolean feature FOCUS, which may occur on any constituent. The focus distribution affects the syntactic realization of participants and has a major impact on the word order. It is not explicitly given in Figure 1; we will discuss its contribution in Section 5.

The thematic roles will be realized as phrases or subordinated sentences. Normally, it is not necessary to specify their category, as this will be deduced by the generator. However, the input needs to be partly lexicalized; that is, it has to contain the lemmas of content words. For instance, it must contain the lemma for the main verb of the sentence (cf. *betalen* 'pay'). For thematic roles, providing a lemma is optional. If no lemma is given, SEM2SYN will assume that some form of referring expression must be generated.

The thematic roles may be internally structured. For instance, the THEME in Figure 1 has two DESCRIBERS, and the INSTRUMENT has a possessive determiner. In addition, they may carry a number of semantic features that affect their syntactic realization (e.g. PROPER and NUMBER).

As is clear from Figure 1, many features can be left out of the input, either because they receive an appropriate default value from SEM2SYN, or because their value can inferred from the other features. There are, for example, no values for MOOD and VOICE in Figure 1, so SEM2SYN will assign default values to them (*declarative* and *active* respectively).

3 Tasks of the generator

The tasks that SEM2SYN carries out during the generation of a syntactic structure can be divided into three main tasks: insertion of function words, inflection of content words, and determination of word order. As we will see in the next section however, these tasks are not necessarily carried out in this order.

3.1 Insertion of function words

An important distinction in syntactic generation is that between *function* (closed-class) words and *content* (open-class) words. Function words include auxiliary verbs, articles, pronouns and pronominal adverbs, complementizers, conjunctions and prepositions. Content words form the complement of the set of function words.

trying to evaluate the various proposals, or adding yet another one, we adopt the thematic roles from (Frawley 1992, Chapter 5), which is an attempt to compile a comprehensive set of roles out of several earlier proposals.

Content words result from a lexical choice process earlier on in the generation process, and their lemmas are provided in the semantic input to the syntactic generator (Stede 1995). In contrast, function words are in principle not provided in the input, because they are predictable from the semantic features in the input and the syntactic knowledge encapsulated in the syntactic generator.

Auxiliaries Auxiliaries are used, among other things, to express the voice and tense of a sentence. For instance, auxiliaries of tense are inserted according to the tense specification in the input, which allows the conventional eight tenses of Dutch to be described by means of three Boolean features: PAST, FUTURE and PERFECT (Haeseryn, Romijn, Geerts, de Rooij and van den Toorn 1997, Section 2.4.8). In Figure 1, the presence of the feature [PERFECT *true*] results in the insertion of the auxiliary *heeft* ('has').

Determiners The syntactic generator first has to decide whether a determiner is required at all. Proper nouns, by default, take no determiner (cf. *Hein* and *Niels*); neither do plural indefinite nouns. Moreover, a noun that is already specified by means of a possessor, a numeral or a quantifier normally does not take a determiner (cf. *Jans spaarcenten*). If it has been established that a determiner is obligatory, however, its lexicalization depends on both semantic and syntactic factors. Semantically, it depends on the definiteness and the number of the following noun. These are either provided in the input or given their default values (*indefinite* and *singular* respectively). Syntactically, it depends on whether the noun is neuter or non-neuter. This syntactic property is retrieved from a syntactic lexicon. We will return to the use of a lexicon in Section 4.3.

Pronouns and pronominal adverbs Pronouns handled by SEM2SYN include personal, possessive, demonstrative, relative, and wh-pronouns. Whenever the syntactic generator encounters a semantic constituent that can only be realized as a noun, but that lacks a lemma, it will try to insert an appropriate pronoun. For this, it uses the semantic features from the input (or defaults for them) as well as the syntactic context of the pronoun. For instance, the choice of a personal pronoun is based on the semantic features NUMBER, PERSON, GENDER, FORMAL and FOCUS, the purely syntactic feature CASE, and the Boolean feature REDUCED.[3] The case value of a personal pronoun is either *nominative* or *accusative*, depending on whether it functions as a subject or not. This is in fact one of the reasons why syntactic functions have to be assigned as part of the generation process.

Pronominal adverbs include those of the demonstrative type (e.g. *hier* 'here', *daarmee* 'with that') and those of the question type (*waar* 'where', *waarmee* 'with what'). Their realization results from their thematic role and their semantic features. For instance, the presence of the feature [DISTANCE *near*] at the thematic role of LOCATION in Figure 1 triggers its realization as *hier* 'here', instead of its default realization *daar* 'there'. If the thematic role had been that of INSTRUMENT,

[3]The feature REDUCED distinguishes a full pronoun like *jij* 'you' from a reduced one like *je*.

its realization would have been *hiermee* 'with this'. This illustrates one of the uses of thematic roles in the input.

Prepositions The fact that the distinction between function and content words is not always clear-cut comes to light especially in the case of prepositions. On the one hand, a preposition can be completely functional, i.e. with no meaning of its own, when it is subcategorized for by a verb (e.g. *wachten op* 'to wait for', *denken aan* 'to think about', etc.). If so, SEM2SYN retrieves the preposition from its lexicon. On the other hand, a preposition can carry a distinct semantic content, e.g. a particular spatial orientation as in *bovenop* 'on top of' and *onderaan* 'at the bottom of'. Prepositions of this type must be determined by the lexicalization process that precedes syntactic generation. In other words, they are treated just like content words, which must be provided in the input. In between these two extremes are cases where a default preposition is predictable from the thematic role. For instance, *om* 'at' to realize a Time and *met* 'with' to realize an Instrument (cf. Figure 1).

Conjunctions The default conjunction supplied by the syntactic generator is *en* 'and'. This choice can be overruled, however, by giving another lexical item in the input. In Figure 1, *of* 'or' is forced as the conjunction to use. This demonstrates that the input can (almost) always overrule the default lexical realization of a function word.

Complementizers Complementizers are inserted in subordinated sentences according to their tense and mood: *om* 'for' for non-finite sentences, and *dat* 'that' or *of* 'whether' for declarative and interrogative finite sentences respectively. Again, the input may overrule these defaults.

3.2 Inflection of content words

Content nouns Content nouns are inflected for number (cf. *spaarcenten* 'savings'), provided they are countable. In addition, they are inflected with the suffix *-s* when they are possessive (cf. *Jans* 'John's').

Content verbs Content verbs can be either finite or non-finite. Finite verbs are inflected for number (cf. *heeft* 'has'), and for person as well when combined with a personal pronoun. This requires the establishment of subject-verb agreement, which in turn presumes subjects are identified.

Non-finite verbs can take a special form according to the voice, tense or mood of the sentence. SEM2SYN infers the required form and retrieves the lexical form from its lexicon.

A special class is formed by compound verbs like *aankomen* 'arrive', because the verbal particle can be separated from the verbal root under certain conditions (e.g. *komt aan* litt. 'comes at' ('arrive') and *aan te komen* litt. 'at to come' ('arrive')). This means that SEM2SYN has to identify a verb as being a compound verb, and has to take care of the correct surface position of the verbal particle.

Inflection of adjectives Only attributively used adjectives can be inflected. The regular pattern is that the suffix *-e* is attached, unless the noun that they modify is definite, singular and neuter (cf. *een dure auto* 'an expensive car' versus *een duur schilderij* 'an expensive painting'). Evidently, a noun's grammatical gender is a purely syntactic property, and therefore should not be provided in the generator's semantic input. Instead, SEM2SYN retrieves this property of a noun from its syntactic lexicon.

Spell-out numeric expressions SEM2SYN is able to spell-out numeric expressions for time, date, dimensions, etc. (cf. 12:16 becomes *twaalf uur zestien*).

3.3 Determining word order

Word order at the phrase level is fairly rigid in Dutch. For instance, the order of the constituents in the NP *een dure nieuwe auto* (DET AP AP N) is almost completely fixed by the phrase structure rules of the grammar. Notice however, that the order of the adjectives is somehow determined by their semantic classification. Since this appears to be a purely semantic matter (Haeseryn et al. 1997, Section 14.5), the order among adjectives is assumed to be determined by the sentence planner. Multiple describers are given under CO1, CO2, CO3, ...in the input, and SEM2SYN outputs their respective realizations in this order.

In contrast, word order at the clause level is far less rigid. In fact, it is probably one of the most challenging aspects of syntactic generation for Dutch. We will explain how SEM2SYN handles word order at the sentence level in Section 5.

4 Implementation of the generator

4.1 SEM2SYN as a Functional Unification Grammar for Dutch

One of the syntactic generators for English mentioned earlier was the *Systemic Unification Realization Grammar of English* (SURGE) (Elhadad and Robin 1996, Elhadad, McKeown and Robin 1997, Elhadad and Robin to appear). SURGE is actually the name of the generation *grammar* that defines a mapping from semantic input to syntactic output. It is written in an extended version of a grammar *formalism* called *Functional Unification Grammar* (FUG), which is especially suited to NLG and MT (Kay 1984).[4] The extended version of FUG is interpreted by a Lisp generation program called the *Functional Unifier* (FUF) (Elhadad 1993). For SEM2SYN, we used the same formalism supported by the same generation program, but with a

[4]FUG is *not* a *linguistic theory* like e.g. GPSG, HPSG, LFG, etc. Instead, it is a formalism to express a linguistic theory in. In this sense, it is comparable to other theory-neutral formalisms like DCG and PATR.

Table 1: A comparison between SURGE and SEM2SYN with respect to target language, generation grammar, grammar formalism and generation program

Target language:	English	Dutch
Generation grammar:	SURGE	SEM2SYN
Grammar formalism:	FUG	FUG
Generation program:	FUF	FUF

different generation grammar, i.e. a grammar for Dutch. So SEM2SYN is actually a FUG for Dutch, and to some extent it can be considered as a first attempt to create a Dutch version of SURGE. This state of affairs is summarized in Table 1.

As the name implies, FUG primarily relies on the unification of *feature structures* (FD's), also known as *attribute-value matrices*. Both the input and the grammar are FD's. The generation process proceeds in two stages. First, a step-wise unification of input and grammar is performed, which enriches the input with syntactic information from the grammar. Second, the resulting FD is linearized to a string of words and punctuation marks. Unfortunately, we cannot go into the technical details of FUG here; but see (Shieber 1986, Elhadad 1993, Marsi to appear).

4.2 Structure of the grammar

SEM2SYN is not an implementation of a particular linguistic theory, since there appears to be no theory available that provides solutions for all the syntactic phenomena that a syntactic generator must deal with. Instead, the implementation incorporates approaches from Transformational Grammar (Model 1991), unification-based theories of grammar like HPSG (Pollard and Sag 1994), Systemic Functional Linguistics (Halliday 1994), and especially from structuralistic descriptions of Dutch (Haeseryn et al. 1997).

SEM2SYN has a modular structure. At the highest level, it consists of a grammar for generating phrases and sentences, and a lexicon for generating words. The distinction is a conceptual one, because technically, both grammar and lexicon are implemented as FUG's. The grammar contains separate modules for generating sentences, verb clusters, NP's, AP's, PP's, AdvP's, complex or coordinated structures, and some special categories like Numeral Phrases. Some of these modules consist of submodules themselves, which take care of particular syntactic aspects. For instance, the sentence module has submodules that handle sentence type, mood, subcategorization, linking thematic roles to syntactic functions, voice, realization of syntactic functions, and realization of circumstances. Modules can also have submodules for realizing particular subtypes of phrases. The module for NP's, for example, has submodules for handling pronouns, common nouns and proper nouns.

4.3 The use of a syntactic lexicon

One of the differences between SURGE and SEM2SYN is that the former uses the built-in morphological component of FUF. This morphological component takes care of the inflection of English content words during the linearization step. As it is not suitable for handling Dutch morphology, inflection in SEM2SYN is handled by means of a lexicon that lists the inflected forms of content words. On the one hand, this is a disadvantage, because it fails to capture a considerable amount of regularity and forces the user to extend the lexicon for each new word that needs to be generated. On the other hand, Dutch morphology contains many irregularities, which have to be stored in a lexicon anyway. Moreover, the survey of the tasks of a syntactic generator given in Section 3 revealed several cases in which the generator needs to know certain syntactic properties of content words:

- The choice of an article and the inflection of attributive adjectives depends on the grammatical gender of the noun.
- Some verbs take a prepositional object with an idiosyncratic preposition.
- Some verbs are compound verbs containing a stem and a separable particle.
- Some verbs take *hebben* 'have' as their perfect tense auxiliary, whereas others take *zijn* 'be'.

A syntactic lexicon therefore appears to be inevitable. The alternative, i.e. including this information in the input, would imply that more syntactic information is required in the input, which goes against the idea of modularization and the concentration of all syntactic knowledge in the syntactic generator.

5 Word order at the sentence level

Although strictly speaking processing order ought to be irrelevant in a unification-based grammar, it is convenient, both from a conceptual and a practical point of view, to look upon sentence generation as a process consisting of a number of steps. Starting with an almost completely underspecified syntactic structure, each step adds some constraints to the syntactic structure, ultimately resulting in a fully specified syntactic description. In order to demonstrate SEM2SYN's implementation as a FUG, we will focus on two of these steps here: sentence type and mood.

Sentence generation starts with the minimal assumption that every Sentence (S) must have a constituent with the function of *predicate* and the syntactic category of Verb Cluster (VC). The resulting pattern for an S is given in (1). The numbers are 'anchors' that will be used later on to determine the position of other constituents.

(1) $\prec$ ... 1 ... 2 ... 3 ... 4 ... PREDICATE ... $\succ$

5.1 Sentence type

Like most syntactic categories in SEM2SYN, S is specialized in terms of a type hierarchy. The direct subtypes of S are main-S and sub-S. In addition, an S can be finite or non-finite. The somewhat simplified FD in Figure 2 shows how S-type and finiteness affect the presence and position of a finite verb or a complementizer. The two top-level alternatives, surrounded by the outer braces, are for finite and non-finite sentences respectively (cf. the value of the feature FINITE). The first alternative contains two alternatives itself: one for a main-S (cf. #1a) and one for a sub-S (cf. #1b). If the sentence is a finite main-S, a constituent with the name FINITE is added to the sentence's pattern. In FUG, the special feature PATTERN is used to derive the internal word order of a constituent after the input FD and grammar FD have been unified. The realization of FINITE is shared with that of a constituent with the same name in PREDICATE, i.e. in the verb cluster.[5] This can be looked upon as SEM2SYN's equivalent of *verb second* in Transformational Grammar. If the sentence is a finite sub-S, the finite verb is added to the predicate's pattern; in other words, the finite verb will be realized in the verb cluster. If, however, the sentence is non-finite (cf. #2), it must be a sub-S, without a subject and without a finite verb. Optionally (as indicated by the round brackets), it starts with the complementizer *om* 'for'.

This grammar fragment also demonstrates how default values are implemented. During the unification of input and grammar, the alternatives are tried in the order they occur in the grammar. This means that if the input for an S does not specify for a particular S-type, nor for finiteness, it will be unified with the first alternative available (cf. #1a), and will therefore become a finite main-S. Notice that changing the order of the alternatives would give rise to different default values.

5.2 Mood

SEM2SYN distinguishes the common types of mood: declarative, imperative, polar-question (or yes-no-question), and wh-question. The subgrammar that handles most of the mood is shown in Figure 3. The first two alternatives at the top-level differentiate between main and subordinated sentences. For a main-S, a *declarative* or *wh-question* mood has no direct consequences – these will be taken care of elsewhere – whereas a *polar-question* or *imperative* mood forces a sentence to start with a finite verb. In addition, an *imperative* mood allows no subject. For finite subordinated sentences, all moods except *imperative* are allowed. A *declarative* mood forces the sentence to start with the complementizer *dat* 'that' (unless another lexical item is given in the input); a *polar-question* mood requires the complementizer *of* 'whether' (again, unless another one is given). A *wh-question* mood is allowed as well, although its realization will be taken up elsewhere. All remaining sentences, i.e. non-finite sentences, allow no value for mood.

[5]Feature sharing in FUG is expressed by means of a *path* that points to another location in the FD, analogous to the notion of a directory path as used in operating systems. The relative path $<\uparrow$ PREDICATE FINITE$>$ means: go up one level and from there follow the features PREDICATE and FINITE to reach the intended value.

$$\left\{\begin{array}{l}
\left[\begin{array}{l}
\text{SYNSEM} \mid \text{EXP} \mid \text{FINITE} \quad \textit{true} \\
\left\{\begin{array}{l}
\text{\#1a: Finite main sentence} \\
\left[\begin{array}{ll} \text{CAT} & \textbf{main-S} \\ \text{FINITE} & (\uparrow \text{PREDICATE FINITE}) \\ \text{PATTERN} & \prec \ldots \text{FINITE} \ldots 1 \ldots \succ \end{array}\right] \\
\\
\text{\#1b: Finite subordinated sentence} \\
\left[\begin{array}{ll} \text{CAT} & \textbf{sub-S} \\ \text{PREDICATE} \mid \text{PATTERN} & \prec \ldots \text{FINITE} \ldots \succ \end{array}\right]
\end{array}\right\}
\end{array}\right] \\
\\
\text{\#2: Non-finite sentence} \\
\left[\begin{array}{l}
\text{SYNSEM} \mid \text{EXP} \mid \text{FINITE} \quad \textit{false} \\
\text{CAT} \quad \textbf{sub-S} \\
\text{SUBJECT} \quad \text{NONE} \\
\left(\left[\begin{array}{ll} \text{COMP} & \left[\begin{array}{ll} \text{CAT} & \textbf{comp} \\ \text{LEX} & \textit{om} \end{array}\right] \\ \text{PATTERN} & \prec \text{COMP} \ldots \succ \end{array}\right]\right)
\end{array}\right]
\end{array}\right\}$$

Figure 2: FUG fragment for sentence type

5.3 Remaining steps in sentence generation

Subcategorization Next, the subcategorization frame of a verb is retrieved from the lexicon and unified with the input. Subcategorization requirements may enforce, among other things, the realization of a thematic role as a particular syntactic category. For instance, the verb *wachten (op)* 'wait (for)' forces its Theme to be realized as a PP. This in turn affects the word order, since PP's can be moved around more easily than NP's.

Linking SEM2SYN implements a version of *Linking Theory* (Jackendoff 1990), which links the participant thematic roles to core syntactic functions. There are maximally three of these syntactic functions: subject, first object and second object.[6] The assignment of grammatical functions facilitates the determination of word order. The position of a subject, for instance, is relatively fixed. Likewise, prepositional objects need to be closer to the predicate than adjunct PP's (unless they are fronted).

Passive voice may also be considered a source of word order variation, especially when it is analysed as movement of syntactic constituents. However, SEM2SYN accounts for passive voice in terms of a different linking of thematic roles to syntactic functions: the most prominent thematic role is demoted and

[6]We ignore relational verbs (e.g. 'be'), which are linked differently.

$$\left\{\begin{array}{l}
\left[\begin{array}{ll}
\text{CAT} & \textbf{main-S}\\
 & \left\{\begin{array}{l}
\left[\text{SYNSEM}\mid\text{EXP}\mid\text{MOOD}\quad (\textit{declarative wh-question})\right]\\
\left[\begin{array}{ll}\text{SYNSEM}\mid\text{EXP}\mid\text{MOOD} & \textit{polar-question}\\ \text{PATTERN} & \prec \text{FINITE}\ldots\succ\end{array}\right]\\
\left[\begin{array}{ll}\text{SYNSEM}\mid\text{EXP}\mid\text{MOOD} & \textit{imperative}\\ \text{PATTERN} & \prec \text{FINITE}\ldots\succ\\ \text{SUBJECT} & \text{NONE}\end{array}\right]
\end{array}\right\}
\end{array}\right]\\
\left[\begin{array}{ll}
\text{CAT} & \textbf{sub-S}\\
\multicolumn{2}{l}{\text{SYNSEM}\mid\text{EXPRESS}\mid\text{FINITE}\quad \textit{true}}\\
 & \left\{\begin{array}{l}
\left[\begin{array}{ll}\text{SYNSEM}\mid\text{EXP}\mid\text{MOOD} & \textit{declarative}\\ \text{COMP} & \left[\begin{array}{ll}\text{CAT} & \textbf{comp}\\ \text{LEX} & (\textit{dat}\ \text{GIVEN})\end{array}\right]\\ \text{PATTERN} & \prec \text{COMP}\ldots\succ\end{array}\right]\\
\left[\begin{array}{ll}\text{SYNSEM}\mid\text{EXP}\mid\text{MOOD} & \textit{polar-question}\\ \text{COMP} & \left[\begin{array}{ll}\text{CAT} & \textbf{comp}\\ \text{LEX} & (\textit{of}\ \text{GIVEN})\end{array}\right]\\ \text{PATTERN} & \prec \text{COMP}\ldots\succ\end{array}\right]\\
\left[\text{SYNSEM}\mid\text{EXP}\mid\text{MOOD}\quad \textit{wh-question}\right]
\end{array}\right\}
\end{array}\right]\\
\left[\text{SYNSEM}\mid\text{EXP}\mid\text{MOOD}\quad \text{NONE}\right]
\end{array}\right\}$$

Figure 3: FUG fragment for mood

linked to an adjunct PP, while the second most prominent thematic role is linked to the subject function.

Participant realization After being linked to one of the core syntactic functions, a participant needs to be syntactically realized. If it is in the scope of a relative clause or a wh-question, it will be realized as a relative pronoun or a (phrase containing a) wh-element, and it will occupy the first sentence position. Otherwise, its position is affected by a combination of factors:

- *Syntactic function.* As already mentioned above, a subject takes a different position than objects.
- *Syntactic category.* Normally, extraposition of NP objects is impossible. In contrast, PP objects may be extraposed, whereas S objects are by default extraposed.
- *Focus.* The focus distribution, encoded by means of a Boolean feature FOCUS, affects word order, because the canonical word order requires unfocused material to precede focused material. Therefore, unfocused NP ob-

jects are put before anchor 1 (cf. (1)), while focused NP objects are put between anchors 3 and 4. Furthermore, an unfocused PP object cannot be extraposed. Since SEM2SYN by default focuses content words and defocuses function words, an object realized as a referring expression will normally precede fully lexicalized constituents. In other words, the distinction between content and function words influences the word order as well, albeit in an indirect way.

- *Complexity.* The Boolean feature COMPLEX marks a constituent as exceptionally complex (heavy). It can be used to force extraposition of an NP or PP, or to suppress extraposition of an S.

Circumstance realization Circumstantial thematic roles are not linked to syntactic functions, but realized directly as syntactic constituents, usually PP's or AdvP's. Like participants, their position is affected by their syntactic category, their focus value and their complexity. Unless topicalized or extraposed, unfocused circumstances are located between anchor 1 and 2, whereas focused circumstances are located between anchor 2 and 3.

6 Evaluation of the generator

Since SEM2SYN is really work in progress, an evaluation is rather preliminary. In due time, a detailed comparison with other generators is desirable, especially with other generators for Dutch, but right now we are not in a position to perform an extensive evaluation. Instead, we will discuss SEM2SYN with respect to a number of criteria which in our opinion are essential to a reusable syntactic generator.[7]

1. It should be able to generate *all* syntactically well-formed expressions, given that their corresponding semantic structures are provided. In other words, the generator should cover all expressions of the target language.

   SEM2SYN can handle the standard moods, voice alternations (active, passive, pseudo-passive), the conventional tenses, 'movements' like topicalization and extraposition, coordinated constituents, modifiers at the sentence level (circumstances) and at the phrasal level (including relative clauses), compound verbs, a range of pronouns and pronominal adverbs, numbers and times. Among the phenomena not covered yet are subject and object control, modal verbs, reflexive and reciprocal pronouns, cleft and pseudo-cleft constructions, and ellipses. Also, the lexicon is still relatively small. We can therefore conclude that although a considerable part remains to be covered, SEM2SYN already accounts for a major part of the 'core grammar' of Dutch. Moreover, we see no principle obstacles that would prevent extension of SEM2SYN.

2. It should generate *only* syntactically well-formed expressions.

[7]Most of these criteria were taken from (Elhadad and Robin 1996, Elhadad and Robin to appear).

Since SEM2SYN is intended as a syntactic generator and not as a recognizer of well-formed semantic structures, it is not possible to reject all ill-formed input. Therefore, generation of ungrammatical expressions cannot be completely excluded. However, in practice over-generation does not occur.

3. For every input, it must terminate (within a reasonable period of time).

   On a moderate PC (Pentium 133Mhz), short sentences (less than 10 words) are usually generated within a second, whereas longer sentences (20 words or more) may take a few seconds. Since FUF allows one to put an upper limit on the number of backtracking points, the system always terminates.

4. The form and content of its semantic input have to be well-defined and documented in order to make a syntactic generator reusable.

   The form and content of the semantic input is not completely stable yet. Apart from the comments in the grammar code, there is no documentation yet.

5. It should minimize the amount of syntactic information that is required in the input. In other words, all syntactic knowledge should be concentrated in the syntactic generator.This requirement follows from the goal of modularity of an NLG system.

   SEM2SYN appears to be fairly successful at this point. In this respect, it is better than SURGE, due to the fact that it uses a syntactic lexicon, which contains syntactic information that would otherwise have to be supplied in the input.

6. It has to provide sensible defaults whenever a semantic structure is underspecified. This relieves a user or client application from the obligation to know all the details of the semantic structure. Instead, they can safely ignore irrelevant details, because the generator will choose sensible default values for them. In addition, it allows for a compact specification of the input.

   SEM2SYN's default values are adequate in the sense that almost every feature can be omitted from the input.

7. Its implementation should be grammar-based, so that there is a clear division between the declarative and procedural parts of the grammar.This significantly facilitates comprehension, maintenance and extension of the grammar.

   As a FUG implemented in FUF, SEM2SYN meets this requirement.

8. For the same reasons, the grammar has to be modular.

   SEM2SYN is a modular grammar.

9. It has to be portable across platforms, as this increases its reusability.

   As a program written in FUF, which is in turn coded in Lisp, SEM2SYN is portable across almost all platforms.

10. It has to be computationally efficient (in terms of time and space). Among other things, this is required to make experiments with long and complex sen-

tences feasible.

We have not been bothered by efficiency considerations yet, as SEM2SYN is fast enough for current purposes. This may change if the grammar grows. However, FUF provides a number of ways to control and optimize backtracking during unification, which may be explored when needed.

Acknowledgements

This paper benefitted from useful comments by an anonymous reviewer and the CLIN editors. The author would also like to thank Peter-Arno Coppen, Carlos Gussenhoven and Toni Rietveld.

References

Appelo, E., Leermakers, M. and Rous, J.(1993), Template-based generation of natural language expressions with controlled m-grammar, *IPO Annual Progress Report*, number 28, Eindhoven, The Netherlands, 131–138.

Bateman, J. A.(1994), KPML: The KOMET-Penman (multilingual) development environment, *Technical report*, Institut für Integrierte Publikations- und Informationssysteme (IPSI), GMD.

Bateman, J. A. and Hovy, E. H.(1991), Computers and text generation: Principles and uses, *in* C. Butler (ed.), *Computers and Texts: An applied Perspective*, Basil Blackwell, Oxford, England, 53–74.

Claassen, W.(1992), Aspects of automated natural language generation, *in* R. D. ad E. Hovy, D. Rösner and O. Stock (eds), *Generating Referring Expressions in a Multimodal Environment*, Springer, Berlin, 247–262.

De Smedt, K.(1990), IPF: an incremental parallel formulator, *in* R. Dale, C. Mellish and M. Zock (eds), *Current Research in Natural Language Generation*, Academic Press, London, 167–192.

De Smedt, K., Horacek, H. and Zock, M.(1996), Architectures for natural language generation: Problems and perspectives, *in* G. Adorni and M. Zock (eds), *Trends in Natural Language Generation: An Artificial Intelligence Perspective*, Springer, Berlin, 17–46.

Degand, L.(1993), Dutch grammar documentation, *Technical report*, Institut für Integrierte Publikations- und Informationssysteme (IPSI), GMD.

Elhadad, M.(1993), *FUF: The Universal Unifier - User Manual, version 5.2*, New York. Technical Report CUCS-038-91.

Elhadad, M. and Robin, J.(1996), An overview of SURGE: A reusable comprehensive syntactic realization component, *Technical Report 96-03*, Ben Gurion University, Department of Computer Science.

Elhadad, M. and Robin, J.(to appear), SURGE: a comprehensive plug-in syntactic realization component fot text generation, *Computational Linguistics*.

Elhadad, M., McKeown, K. and Robin, J.(1997), Floating constraints in lexical choice, *Computational Linguistics* **23**(2), 195–239.

Frawley, W.(1992), *Linguistic Semantics*, Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

Haeseryn, W., Romijn, K., Geerts, G., de Rooij, J. and van den Toorn, M.(1997), *Algemene Nederlandse Spraakkunst*, second edn, Martinus Nijhoff Uitgevers, Groningen.

Halliday, M.(1994), *An Introduction to Functional Grammar*, Edward Arnold, London.

Jackendoff, R.(1990), *Semantic Structures*, Vol. 18 of *Current Studies in Linguistics*, MIT Press, Cambridge, Massachussetts.

Kay, M.(1984), Functional unification grammar: A formalism for machine translation, *Proceedings of COLING-84*, ACL, Stanford University, 75–78.

Kempen, G. and Hoenkamp, E.(1987), An incremental procedural grammar for sentence production, *Cognitive Science* (11), 201–258.

Mann, W. and Matthiesen, C.(1983), NIGEL: A systemic grammar for text generation, *Technical Report ISI RR-83-105*, ISI.

Marsi, E.(1995), Intonation in a spoken language generator, *in* H. Strik, N. Oostdijk, C. Cucchiarini and P. Coppen (eds), *Proceedings 19*, Department of Language and Speech - Univerity of Nijmegen, 85–97.

Marsi, E.(to appear), *Prosody in concept-to-speech*, PhD thesis, Department of Language and Speech - Univerity of Nijmegen.

Meteer, M.(1989), The SPOKESMAN natural language generation system, *Technical Report 7090*, BBN Systems and Technologies.

Model, J.(1991), *Grammatische Analyse*, ICG Publications, Dordrecht.

Pollard, C. and Sag, I.(1994), *Head Driven Phrase Structure Grammar*, University of Chicago Press, Chicago.

Reiter, E.(1994), Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible?, *Proceedings of the Seventh International Workshop on Natural Language Generation (INLGW-1994)*, 163–170.

Reiter, E.(1995), NLG versus templates, *Proceedings of 7th European Workshop on Natural Language Generation*, Leiden, The Netherlands.

Reiter, E. and Dale, R.(1998), Building applied natural language generation systems, *Natural Language Engineering*.

Shieber, S.(1986), *An Introduction to Unification-Based Approaches to Grammar*, Vol. 4 of *CSLI Lecture Notes*, University of Chicago Press, Chicago.

Stede, M.(1995), Lexicalization in natural language generation: A survey, *Artificial Intelligence Review* (8), 309–336.

Teunissen, L.(1997), GENIUSS: Generative Implementation of Universal Semantic Syntax, *in* W. Hoepnner (ed.), *6th European Workshop on Natural Language Generation*, Gerhard-Mercator-Universität-GH Duisburg.

Winograd, T.(1983), *Language as a Cognitive Process*, Vol. I: Syntax, Addison-Wesley, Reading, Massachussetts.