Grammatical Analysis in a Spoken Dialogue System

Gosse Bouma, Rob Koeling,
Mark-Jan Nederhof and Gertjan van Noord

1. Introduction

Consider a computer system that you can call by phone, and that you can ask questions — in ordinary Dutch — concerning the time table of public transportation. For example, you might ask such a system the departure time of a train from Groningen to Amsterdam, such that you can arrive in Amsterdam at 10. Before proceeding to answer your query such a system will probably ask you at which date you want to travel, and whether you mean '10 in the morning' or '10 in the evening'. After such a clarification, it will then proceed to give you the desired information, e.g. that you could take the 7.37 train from platform 2b.

Such a system is capable of recognizing and understanding spoken language. It is furthermore capable of conducting a natural dialogue in order to obtain the information necessary for answering your questions. Moreover, the system is capable of producing spoken natural language in order to conduct this dialogue, and in order to produce the answer to your request for information. Such a system does not yet exist.

1.1. The OVIS2 *Spoken Dialogue Systems*

The NWO Priority Programme *Language and Speech Technology* is a research programme aiming at the development of spoken language information systems. Its practical goal is to develop a demonstrator of a system we just described. This demonstrator is called OVIS, Openbaar Vervoer Informatie Systeem (*Public Transport Information System*). In order to meet this practical goal, scientific contributions are envisaged in speech recognition, natural language processing, and dialogue management.

Apart from the practical goal of building the OVIS demonstrator, the NWO Priority Programme *Language and Speech Technology* is motivated by a number of scientific, cultural and economical goals.

An important scientific goal is to combine the insights of linguistics and phonetics on the one hand, and computer science on the other. It is expected that the technological requirements of the proposed demonstrator necessitates a closer study of language

Roel Jonkers, Edith Kaan and Anko Wiegel (eds.), *Language and Cognition 5.* Groningen, 1996.

'performance' rather than language 'competence'; therefore information-theoretic approaches (such as probabilistic techniques) will be investigated.

Furthermore, two — quite different — cultural goals are identified. Firstly, the programme aims at an integration of two different research paradigms, by confronting the experimental, corpus-based approach to language research (prominent in speech technology) with the 'knowledge-based' approaches in linguistic research and language technology.

The second cultural goal is to maintain the status of Dutch as a cultural language. It is expected that the status of Dutch will decrease if spoken language technology is developed to a further extent for languages such as English and German than it is for Dutch.

Finally, the Priority Programme is of economic importance. Without research into Dutch language technology it is not possible to develop Dutch information systems. The programme will support engineers from industry to bridge the gap between scientific research and commercial Dutch information processing systems.

The NWO Priority Programme 'Language and Speech Technology' is a five-year research programme. The programme is co-funded by Philips Corporate Research and KPN Research. Research is carried out by four different groups. Work on speech recognition is carried out at the University of Nijmegen; work on probabilistic natural language processing is carried out by a group at the University of Amsterdam; work on the pragmatic module and the dialogue manager is carried out by a group at the Institute for Perception Research at Eindhoven. Finally the work on grammar-based natural language processing is carried out by the authors of this paper, at the University of Groningen. The remainder of this paper will concentrate on the work on grammar-based natural language processing.

In the next section we will describe the architecture of the OVIS2 system. OVIS2 is a modularized version of the OVIS1 prototype. The OVIS1 prototype is a version of a German system developed by Philips Dialogue Systems in Aachen (Steinbiss et al 1993; Aust et al 1994), adapted to Dutch. In section 1.3. we then describe the problems that need to be solved in the natural language understanding module. The paper then continues by describing the components that are currently under construction as part of the natural language understanding module.

1.2. Architecture of OVIS2

The architecture of the OVIS2 prototype is given in figure 1. In this overview of the architecture, the arrows indicate the flow of information. Boxes are modules of the system. The idea is that the user produces an utterance which is input for the speech recognition module. The result of speech recognition is passed on to the linguistic processing component. The result of linguistic analysis is input for the pragmatic component, which passes on data to the dialogue manager. The dialogue manager checks whether enough information is available to consult the database, or whether further information needs to be requested from the user. Such requests and the answers to the user's query are articulated by means of the synthesis modules. The circled boxes in the center of the figure illustrate the datastructures that are maintained by the

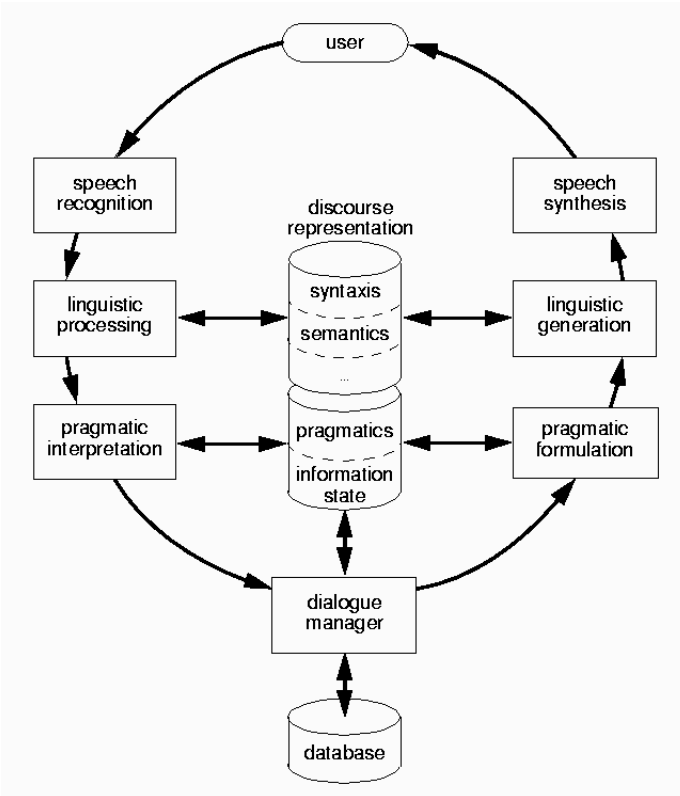

Figure 1: The Architecture of OVIS2.

different modules. For the motivation behind the chosen architecture we refer to Boves et al (1995) and Veldhuijzen van Zanten (1995).

1.3. Linguistic Processing in OVIS

In this paper we concentrate on the linguistic processing (NLP) component. This component receives its input from the speech recognizer, and passes the result of linguistic analysis — some kind of semantic representation — on to the pragmatic interpretation component.

The output of the speech recognizer is a *word graph* (cf. section 3.2.), which represents all different hypotheses for the spoken input from the user. Linguistic analysis of such a structure is more complicated than analysis for a single input sentence (cf. linguistic analysis for *written* input), as we will show later.

The interface between the linguistic analysis component and the pragmatic interpretation module is described in Veldhuijzen van Zanten (1995). This interface is defined by a special interface language, called the *Update Language*. Some examples will be provided later.

In the design of the linguistic processing component, the following problems must be addressed.

Firstly, a natural language analysis component will be faced with *ambiguity*. The combination of lexical and structural ambiguities often leads to an enormous amount of possible readings for an input utterance. In OVIS2 this problem is even more acute because of the use of word graphs. Techniques will have to be developed to be able to deal with large numbers of analyses, and to be able to choose the most appropriate reading from such a set of candidate analyses.

While the system must somehow deal with large numbers of analyses, many of which may eventually turn out to be useless, it must not make the mistake of overlooking useful analyses. This is the problem of *robustness*.

The linguistic analysis needs to be robust for three reasons. Firstly, it is quite difficult to anticipate in the grammar all linguistic constructions that might occur. This is the traditional problem for grammar-based NLP. Secondly, spoken language is full of hesitations, corrections, false starts etc. which are not always easy to detect. The third reason is that the utterance that was actually spoken is not guaranteed to be a path in the word graph, due to limitations of state-of-the-art speech recognition.

These observations indicate that *robustness* and *disambiguation* are two very important problems to be solved in the NLP component. A third problem can be added to this list: *efficiency*. Given the nature of the proposed application, it will be clear that the system is supposed to run in *real-time*, i.e. it is not supposed to leave the user waiting for the requested information. This requirement provides a further challenge for the NLP component.

1.4. Components of the NLP module

We currently assume that the following phases can be distinguished in the NLP module. Firstly, we foresee a pre-processing phase in which the word graph may be modified in order to repair hesitations, corrections, false starts, etc.

Secondly, this modified word graph is then input for the syntactic and semantic analysis phase. The *grammar* defines what the appropriate syntactic and semantic analysis should be for a given input. The *parser*, which is a device derived from the grammar, is capable of computing this analysis. The grammar and the parser are therefore two important components, and will be discussed in more detail in the remainder of this paper.

Finally, in case semantic analysis results in either too few or too many analyses, special action should be undertaken. If no semantic analysis was found, then the *robustness* component tries to come up with a partial semantic analysis on the basis of the partial results that were discovered by the parser. If many competing semantic analyses were produced then the *disambiguation* component should decide which analysis is the most appropriate candidate.

2. The Grammar

The grammar for OVIS2 contains the grammatical knowledge required to analyse a word graph and to determine what the meaning of the utterance corresponding to the word graph is.

Ideally, in order to construct such a grammar one would take a general-purpose computational grammar of Dutch, and adapt it to the current domain and application. Unfortunately, however, we do not know of any computational grammar for Dutch that could easily be adapted to the present task (which, among others, requires processing of spoken language, extensive coverage of locative and temporal expressions, and the construction of fine-grained semantic representations).

The OVIS-grammar is being developed especially for the purposes of this project. On the one hand this has the advantage that the grammar can be tailored to the specific requirements of the present project. On the other hand, we want to adopt general solutions as much as possible, as this increases the chances that the grammar can be used in other domains as well. Thus, in designing the grammar we seek a balance between short-term goals (a grammar which covers utterances typical for the OVIS-domain and is reasonably robust and efficient) and long-term goals (a grammar which covers the major constructions of Dutch in a general way).

The grammar currently covers some of the more common verbal subcategorization types (intransitives, transitives, verbs selecting a PP, and modal and auxiliary verbs), NP-syntax (including pre- and postnominal modification, with the exception of relative clauses), PP-syntax, the distribution of VP-modifiers, various clausal types (declaratives, yes/no and WH-questions, and subordinate clauses), all temporal expressions and locative phrases relevant to the domain, and various typical spoken language constructs.

2.1. The Grammar Formalism

From a linguistic perspective, the OVIS-grammar can be characterized as a constraint-based grammar, which makes heavy use of lexical information. The design of the grammar was inspired to a certain extent by Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag 1994).

The OVIS-grammar formalism is essentially equivalent to Definite Clause Grammar (DCG) (Pereira and Warren 1980). The choice for DCG is motivated by the fact that this formalism provides a balance between computational efficiency and linguistic expressiveness, and the fact that it is closely related to constraint-based grammar formalisms, such as HPSG, Categorial Unification Grammar, and Lexical Functional Grammar. Another important reason to choose DCG instead of a more restricted formalism such as context-free grammar, is the fact that DCG allows the kind of integration of syntax and semantics that is standard in constraint-based formalisms such as HPSG.

Grammar rules consist of a context-free skeleton to which feature-constraints are added. The context-free skeleton is important, as it ensures a reasonable level of processing efficiency and facilitates experimentation with different parsing techniques.

The central formal operation in constraint-based grammar formalisms is unification of (typed or untyped) feature-structures (Shieber 1986). The OVIS-formalism employs typed feature-structures in the definition of rules as well as lexical entries. During the construction of the parser, feature-structures are translated into Prolog terms. Because of this translation step, parsing can make use of Prolog's built-in term-unification, instead of the more expensive feature-unification.

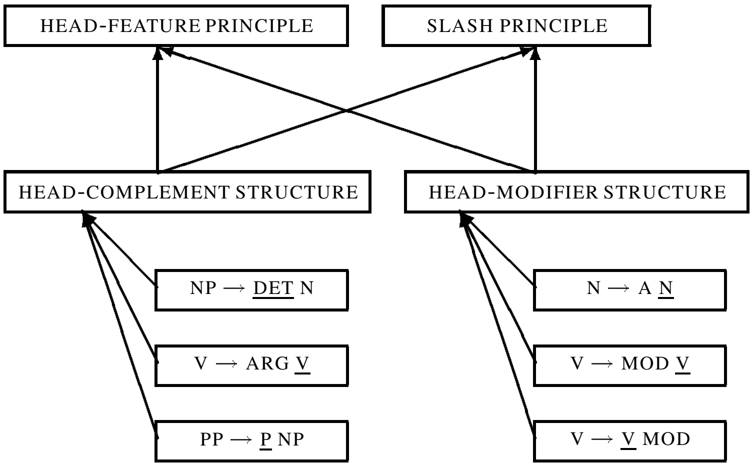

Figure 2: Part of the rule hierarchy

As in HPSG, generalizations about rules are captured by means of principles. For instance, most rules introduce a kind of *head-complement* or *head-modifier structure*. Both structures inherit from the *head-feature principle* (which states that the HEAD-features of the head-daughter and the mother must be unified) and the *slash-principle* (which governs the percolation of the feature SLASH, used to account for unbounded-dependencies). Figure 2 illustrates part of the rule hierarchy. An arrow from a box A to a box B indicates that A inherits information from B, where inheritance amounts to unification. In the rules shown in figure 2, the underlined daughter is the head of the rule. Note that we assume that DET is the head of NP. The rules V $\rightarrow$ ARG V, V $\rightarrow$ MOD V, and V $\rightarrow$ V MOD are abbreviations for a finite number of rules (ARG can be instantiated as NP, PP or V for instance, the possible instantiations of MOD depend on whether it precedes or follows its verbal head).

Note that the OVIS-grammar does not use the general rule-schemata of HPSG and thus there is no need for linear precedence statements. On the other hand, the requirement that all rules must have a context-free skeleton implies that the number of rules is larger than in most HPSG fragments. As the rule-set is structured as an inheritance network, this does not give rise to unacceptable redundancy.

As is the case in most lexicalist grammar formalisms, subcategorization is handled lexically. Thus, all major categories (verbs, nouns, prepositions, and determiners) have a feature SUBCAT, whose value is the list of complements they subcategorize for. Expressing this information lexically, instead of using more detailed syntactic rules, has the advantage that idiosyncratic subcategorization requirements need not be stated in the rules (such as the restriction that *denken* (*to think*), if it selects a PP-complement, requires this complement to be headed by *aan* (*about*); or the fact that *komen* (*to come*) may combine with the particle *aan* (the combination of which means *to arrive*)). Similarly, all constraints having to do with case marking and agreement can be expressed lexically. Finally, the semantic relation between a head and its dependents

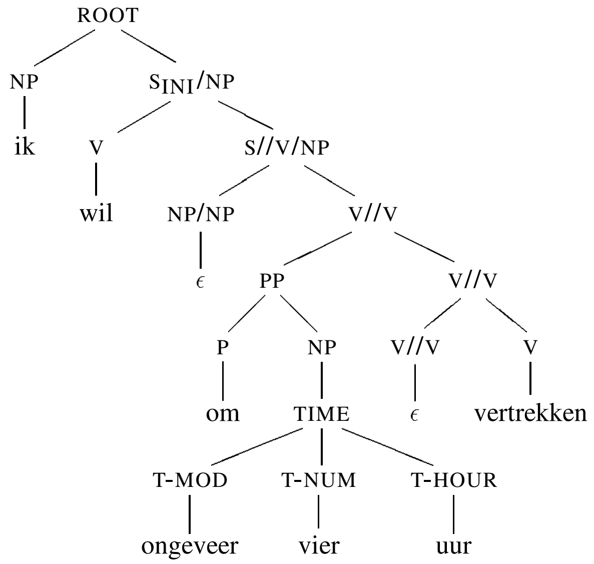

Figure 3: *"ik wil om ongeveer vier uur vertrekken"*

can be expressed lexically.

Figure 3 illustrates the grammatical analysis for *ik wil om ongeveer vier uur vertrekken* (*I want to leave around four o'clock*). The subject *ik* combines with a verb-initial clause missing an NP (indicated as $\text{S}_{\text{INI}}/\text{NP}$) to form a root clause. The finite verb *wil* combines with a sentential constituent missing an NP and a verb (where the verbal gap is indicated as //V). The idea is that *wil* may appear in this fronted position only if it combines with a sentence in which a verbal gap occurs which is unifiable with *wil*. This type of analysis resembles the transformational analysis in terms of *verb movement* and allows us to give a uniform treatment of main and subordinate clauses. Note that gaps are introduced by means of empty elements (i.e. NP/NP is the empty element corresponding with the fronted subject, and V//V is the empty element corresponding with the fronted verb). The temporal NP is analysed using a special grammar module (see below).

2.2. The Lexicon

Lexicalist grammar formalisms tend to store lots of syntactic information inside the lexicon. To avoid massive reduplication of identical information in the lexicon, the use of inheritance is essential. A fragment of the lexical inheritance hierarchy is shown in figure 4. The lexical entry for a verb such as *vertrekken* (*to leave*) is defined in terms of the templates *intransitive* and *iv-sem*; *intransitive* in its turn is defined in terms of a template *verb*, which defines the properties common to all verbs.

Lexical rules provide another means for capturing lexical generalizations. While

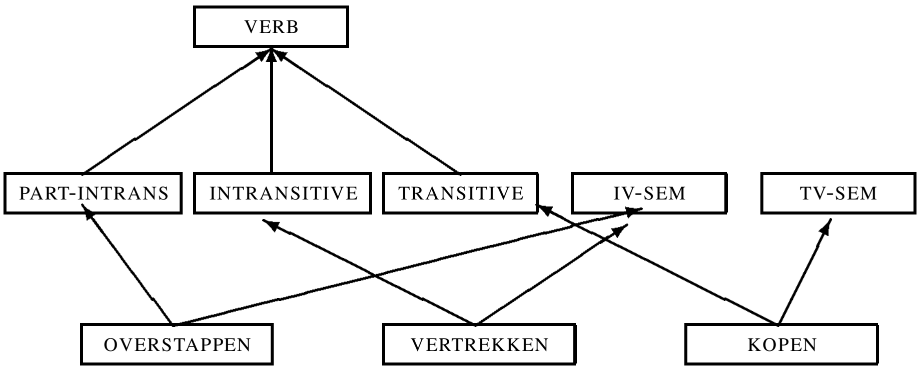

Figure 4: Part of the lexical hierarchy

inheritance can be used to express information present in various lexical entries succinctly, lexical rules are used to express relations that are of an implicational nature (if a lexical entry satisfies constraints C, it is valid to infer the existence of a lexical entry C'). At the moment, lexical rules are used to encode inflectional morphology and, to a limited extent, to account for unbounded dependencies.

2.3. The Grammar of Temporal Expressions

Temporal expressions occur very frequently in the OVIS domain. It is therefore essential that the grammar provides extensive coverage of these expressions. Some examples of temporal expressions of category NP covered by the grammar are presented in (1). These expressions can be combined with prepositions such as *op (on), om (at), voor (before), na (after)* to form a temporal expression of category PP.

(1)	a.	`drie januari`	*January, 3*
	b.	`de derde (januari)`	*the third (of January)*
	c.	`maandag, maandagmiddag`	*Monday, Monday afternoon*
	d.	`overmorgen`	*the day after tomorrow*
	e.	`drie uur, drie uur vijftien`	*three o'clock, three fifteen*
	f.	`kwart over drie`	*quarter past three*
	g.	`vijftien minuten over (drie)`	*fifteen minutes past (three)*

At the moment, the grammar of temporal expressions is accounted for in a separate module of the grammar. Thus, the general principles which hold for other parts of the grammar (such as the *head-feature principle* and the *slash principle*) need not apply within this module. We believe that such an approach is motivated by the fact that the grammar of temporal expressions contains many idiosyncrasies (such as the fact that *drie uur* combines a singular noun with a plural numeral, or the fact that *vijf over drie* combines two numerals and a preposition) which seem to be limited to this type of expression only. Note that this does not imply that no principled account of the syntax of temporal expressions could be given (see Flickinger (1995) for instance). However, as general principles of the grammar do seem to play only a minor role in the grammar of temporal expressions, we are satisfied with providing a few simple, but idiosyncratic, rules and lexical entries to account for these expressions.

2.4. Semantics

The output of the grammatical analysis is a semantic, linguistically motivated and domain-independent, representation of the utterance. This representation is translated into one or more domain-specific *updates*, which are passed on to the pragmatic interpretation module and dialogue manager (DM) for further processing. Below, we motivate our choice for *quasi logical forms* (QLF's) as semantic representation language and we discuss how these forms are translated into updates.

The semantic representation language

Predicate logic is often used to represent the meaning of sentences. Due to its long tradition in describing semantics of natural languages it is now a well-established and well-understood technique. Nevertheless, it often seems helpful to extend the predicate calculus with more advanced techniques, such as *generalised quantifiers* or *discourse markers*. For OVIS we use the QLF-formalism (Alshawi 1992), whose most important feature is that it makes extensive use of underspecification to represent ambiguities.

A property of artificial languages like predicate logic is that they are unambiguous. An ambiguous natural language utterance will therefore correspond with more than one expression in predicate logic, one for each reading of the utterance. The disadvantage of this approach is that for very ambiguous inputs, expensive computations must be carried out to compute all readings. The alternative adopted in the QLF-formalism is to represent ambiguity by means of underspecification and to postpone the computation of all possible readings to the disambiguation component.

A QLF is a *description* of one or more predicate logical formulas. While building a QLF for an utterance, information is added *monotonically* (Cooper et al 1994). Whenever information is added, the amount of underspecification will decrease and, consequently, the resulting QLF will describe a smaller set of logical formulas. After grammatical processing of an utterance is completed, several *knowledge sources* (such as an algorithm for generating quantifier scope or for finding referents for anaphoric expressions) may be used to further reduce the amount of underspecification.

In figure 5 we give the (simplified) QLF which the current grammar produces for the example in figure 3. Remember that the grammar manipulates feature-structures, so the QLF is represented as a feature-value matrix. Note that the arguments of the predicates *willen* and *vertrekken* are (generalized) quantifiers that are unscoped with respect to each other. Furthermore, we assume that verbs introduce an (existentially quantified) event-variable.

The QLF in figure 5 represents the domain-independent meaning of the utterance. If we want to reuse the present grammar for other domains, or for extensions of the present domain, it is essential to have representations that are not based on domain-specific assumptions. Representing ambiguity by means of underspecification is particularly important in the present setting, as the system must deal with highly ambiguous input (i.e. word graphs). Furthermore, it turns out to be the case that the computation of the

$$\left[\begin{array}{l} \mathit{pred}: \text{and} \\ \mathit{args}: \left\langle \left[\begin{array}{l} \mathit{pred}: \text{willen} \\ \mathit{args}: \left\langle \left[\begin{array}{l} \mathit{index}: \text{A} \\ \mathit{res}: \lambda\text{E.event(E)} \\ q: \text{exist} \end{array}\right], \left[\begin{array}{l} \mathit{index}: \text{I} \\ \mathit{res}: \lambda\text{K.name(K,ik)} \\ q: \text{exist} \end{array}\right], \right. \\ \qquad \left. \left[\begin{array}{l} \mathit{pred}: \text{vertrekken} \\ \mathit{args}: \left\langle \left[\begin{array}{l} \mathit{res}: \lambda\text{S.event(S)} \\ q: \text{exist} \end{array}\right], \text{I} \right\rangle \end{array}\right] \right\rangle \end{array}\right], \left[\begin{array}{l} \mathit{pred}: \text{at} \\ \mathit{args}: \langle \text{A}, \text{hour}(4) \rangle \end{array}\right] \right\rangle \end{array}\right]$$

Figure 5: (Simplified) QLF for 'Ik wil om ongeveer vier uur vertrekken'

domain-specific meaning of an utterance can be carried out without resolving all the sources of ambiguity in the QLF for that utterance. For instance, the scope of quantifiers seems to be to a large extent irrelevant for the computation of (domain-specific) *updates*, and thus a representation in which quantifier-scope is undetermined is sufficient for our purposes.

Domain-specific interpretation

The dialogue manager (DM) expects the linguistic module to produce so-called *updates* representing the domain- and context-specific meaning of the user-utterance. The DM keeps track of the information provided by the user by maintaining an *information state* or *form*. This form is a hierarchical structure, in which there are slots for the origin and destination of a connection, for the time at which the user wants to arrive or leave, etc. Updates specify the values of these slots. For example, the QLF in figure 5 is translated into the following update:

(2) `userwants.travel.origin.time.at.clock-hour.[=4]`

The first part of the update (`userwants...clock-hour`) is a path specifying the slot that is filled, the second part (`4`) is the value.

As the user responds to the system's questions, responses will usually not be a complete sentence, but an NP or PP, or a sequence of such constituents. In such cases, the QLF produced by the grammar will be severely underspecified. An update can be constructed, however, by adding information from the question to the semantic representation of the answer. The response in (3b), for instance, can be translated into the update in (3c) by taking into account the fact that this was a response to the question in (3a).

(3) a. `Naar welk station wilt u vanuit Groningen reizen?`
To which station do you want to travel from Groningen
b. `Amsterdam`
Amsterdam
c. `userwants.travel.destination.place[=Amsterdam]`

3. Parsing

As explained before, a user communicates with the OVIS system by means of spoken language over a telephone line. The analysis of a sentence uttered by the user first requires automatic speech recognition. In the ideal case, this consists in identifying a single list of words that were uttered by the user. The next step would be to parse this list of words, i.e. to recognize the list of words as a correct sentence with a certain structure, i.e. *parse*. Ideally, only one such parse can be assigned to the sentence. The semantics, i.e. the QLF can be obtained from the parse. This QLF is subsequently translated into an *update* and passed on to the other modules of the OVIS system.

There are three obstacles to straightforward application of the syntactic analysis as described above:

- The speech recognizer may not be able to uniquely identify the list of words uttered by the user.
- The sentence uttered by the user may not be grammatical.
- Sentences may be ambiguous, i.e. there may be several parses for a single sentence.

In the following, we will first explain what the parsing process looks like if we do not take these problems into account. Then we will consecutively discuss these problems and the way we deal with them.

3.1. Introduction to parsing

In the ideal case, the input of the parser is a single list of words, such as

(4) `Ik wil om ongeveer vier uur vertrekken`
I want to leave around four o'clock

and the intended output is a single parse, such as the one in figure 3, which has been constructed according to the grammar (discussed in the previous section).

There are many *parsing algorithms* which can be applied in order to find such parses, given an input and a grammar. A simple parsing algorithm is the CYK algorithm (Hopcroft and Ullman 1979). It is based on the divide-and-conquer strategy: the problem of finding a parse for some input is reduced to finding subparses for the substrings of the input first; these are then combined. More specifically, we start with finding subparses for the smallest units in the input (lexical entries), collections of these subparses are then combined to form larger parses for larger chunks of the input, until the parse for the complete input is found.

In the example, we start by labeling the words in the sentence with categories NP, V, P, T-MOD, T-NUM, T-HOUR, V, by looking up the words in the lexicon. Then, the result is used to construct small subparses, such as the subparse covering the input 'ongeveer vier uur' in figure 3, which has category NP in its root. This subparse is later combined with a subparse covering adjacent input 'om' in order to construct the larger subparse covering the input 'om ongeveer vier uur' and having PP as its root category, and so on, until the complete parse in figure 3 has been found. Such a combination of sub-parses is allowed only in case there is a grammar rule that licenses the combination. In the running example we can combine P and NP because there is a grammar rule PP $\rightarrow$ P NP.

CYK is a bottom-up algorithm. Other parsing algorithms, such as top-down parsing, are more selective: the kinds of subparse that are constructed from a certain input position onward are dependent on subparses already found to the left of that input position. An example is Earley's algorithm (Earley 1970). Other examples can be found in e.g. Nederhof (1994).

When complex categories are used (as is the case in OVIS), the parsing problem becomes even more complicated and further distinctions can be made between different top-down strategies and different bottom-up strategies.

A further distinction between parsing algorithms can be made by identifying in which order the input string is traversed. Many algorithms proceed from left to right, but it has been observed that a bidirectional flow of control, as in *head-driven parsers* (Kay 1989; van Noord 1993; Sikkel 1993; Nederhof and Satta 1994), has practical advantages (Bouma and van Noord 1993).

Although the end result of parsing is always the same, the speed of different parsing algorithms may differ significantly. This is a relevant issue for a real-time system such as OVIS. For this reason we are currently developing implementations of a number of different parsing algorithms. These implementations are then compared to find out which parser works best for the OVIS grammar and input which is typical of the application.

For the construction of parsers from the grammar, we use a parser generator that is capable of detailed analyses of the grammar. For example, it performs a separation of grammar symbols into a part that is required for finding the correct parses, and a part that is merely required for finding the semantics of correct parses. During syntactic analysis, the latter part only needs to be applied for complete parses, and not for all subparses, some of which may turn out to be useless later. This may speed up the syntactic analysis.

3.2. Word graphs

In practice, speech recognition does not allow unique identification of *the* list of words that a person actually utters at the other end of a telephone line. There are a number of reasons why this is not possible. First, there are technical limitations in state-of-the-art speech recognition for connected speech, which makes that full recognition of all words is unattainable for all speakers regardless of dialects and foreign accents, the pitch of their voices, how well they articulate, etc. The second reason is the varying quality of telephone connections, which may blur the spoken text to some degree. Also background noise, such as slamming doors, noisy pets, and conversations in the background, constitute problems for speech recognizers. Finally, some sequences of sounds may actually be ambiguous, in the sense that they may represent multiple sequences of words. The reason that people are usually unaware of such ambiguity is because we unconsciously disambiguate by means of syntax and other kinds of non-phonetic information.

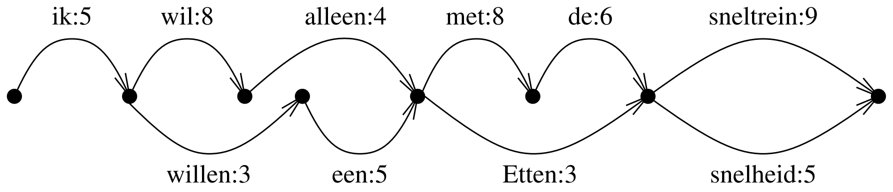

Figure 6: An example of a word graph.

These considerations have led to the notion of *word graph* (Oerder and Ney 1993). A word graph, produced by an automatic speech recognition device, is a succinct representation of all alternative sequences of words where the device cannot identify a unique sequence. Most parsing algorithms for linear input can easily be generalized to deal with word-graphs (Bar-Hillel, Perles, and Shamir 1961; Lang 1989; van Noord 1995).

An example of such a graph is given in figure 6. The nodes represent the points in time. The left-most node represents the point in time where a sentence may start, the right-most node represents the point in time where the sentence may end.

All paths from the initial to the final node represent one alternative sentence. The edges in the graph are labeled with words, but also with *scores*, which indicate some degree of certainty with which words are recognized. The lower the score, the likelier it is that a word should represent the sounds perceived between the corresponding points in time. For a sequence of words, the scores can be summed. The result is a degree of likelihood that that sequence of words corresponds with the actual utterance.

One could try to find the path in the graph from the initial node to the final node which has the lowest score, and take this to be the intended sentence. However, as we have argued before, non-phonetic information such as syntax and world knowledge is often needed to find the correct sentence, and therefore selection of one particular sentence based on phonetic scores alone may be premature.

In the example of figure 6, the path corresponding with the 'sentence'

(5) `Ik willen een Etten snelheid`
I want(plural) a Etten speed

has the lowest score, yet it is unlikely that this is the sentence uttered by the user, for obvious reasons.

Therefore, the other sentences should not be discarded before syntactic and semantic considerations have been taken into account.

In the running example there are 8 paths in the word graph. Only two are more or less syntactically correct:

(6) `Ik wil alleen met de sneltrein`
I want only with the inter-city

(7) `Ik wil alleen met de snelheid`
I want only with the speed

The second has the lowest score, viz. $5+8+4+8+6+5 = 36$, and may be taken as the intended sentence, unless later phases of the analysis (e.g. pragmatic analysis) prefer the first, which is likely in this case.

3.3. Robust parsing

As a first approximation, we suggested in the previous section that the analysis of the input should restrict itself to the sentences in the word graph that could be processed by the syntactic analysis. Strictly speaking, syntactic analysis means assigning a structure to a syntactically correct sentence. In this section we have to refine this idea. The reason is that typical sentences that occur in spoken dialogues may *not* be syntactically correct. Since in everyday human-human conversations, syntactic errors hardly ever disrupt the normal course of dialogues, one would like the OVIS system to be equally robust against any kind of syntactic error in the sentences spoken by the user.

More specifically, if the grammar does not assign an analysis to a given input, then the system should still try to construct some form of parse which allows a semantic value to be computed. This semantic value should in the ideal case correspond with the intentions of the user, as expressed in the ungrammatical sentence (note that robustness concerns both ill-formed input and well-formed input which our grammar fails to recognize).

The realisation of this idea can be performed by three mechanisms, which may be combined in order to obtain the desired result:

- The grammatical constraints are relaxed.
- The input is changed in order to satisfy the grammar.
- Partial sentences are recognized.

A typical application of the first mechanism is the treatment of mismatches of agreement. For example,

(8) `Ik wilt naar Amsterdam`
I wants to go to Amsterdam

is a grammatically incorrect sentence, because there is no agreement between the subject and the verb. By relaxing this constraint, the analysis of the input can proceed as if the correct verb form were substituted. A disadvantage of this approach is that unwanted analyses now may be derived as well.

Another typical source of violation of grammaticality is the phenomenon of *hesitations*. A typical example is

(9) `Ik wil naar ...eh...naar Groningen`
I want to go to…hm…to Groningen

The user hesitates to mention the place of destination, and when he is ready to continue the sentence, he repeats some of the words mentioned before (‘naar’ in the example).

Such errors may be treated by changing the input. Techniques for accomplishing such repairs are proposed in Carter and Rayner (1994) and Carter (1994). Since the input for the syntactic analysis in our case is the word graph, we realise this by adding

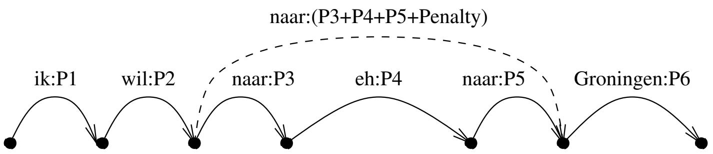

Figure 7: Adding an edge according to a rewrite rule.

some edges. We give an example for the phenomenon of hesitations. The desired correction is described by the following *rewrite rule*.

w `eh` $w \quad \rightarrow \quad w$

In words, this rewrite rule says that an occurrence of a string w, which consists of arbitrary words, followed by some sign of hesitation (represented by `eh`), followed by the same string w, can be changed into the string w. An example of the application of this rule is given in figure 7. Many other forms of ungrammaticality can be corrected by means of rewrite rules.

Some input from the user may not at all be intended to be a complete sentence: it may consist of phrases or merely partial phrases. For example, if the system asks

(10) `Waar wilt u heen?`
Where do you want to go to?

the user may answer with any of the following utterances, none of which is a sentence:

(11) a. `Delft`
b. `naar Delft`
c. `Delft om negen uur`
d. `naar Delft om negen uur`
to Delft at nine o'clock

It would be fruitless to try to turn these answers into full sentences in all possible ways. Instead, it obviously suffices that the system should parse the answer as a phrase or partial phrase, then compute the semantics, and finally complete this semantics to be the intention of the user, using the elements of the question.

In the example (11a), the syntactic analysis should merely recognize 'Delft' as the name of a station and translate it into the internal name for this station. Subsequent combination of this data with the meaning of the question then yields some semantics reflecting the intention of the user to go to Delft (and not to *depart* from Delft, for example).

The above problem can also be solved by designing the grammar in such a way that the language that is described is not merely the language of all correct and complete Dutch sentences, but the language of all relevant answers to questions in the application domain, in colloquial Dutch, including (partial) phrases.

More interesting is the treatment of cases where the input consists mostly of uninterpretable subsequences of words, with only a few clear phrases, for example

(12) `de naar een vanuit Amsterdam uur`
the to a from Amsterdam o'clock

The only information discernible in this sentence is that the user wants to depart from Amsterdam. The remainder of the intentions of the user may not become clear from the utterance as the speech recognizer has processed it (we assume other paths in the word graph are equally uninterpretable). The OVIS system may however use the partial information obtained thus far and continue by requesting the desired time of departure, the place of destination, etc.

In all cases where paths in the word graph cannot be recognized as grammatical sentences, the score of the paths should be incremented. This is done under the assumption that the more we change to the structure of sentences as we receive them through the speech recognizer, the larger the chances are that the semantics we eventually derive is inconsistent with the intentions of the user.

For example, in figure 7 the newly added edge has a score which is not merely the sum of the edges that it was derived from, but an extra penalty is added, because this edge is a derived edge, not an original one.

3.4. Ambiguous sentences

As we have remarked before, word graphs may contain several sentences that are grammatical, and some that are ungrammatical, but allow easy correction. In practice, additional ambiguity is introduced by the grammar. The use of underspecified semantic representations (QLF) can be applied only to a limited extent to reduce ambiguities.

A typical example is *PP-attachment*:

(13) `Ik wil naar Hengelo reizen om zes uur`
I want to travel to Hengelo at six o'clock

The phrase `om zes uur` may refer to `wil` (the desire to go to Hengelo is restricted to six o'clock) or to `reizen` (the travelling is to take place at six); even more interpretations are possible.

The consequence of such ambiguity is that several parses need to be constructed, only some of which will prove to be interpretable in later phases of analysis. In the above example, the first reading of the sentence will probably not fit in any pragmatic framework, and will therefore be rejected.

At the current moment it is not clear whether such ambiguity can and should already be solved during syntactic analysis. One conceptual objection against this solution is that it undermines the modular structure of the system. This is because the syntactic module would need to be contaminated with pragmatic information. In other application domains where different pragmatic considerations prevail such a syntactic module would then no longer be applicable.

4. Status of the work

Currently we have implemented a preliminary version of the OVIS grammar including semantic analysis and a component to translate semantic formulas into updates. Furthermore a number of different parsers have been implemented. Each of these parsers is capable of treating word graphs. An experimental robustness component has been implemented which produces promising results.

The disambiguation component is under construction. The future disambiguation component will take into account the question that was asked by the system.

Furthermore detailed improvements will be made to the grammar and to the component which produces updates from semantic formulas.

Within one year the NLP components will be integrated with the other components of the system.

Acknowledgements

The authors are supported by the Dutch Organization for Scientific Research (NWO), under grants 305-00-801, 305-00-802 and 305-00-803. An anonymous reviewer provided us with valuable comments that we have used to improve the text.

References

Alshawi, H. (ed.) (1992). *The Core Language Engine*. ACL-MIT press.

Aust, H., M. Oerder, F. Seide, and V. Steinbiss (1994). 'Experience with the Philips automatic train timetable information system'. In *2nd IEEE Workshop on Interactive Voice Technology for Telecommunications Applications*, Kyoto Japan.

Bar-Hillel, Y., M. Perles, and E. Shamir (1961). 'On formal properties of simple phrase structure grammars'. *Zeitschrift für Phonetik, SprachWissenschaft und Kommunicationsforschung*, 14, 143–172. Reprinted in Bar-Hillel's Language and Information – Selected Essays on their Theory and Application, Addison Wesley series in Logic, 1964, 116-150.

Bouma, G. and G. van Noord (1993). 'Head-driven parsing for lexicalist grammars: Experimental results'. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht.

Boves, L., J. Landsbergen, R. Scha, and G. van Noord (1995). *Language and Speech Technology*. NWO Den Haag. Project plan for the NWO Priority Programme 'Language and Speech Technology'.

Carter, D. (1994). Chapter 4: Linguistic analysis. In M-S. Agnäs, H. Alshawi, I. Bretan, D. Carter, K. Ceder, M. Collins, R. Crouch, V. Digalakis, B Ekholm, B. Gambäck, J. Kaja, J. Karlgren, B. Lyberg, P. Price, S. Pulman, M. Rayner, C. Samuelsson, and T. Svensson, (eds.), *Spoken Language Translator: First Year Report*. SICS Sweden / SRI Cambridge. SICS research report R94:03, ISSN 0283-3638.

Carter, D. and M. Rayner (1994). 'The speech-language interface in the spoken language translator'. In *Proceedings Twente Workshop on Language Technology 8*, University of Twente, The Netherlands.

Cooper, R., R. Crouch, J. van Eijck, C. Fox, J. van Genabith, J. Jaspars, H. Kamp, M. Pinkal, M. Poesio, and S. Pulman (1994). Fracas deliverable d8: Describing the approaches. Technical report, Centre For Congitive Science, Edinburgh. Available by ftp from ftp.cogsci.ed.ac.uk, pub/FRACAS/.

Earley, J. (1970). 'An efficient context-free parsing algorithm'. *Communications of the ACM*, 13(2), 94–102, February.

Flickinger, D. (1995). Pieces of time: Toward the syntax of temporal expressions in english. Talk presented in Groningen, October, 31, 1995.

Grosz, B., K. Sparck Jones, and B. Lynn Webber, (eds.) (1986). *Readings in Natural Language Processing*. Morgan Kaufmann.

Hopcroft, J.E. and J.D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Kay, M. (1989). 'Head driven parsing'. In *Proceedings of Workshop on Parsing Technologies*, Pittsburg.

Lang, B. (1989). 'A generative view of ill-formed input processing'. In *ATR Symposium on Basic Research for Telephone Interpretation (ASTI)*, Kyoto Japan.

Nederhof, M.J. (1994). *Linguistic Parsing and Program Transformations*. Ph.D. thesis, Katholieke Universiteit Nijmegen, Nijmegen.

Nederhof, M.J. and G. Satta (1994). 'An extended theory of head-driven parsing'. In *32th Annual Meeting of the Association for Computational Linguistics*, New Mexico State University.

van Noord, G. (1993). *Reversibility in Natural Language Processing*. Ph.D. thesis, University of Utrecht.

van Noord, G. (1995). 'The intersection of finite state automata and definite clause grammars'. In *33th Annual Meeting of the Association for Computational Linguistics*, MIT Boston.

Oerder, M. and H. Ney (1993). 'Word graphs: An efficient interface between continuous-speech recognition and language understanding'. In *ICASSP Volume 2*, pages 119–122.

Pereira, F.C.N. and D. Warren (1980). 'Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks'. *Artificial Intelligence*, 13.

Pollard, C. and I. Sag (1994). *Head-driven Phrase Structure Grammar*. University of Chicago / CSLI.

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

Sikkel, K. (1993). *Parsing Schemata*. Ph.D. thesis, Twente University, Enschede.

Steinbiss, V., H. Ney, R. Haeb-Umbach, B.H. Tran, U. Essen, R. Kneser, M. Oerder, H.G. Meier, X. Aubert, C. Dugast, D. Geller, W. Höllerbauer, and H. Bartosik (1993). 'The Philips research system for large-vocabulary continuous-speech recognition'. In *Proceedings EUROSPEECH*, pages 2125–2128, Berlin.

Veldhuijzen van Zanten, G. (1995). The interface between the natural-language interpreter and the dialogue manager in OVIS2. Priority Programme Language and Speech Technology, document number 15. Not publicly available.