Algorithms for LINGUISTIC PROCESSING

bcn

PIONIER Project Proposal

Gertjan van Noord

Algorithms for Linguistic Processing

Gertjan van Noord

Graduate School for Behavioral and Cognitive Neurosciences
Alfa-informatica
P.O. Box 716
NL 9700 AS Groningen
`vannoord@let.rug.nl`
http://www.let.rug.nl/~vannoord/alp

Contents

Introduction

Algorithms for Linguistic Processing is a research proposal in the area of *computational linguistics*. The proposal focuses on problems of *ambiguity* and *processing efficiency* by investigating *grammar approximation* and *grammar specialization* techniques. In this section we will try to explain for a broader audience what computational linguistics is, and what the current proposal is all about.

What is computational linguistics?

Linguistics is concerned with the study of natural languages such as Dutch, English, Korean, Nepali, Tagalog etc. Linguists hypothesise models ('grammars') of such languages which indicate the form utterances in the language can have. A grammar of English, for instance, will account for the fact that the first example below is an English sentence, whereas the second is not, even though it is made up of the very same English words:

(1) a. The small man loves the tall lady
    b. The lady tall man loves small the
    c. The tall lady loves the small man

Such grammars also describe the relation between *form* and *meaning* of the utterances in a given natural language. The grammar must distinguish the meaning of sentence (1-a) from the meaning of sentence (1-c). For the non symmetric verb *love* the difference in syntactic order corresponds not only to a difference in grammatical function (such as subject, direct object), but also to a difference in semantic roles (e.g. who is loved, and who is loving). Syntactic analysis will therefore determine the semantic interpretation.

In linguistics, grammars can play various roles. For example in theoretical linguistics they are sometimes considered as models of the knowledge native speakers have of their language. In computational linguistics, another perspective is taken: computational linguistics is about the *computation* of the relation between form and meaning in natural language. Thus, rather than focusing on *what* the relation between form and meaning is, computational linguists want to know *how* this relation can be computed. Suppose we have a grammar of Dutch and we are given a Dutch sentence, how can we compute its corresponding meaning? And in the reverse direction, suppose we are given a certain meaning, how can we compute a Dutch or an English sentence expressing that meaning?

Thus, computational linguistics is concerned with the way in which natural language is or can be processed.

Why study computational linguistics?

Why do we want to know how the relation between form and meaning can be computed? We identify three motivations to study computational linguistics.

As humans, we process language very easily many times a day in our normal communications with each-other. Hence, in our terminology, we are able to compute a meaning for a given utterance (since we understand each-other), and to compute an utterance for a given meaning (since we can express our thoughts). We want to know how humans perform this task, in order to further our understanding of natural language. In this sense, computational linguistics is part of linguistics and cognitive science.

A second motivation to study computational linguistics is of a more practical nature. If we know how to compute this relation between form and meaning, then we can write computer programs which perform this computation. Such computer programs will make a broad set of interesting natural language applications possible: spoken information systems, machine translation systems, natural language interfaces, and many others. One of the results of the NWO Priority Programme on Language and Speech Technology is a spoken dialogue system for public transport information. The system is accessible by telephone. A caller can request (in ordinary Dutch) time-table information for all Dutch train connections. The system operates automatically: if all goes well, no human interaction is required. In such a system the computer analyses the utterances of a user in order to find out what connection is being requested. Furthermore, natural language synthesis is used to produce further questions (for instance if not all information is available for a database lookup yet), and to produce the resulting connection, once the database has been consulted. From this perspective, computational linguistics is an engineering science (the term *human language technology* is sometimes used for this type of work).

The third motivation to study computational linguistics is theoretical: we are interested in the computation of the relation between form and meaning for its own sake. This computation has interesting formal properties and relates in interesting ways to theoretical aspects of the theory of computing in general. Construed in this way, computational linguistics is closely related to mathematical linguistics and theoretical computer science.

Two problem areas in computational linguistics

The proposal *Algorithms for Linguistic Processing* focuses on two crucial problem areas in computational linguistics: problems of *processing efficiency* and *ambiguity*. For the problem of efficiency we propose to investigate *grammar approximation* techniques, whereas a number of *grammar specialization* techniques are proposed for the ambiguity problem.

Efficiency and Grammar Approximation. The first problem, efficiency, is posed by the fact that the grammars which are typically hypothesised by linguists are unattractive from the point of view of computation. For a given grammatical formalism, computational efficiency is expressed by stating the (maximum) number of steps required to compute the analysis of a sentence of length n, where n is the number of words. In the simple case, the number of steps will increase proportionally with n. In such cases the complexity is said to be linear. For more complex grammatical formalisms, the number of steps will increase faster than n (e.g. it increases proportionally with n^3 for so-called 'context-free grammars'). For certain even more powerful grammatical formalisms it can be shown that no upper-bound to the number of steps required to find an analysis can be given. The human language user, however, seems to process in linear time; humans understand longer sentences with no noticeable delay. This implies that neither context-free grammars nor more powerful grammatical formalisms are likely models for human language processing. An important issue therefore is how the linearity of processing by humans can be accounted for. For this purpose we will explore *grammar approximation* techniques.

Grammar approximation techniques approximate a given (powerful) grammar by means of devices of a much simpler form: finite-state devices. Finite-state devices are a type of formalism that has been studied extensively in mathematics and computer science. They are known to allow very efficient processing (in linear time). It is also known that they are incapable of treating certain linguistic constructions in full generality. But interestingly, at least some of the constructions that cannot be treated with finite-state devices are also difficult for humans.

For example, constructions involving center-embedding are very hard to process for humans, but are regarded as grammatical by linguists. In English particle verb constructions, the particle can either precede or follow the direct object (*put the book down / put down the book*). If the direct object contains a relative clause, and the particle follows the direct object, then the examples become very hard to understand. For finite-state devices it is equally impossible to process these sentences. This suggests that finite-state devices could offer language models adequately accounting for the efficiency of human language processing.

(2) a. I called the man who put the book that you told me about down up
b. The man who the boy who the students recognised pointed out is a friend of mine
c. The man the boy the students recognised pointed out is a friend of mine
d. The rat the cat the dog chased bit ate the cheese

(3)

De	de	de	oude	schuur	bewonende	boer	toebehorende	kat	haat	ratten
The	the	the	old	shed	living-in	farmer	is-owned	cat	hates	rats

The cat owned by the farmer living in the old shed hates rats

Grammar Specialization. A further problem is posed by *ambiguity*. Many words are ambiguous: a dictionary lists various meanings for a single word. For instance, the Dutch word *zagen* can be the past tense form of the verb *zien* (to see) or the present tense form of the verb *zagen* (to saw). For this reason, a computer program analysing the sentence in (4-a) will need to decide which reading of *zagen* was intended, in order to be able to come up with the appropriate meaning. Ambiguities also arise in certain structural configurations. In (4-b), for instance, the prepositional phrase *in Vietnam* could be attached both to *de oorlog* or to *luisteren*. Both examples are taken from the Eindhoven corpus (den Boogaart, 1975).

(4) a. Mijn vader zagen we niet meer
My father saw we not anymore
We don't saw our father anymore
We didn't see our father anymore

b. Zij luisteren naar de bezwaren tegen de oorlog in Vietnam[, ...]
They listen to the arguments against the war in Vietnam
They listen to the arguments against the Vietnamese war
They listen in Vietnam to the arguments against the war

In many contexts the unintended readings are unlikely or even ridiculous, but it is very difficult to spell out how information concerning the discourse, context, situation and knowledge of the world enforce a particular reading. Therefore, a computer program will come up with such ridiculous readings nonetheless. The phenomenon occurs very frequently for even the most innocent examples. For longer sentences, hundreds or even thousands of readings arise. An important problem therefore is disambiguation: how can we quickly rule out the unintended readings?

We propose to tackle this disambiguation problem by investigating a variety of *grammar specialisation* techniques. These techniques optimise a given grammar on the basis of corpora of representative linguistic behaviour. Such corpora will contain implicit knowledge concerning situations and contexts which can be extracted by means of statistical techniques in order to be helpful in disambiguation. Such specialisation techniques should be able to conclude that certain readings of lexical entries, and certain combinations of readings, are unlikely in certain contexts.

In the example (4-a) above, each reading could be the right one. However, in the absence of any further information, it seems that our best 'bet' is to guess that the second interpretation holds, since fathers are not sawed very often. In this perspective, the use of statistics is an aid to reasoning in circumstances where not enough information is available to infer which reading *must* have been the correct one.

Innovative aspects

Progress in computational linguistics will not only be important in terms of improving our understanding of human language, but it will also have an important effect by furthering human language technology.

It will be possible to develop natural language interfaces which will drastically improve the accessibility of large amounts of information (especially for people without computer training). It will enable and improve linguistic applications such as grammar checkers, dictation systems, language instruction, documentation systems and linguistic aids to the handicapped. Human language technology is one of the key actions of Fifth Framework Programme of the European Community for research, technological development and demonstration activities.

An innovative aspect of the proposal is that it focuses on the Dutch language, and hence on Dutch linguistics and language technology. A recent overview on Dutch Language and Speech Technology conducted by *de Nederlandse Taalunie* (Bouma and Schuurman, 1998) reports that there are many fewer language technology resources available for Dutch (as compared to English). It is important that language technology is developed for Dutch in addition to the current developments for languages such as English, German and French. Language technology applications such as those mentioned above are cultural bonuses that should accrue not only to the speakers of majority languages.

The project aims furthermore at significant spin-offs. The proposed project will devote resources to extending existing Dutch grammars to experiment with the proposed techniques and to test the hypotheses. An extensive Dutch grammar in the public domain will be a major contribution to Dutch computational linguistics and to the international community.

Moreover, we propose to apply some of the innovative techniques in a linguistic research tool for searching bare text-corpora (called **lgrep**). This application is capable of searching text corpora (including arbitrary Dutch texts on the Internet) on the basis of *linguistic* criteria. It extends existing search tools with the possibility to specify search patterns including linguistic criteria such as *part-of-speech* labels (such as *noun, verb, preposition, etc.*), major syntactic category (*noun phrase, verb phrase, subordinate sentence, etc.*), and grammatical relation (*subject, direct-object, specifier, etc.*). Such a tool would be useful for researchers working with corpora such as researchers in linguistics, applied linguistics, comparative literature and communication studies, but perhaps also as an extension of traditional grammars as used by language learners, enabling them to obtain example sentences of particular linguistic constructions upon request.

A successful implementation of *Algorithms for Linguistic Processing* will not only provide new insights concerning the way in which natural language is processed, but it will also provide new techniques which are crucial for human language technology, in particular for Dutch.

Chapter 1

Background and Motivation

1.1 Computational Linguistics

Theoretical linguistics has developed extensive and precise accounts of the grammatical knowledge implicit in our use of language. It has been able to adduce explanations of impressive generality and detail. These explanations account for speakers' discrimination between different linguistic structures, their ability to distinguish well-formed from ill-formed structures, and their ability to assign meaning to such well-formed structures. *Grammars* are hypothesised which model the well-formed utterances of a given natural language and the meaning representations which correspond with these utterances.

The smaller and younger field of computational linguistics has also been successful in obtaining results about the computational processing of language. These range from descriptions of dozens of concrete algorithms and architectures for understanding and producing language (parsers and generators), to careful theoretical analysis of the underlying algorithms. The theoretical analyses classify algorithms in terms of their applicability, and the time and space they require to operate correctly. The scientific success of this endeavour has opened the door to many new opportunities for applied linguistics.

However a number of important research problems have not been solved. An important challenge for computational accounts of language is the observed *efficiency* and *certainty* with which language is processed. The efficiency challenge is both theoretical and practical: grammars with transparent inspiration from linguistic theory cannot be processed efficiently. This can be demonstrated theoretically, and has been corroborated experimentally. In current practice, such grammars are recast into alternative formats, and are restricted in implementation. Effectively, large areas of language are then set aside.

The *certainty* with which language is processed is not appreciated generally. But careful implementation of wide-coverage grammars inevitably results in systems which regard even simple sentences as grammatically ambiguous, even to a high degree. The computational challenge is to incorporate disambiguation into processing.

There are two central leading hypotheses of the project. We shall explore *approximation* techniques which recast theoretically sound grammars automatically into forms which allow for efficient processing. The hypothesis is that processing models of an extremely simple type, namely finite automata, can be employed. The use of finite automata leads to interesting hypotheses about language processing, as we will argue below.

Second, we test the hypothesis that *certainty* can be accounted for—at least to some extent—by incorporating the results of language experience into processing. This will involve the application of machine learning techniques to grammars in combination with large samples of linguistic behavior, called corpora. Such techniques will ensure that a given utterance, which receives a number of competing analyses if considered in isolation, will receive a single analysis if the relevant context and situation are taken into account.

The project aims furthermore at significant partial results. In order to test its processing claims, large scale grammars of some theoretical ambition must be tested. While these exist now for English, the project will devote resources to extending existing Dutch grammars to further test the claims. An extensive Dutch grammar in the public domain would be a major contribution to Dutch computational linguistics and to the international community. Second, the processing techniques and concrete implementations are technology which directly enables a number of interesting applications in spoken language information systems, language instruction, linguistic research, grammar checking, and language aids to the disabled.

Naturally there have been attempts to process theoretically well-informed grammars using a range of grammar theories. For example, the Alvey project implemented a wide coverage Generalized Phrase Structure Grammar (GPSG) (Briscoe et al., 1987; Grover, Carroll, and Briscoe, 1993; Gazdar et al., 1985). The grammar illustrates both difficulties, inefficient processing and ambiguity.

The Alvey grammar eschews the abstract formulation favoured by contemporary theorists. It is purely context-free in form, and makes use of a small number of mostly atomic-valued features.[1] Most GPSG grammar principles are not represented directly at all; rather, the grammarian is required to keep these in mind. In particular, the grammarian should write no rules or lexical entries in violation of the principles, but there is no mechanism for the grammarian to write principles which the parser will attend to. So on the one hand the grammar makes concessions to processing difficulties.

Massive ambiguity on the other hand could not be circumvented. For longer sentences hundreds and even thousands of readings are associated with a single sentence. One of the test-sentences of the Alvey NL Tools grammar is the question *In which abbey or message with which he agrees did he see the crazy anxious abbot who was not appearing to see the message with which kim agrees?* The sentence is peculiar because the lexicon was minimal, but its peculiarity does not explain how the grammar assigns it 2,736 distinct readings, each of which is motivated linguistically. One of the parse-trees is given in figure 1.1.

[1]It should be noted as well that GPSG attempted to get along with a very simple grammar formalism as a matter of principle.

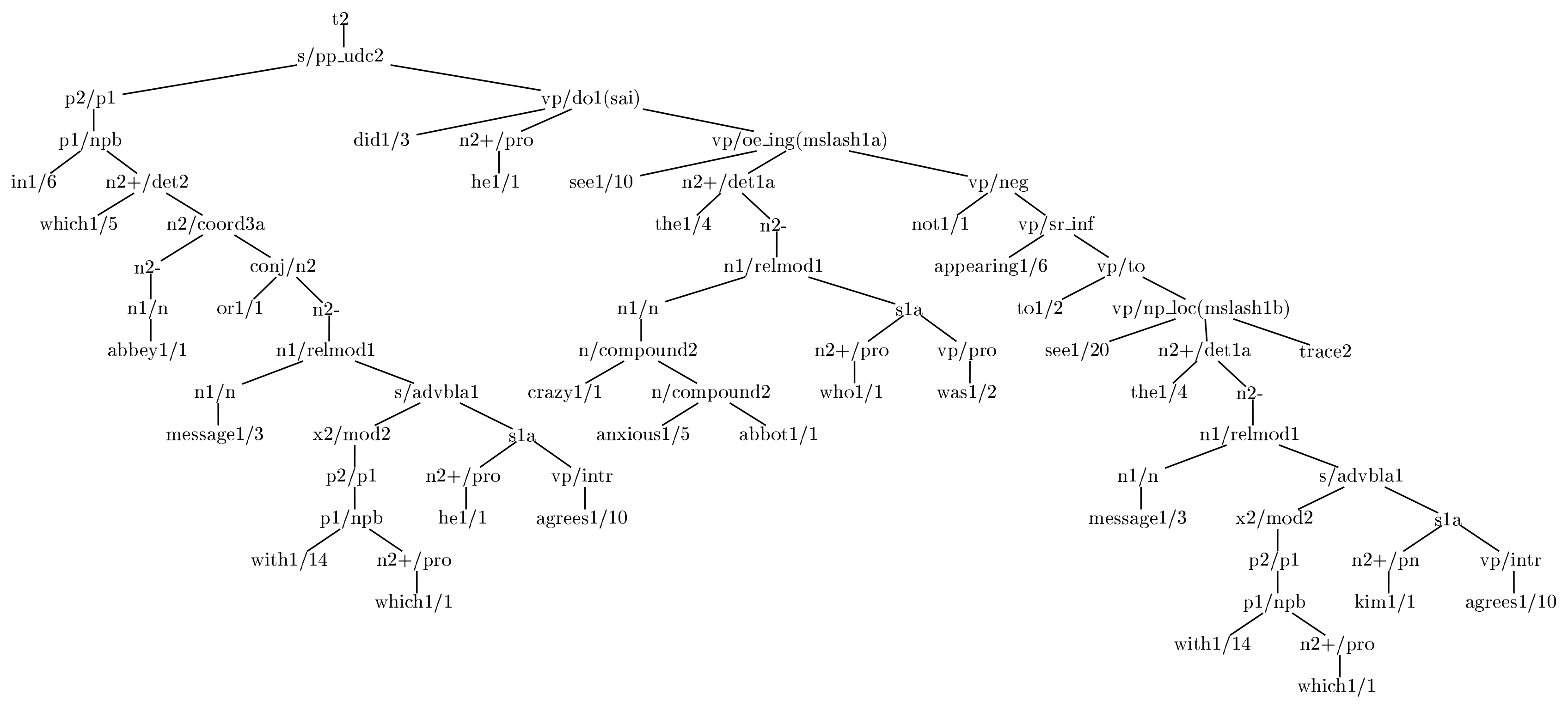


Figure 1.1: One of the 2,736 parse trees assigned by the Alvey Grammar to the sentence *In which abbey or message with which he agrees did he see the crazy anxious abbot who was not appearing to see the message with which Kim agrees?* The contrived example is semantically peculiar because publicly available version of the Alvey Grammar only contains a small lexicon (due to copyright restrictions); the sentence occurs in the test suite distributed with the grammar. The parse was obtained using the Groningen Hdrug package (van Noord and Bouma, 1997). It is important to note that the massive ambiguity is not due to the peculiarity of the example: the average ambiguity for the test suite was over 100 parses per sentence. The Alvey grammar and a web-based demo are available at http://www.let.rug.nl/˜vannoord/CL97/.

1.2 Grammar Approximation and Grammar Specialisation

How shall we deal with inefficiency and massive ambiguity? We propose to investigate solutions to these problems along the following two dimensions.

To prevent inefficiency we shall explore *automated grammar approximation.* Techniques will be investigated in which natural language grammars expressed in powerful constraint-based grammatical formalisms are approximated by devices of a much more simple form (typically by *finite-state* devices). Progress in this area would explain the relation of linguistic competence and linguistic performance; at the same time it would facilitate the practical application of more advanced natural language processing techniques in practical applications. Contributions are foreseen not only from theoretical and computational linguistics, but also from mathematical linguistics and psycholinguistics.

To account for the certainty of human communication, we shall investigate *automated* grammar specialisation, a means of optimising a theoretical grammar on the basis of a corpus of representative linguistic behavior. Applications of specialisation will yield specialised, less ambiguous grammars but in a manner which does not rely on the grammarians' intuitions. Such specialised grammars may be expected to take the form of *hybrid* systems which combine rules with statistical information. Contributions are foreseen from theoretical linguistics, computational linguistics, information theory and machine learning. Progress in this area will contribute to our theoretical understanding of disambiguation and will facilitate the portability of natural language processing components.

1.3 Innovation

The most important innovation of the proposed project is the combination of linguistically sound grammars on the one hand with corpus-based techniques on the other hand. The area of computational linguistics has seen a shift of perspective over the last ten years. After a period in which 'knowledge-based' approaches towards computational linguistics dominated the field, in combination with applications which only had a long-term potential (such as fully automatic, high quality translation), the last ten years corpus-based and probabilistic techniques have become quite popular, together with more emphasis on less ambitious applications with short-term potential.

We feel that this shift of attention was beneficial because of its emphasis on *evaluation* and 'real-world' problems. We also feel, however, that this 'no-nonsense' attitude has neglected some of the potential that linguistics has to offer. We believe that in order to be able to extend the state of the art to larger domains of applicability it is necessary to import linguistic insights again. And furthermore, such a connection with theoretical linguistics is warranted from a theoretical point of view.

The innovative aspect of the current proposal therefore is that it combines a *corpus-based* evaluation methodology with a sound linguistic basis.

Another innovative aspect of the proposal is that it focuses on the Dutch language, and hence on Dutch linguistics, and Dutch language technology. This is an important aspect of the proposal, since we believe it is important that language technology is developed for Dutch in addition to the current developments for languages such as English, German and French. Language technology applications such as grammar checkers, dictation systems, documentation systems and aids to the handicapped are cultural bonuses that should accrue not only to the speakers of majority languages.

A recent overview on Dutch Language and Speech Technology (Bouma and Schuurman, 1998) reports that there are many fewer language technology resources available for Dutch (as compared to English). In particular, the report signals a need for implemented Dutch grammars. The present proposal accords with one of the report's recommendations: fundamental and applied research in language and speech technology in Dutch should be stimulated.

1.4 Overview

The research project aims to answer the question of how it is that knowledge of language is applied in communication. Two major concerns can be identified. Firstly, it is important to understand how it is possible that knowledge of language is applied so *efficiently* by humans when they understand and produce natural language. We propose to investigate the hypothesis that natural language processing is *finite-state* in nature. This hypothesis is explained in more detail in section 2.

Secondly, humans are also very good at *disambiguation*; natural language users quickly discard ridiculous readings of a given sentence by taking into account the context and situation of the utterance. We propose to investigate techniques which are capable of augmenting the knowledge of language (modelled in the form of a grammar) with a theory of how this knowledge of language is usually applied. Below, in section 3, we refer to such techniques as *grammar specialisation* in order to stress the fact that we take a linguistic grammar (the model of the knowledge of language) as an important point of departure for such techniques.

Although the proposed research aims at general answers to the above-mentioned questions we believe strongly in a methodology in which concrete proposals are developed and compared. For this reason we want to be able to apply and evaluate such concrete proposals on a specific grammar of Dutch. This aspect of the proposal is defined in section 4.

Moreover, we propose to *apply* some of the approximation and disambiguation techniques in **lgrep**: a linguistic search tool for bare text-corpora (section 5). Such a tool would be useful for (computational) linguists working with corpora, but also as an extension to traditional grammars as used by language learners, to be able to obtain example sentences of particular constructions upon request. The tool will also be useful for the proposed research itself; for each of the proposed research areas corpus exploration is important and therefore such a corpus exploration tool will be a useful tool.

Chapter 2

Finite-state Language Processing

2.1 Arguments for Finite-state Language Processing

A major research question concerns the possibility of *approximating* an underlying general and abstract grammar by techniques of a much simpler sort. The idea that a competence grammar might be approximated by finite-state means goes back to early work by Chomsky. For instance, in Chomsky (1964) it is proposed that, among other things, the theory of grammar should make available a function g which returns for a given grammar G_i and a given amount of memory n, the description of a finite automaton that takes sentences as input and gives structural descriptions as output. Chomsky then continues (page 121):

> [this] would take us one step closer to a theory of the actual use of language. We can attempt to construct g in such a way that $g(i, n)$ will be a reasonable model for the production (or recognition) of sentences by the speaker (or hearer) who has internalised the grammar G_i and who has a memory capacity determined by the value of n. Notice that although the grammar G_i mastered by the user of a language is of course finite, it is not to be expected (and, in the case of natural languages, it is not in fact true) that a finite automaton can be constructed which will be able to accept (or generate) all and only the sentences generated by G_i, or which will be able to "understand" just these sentences [. . .]. This is no stranger than the fact that someone who has learned the rules of multiplication perfectly (perhaps without being able to state them) may be unable to calculate 3,872 $\times$ 18,694 in his head, although the rules that he has mastered uniquely determine the answer.

There are essentially three observations which motivate the view that the processing of natural language is finite-state:

1. humans have a finite (small, limited, fixed) amount of memory available for language processing

2. humans have problems with certain grammatical constructions, such as center-embedding, which are impossible to describe by finite-state means
3. humans process natural language very efficiently (in linear time)

The first argument is expressed in Chomsky (1963) on page 390 as follows:

> [...] we must conclude that the *competence* of the native speaker cannot be characterised by a finite automaton [...]. Nevertheless, the *performance* of the speaker or hearer must be representable by a finite automaton of some sort. The speaker/hearer has only a finite memory, a part of which he uses to store the rules of grammar (a set of rules for a device with unbounded memory), and a part of which he uses for computation in actually producing a sentence or "perceiving" its structure and understanding it.
>
> These considerations are sufficient to show the importance of gaining a better understanding of the source and extent of the excess generative power of context-free grammars over finite automata (even though context-free grammars are demonstrably not fully adequate for the grammatical description of natural languages).

Pulman (1986) comments as follows (page 198):

> Expressed in this way, the claim that the parsing abilities of speaker-hearers can be modelled in finite-state terms is fairly trivial: the performance of any cognitive ability by any mortal organism with finite powers is likewise guaranteed to be finite state in character.

We could add that not just mortal organisms face this restriction, but any physical object, including modern computers. Pulman then introduces *strict* finite state devices which are finite state devices with a "fairly small" amount of fixed memory, and suggests that Chomsky (and others) had this in mind, when stating the finite-state requirement above.

The second argument, concerning center-embedding, is perhaps the best-known argument for the hypothesis that natural language processing is finite-state in nature. Grammatical constructions involving center-embedding are very hard to process by humans (Miller and Chomsky, 1963; Chomsky, 1965). For example, the following sentences are grammatical, but hard to understand:

(1) a. %I called the man who put the book that you told me about down up
b. %The man who the boy who the students recognised pointed out is a friend of mine
c. %The man the boy the students recognised pointed out is a friend of mine
d. %The rat the cat the dog chased bit ate the cheese

(2) %De de de oude schuur bewonende boer toebehorende kat haat ratten
The the the old shed living farmer owned cat hates rats
The cat owned by the farmer living in the old shed hates rats

If the embedding occurs at the edge of a word group, the examples are easy to understand, indicating that the observed difficulty in understanding is not of a semantic nature:

(3) a. I called the man up who put the book down that you told me about
b. [This friend of mine is [the man who was pointed out by [the boy who was recognised by the students]]]
c. [This friend of mine is [the man pointed out by [the boy recognised by the students]]]
d. This is [the dog that chased [the cat that bit [the rat that ate the cheese]]]

(4) De kat die toebehoort aan de boer die de oude schuur bewoont haat ratten
The cat that is-owned by the farmer who the old shed lives-in hates rats
The cat owned by the farmer living in the old shed hates rats

Such observations find natural description in a finite-state approach, since finite-state devices are incapable of describing such center-embedded constructions (of arbitrary depth).

Of particular interest for a Dutch audience is the observation that the famous Dutch cross-serial dependency construction becomes extremely hard to understand if the number of argument selecting verbs which take part in the construction is larger than 3. This construction has been the basis of the most convincing argument to date that natural languages cannot be described by context-free means (Huybrechts, 1984; Shieber, 1985).

(5) a. dat de inspectie de bewaker de gevangenen zag helpen ontsnappen
that the inspection the guard the prisoners saw help escape
that the inspection saw that the guard help the prisoners to escape
b.?%dat de directie de inspectie de bewaker de gevangenen liet zien helpen ontsnappen
that the management the inspection the guard the prisoners let see help escape
that the management let the inspection see the guard help the prisoners to escape
c. %dat het hele land de directie de inspectie de bewaker de gevangenen hoorde laten zien helpen ontsnappen
that the whole country the management the inspection the guard the prisoners heard let see help escape
that the whole country heard the management let the inspection see the guard help the prisoners to escape

Similar sentences in which the recursion occurs at the edge of the word-group are much easier to understand:

(6) a.

dat	de	inspectie	zag	dat	de	bewaker	de	gevangenen	hielp	ontsnappen
that	the	inspection	saw	that	the	guard	the	prisoners	helped	escape

that the inspection saw the guard help the prisoners to escape

b.

dat	de	directie	de	inspectie	liet	zien	dat	de	bewaker	de	gevangenen
that	the	management	the	inspection	let	see	that	the	guard	the	prisoners

hielp	ontsnappen
helped	escape

that the management let the inspection see the guard help the prisoners to escape

c.

dat	het	hele	land	hoorde	dat	de	directie	de	inspectie	liet	zien	dat
that	the	whole	country	heard	that	the	management	the	inspection	let	see	that

de	bewaker	de	gevangenen	hielp	ontsnappen
the	guard	the	prisoners	helped	escape

that the whole country heard the management let the inspection see the guard help the prisoners to escape

Again, such a limit might easily be explained in a finite-state approach, because finite-state devices are uncapable of describing such crossing dependencies.

Finally, let us consider the third argument, concerning the observed efficiency of language processing for humans. For a given grammar or grammatical formalism, computational efficiency is expressed by stating the number of steps that is required to compute the analysis of a sentence of length n. In the simple case, the number of steps will increase proportionally with n. In such cases the complexity is said to be linear. For more complex grammatical formalisms, the number of steps will increase faster than n, e.g. it increases proportionally with n^3 for context-free grammars. This implies that such more powerful grammatical formalisms are unlikely models of human language, since humans display no noticeable delay in understanding longer sentences.

2.2 Previous Work in Finite-state Language Processing

Finite-state techniques have been used extensively in language processing. For instance, it has been shown that phonological models consisting of sets of context-sensitive rewrite-rules are, given some reasonable assumptions, finite-state (Johnson, 1972; Koskenniemi, 1983; Kaplan and Kay, 1994). Recently, a similar result has been obtained for more modern phonological models expressed in *Optimality Theory* (Karttunen, 1998).

Syntactic research has almost always followed Chomsky's arguments against finite-state grammars as a competence model, but there are some exceptions (Krauwer and des Tombe, 1981; Gross, 1997).

In linguistic *applications* the use of finite-state techniques is very wide-spread; the following overview only lists a small number of recent publications:

tokenization (Grefenstette and Tapanainen, 1994; Silberztein, 1997) Tokenization is concerned with relatively superficial problems such as the determination of sentence boundaries and word boundaries in a given text (which, however, are challenging in languages with writing systems with poor marking of boundaries, e.g., Japanese). The techniques are comparable to the lexical analysis phase of the compilation of programming languages (such as provided by the UNIX tool `lex`): regular expressions are defined for words, sentences, and possibly some further special categories such as dates, numbers, etc.

lexicography (Mohri, 1996; Daciuk, 1998) Recent interest has focused around compact implementations of dictionaries using finite-state minimisation techniques; such techniques yield smaller results than the traditional *trie* data-structure. Other applications of finite-state techniques in lexicography concerns the representation of subcategorisation properties of words by means of finite-state patterns (Gross, 1989).

spell checking Spell checking is naturally described as an application of finite-state automata (Oflazer, 1996; Daciuk, 1998).

part-of-speech tagging Part-of-speech tagging can be implemented by means of finite-state transducers. Roche and Schabes (1995) describe a method to compile a rule set produced by the application of the rule-based tagger by Eric Brill (Brill, 1992; Brill, 1995) into a deterministic finite-state transducer. As a result, an extremely fast tagger is obtained.

speech recognition Language modelling for speech recognition is almost always performed with finite-state models, in particular hidden Markov models (Jelinek, 1998). Further applications of finite-state techniques for speech recognition are described in Pereira and Riley (1996); Mohri et al. (1998).

phonology and morphology The phonology and morphology of natural languages is often implemented by means of finite-state devices; for instance the popular 'two-level' system of Koskenniemi (1983) is finite-state. As we already mentioned above, Kaplan and Kay (1994) show that the context-sensitive rule systems traditionally used in phonology are finite-state in expressive power as well. Further improvements concerning the compilation of such sets of context-sensitive rules into finite-state automata are described in Karttunen (1995) and Mohri and Sproat (1996).

syntax A number of proposals exist for the treatment of natural language syntax by means of finite-state devices. The best-known example probably is the Constraint Grammar approach (Voutilainen and Tapanainen, 1993; Karlsson et al., 1995; Voutilainen, 1997; Tapanainen, 1997). Other approaches can be found for instance in Joshi and Srinivas (1994); Abney (1995); Chanod and Tapanainen (1996); Grefenstette (1996); Roche (1997). Finite-state approximation techniques are described below.

If we restrict our attention to finite-state approaches to natural language syntax in which a distinction is maintained between a *competence* grammar (expressed in a formalism with at least context-free power) and language *performance*, then two different approaches can be identified as methods to realize the link between competence and performance.

Firstly, some have proposed parsing algorithms which interpret such a grammar. The parser, however, is associated with a fixed maximal amount of memory. Publications such as Johnson-Laird (1983); Pulman (1986); Resnik (1992) are examples of this *interpreter* approach. Resnik (1992) shows how a *left-corner parser* (Rosenkrantz and Lewis-II, 1970), augmented with an operation of "eager" composition, and formalised as a push-down automaton, leads to a parser which utilises its stack only for center-embedding constructions. Left-recursive and right-recursive structures, on the other hand, can be processed without exploiting the stack. In contrast, a purely bottom-up or top-down parser formalised in a similar fashion requires to use its stack for right-recursive (respectively left-recursive) structures.

In contrast, in finite-state *approximation* a finite-state grammar is explicitly constructed, often on the basis of a specific parsing algorithm restricted with some mechanism to limit its recognising power to finite-state languages. This approach, exemplified in Pereira and Wright (1991); Rood (1996); Pereira and Wright (1997); Nederhof (1997) and Johnson (1998) could then be called a *compilation* approach.

Chomsky (1964), footnote 10, discussing finite-state devices, quite rightly warns us that

> The assumption that sentences are produced or recognised by a device of this sort tells us almost nothing about the method of processing.

In this context, it is worth mentioning the result by Stearns (1967) that it is possible to determine whether a *deterministic* push-down machine recognises a regular language. Furthermore Stearns shows that if the push-down machine recognises a regular language, then the number of states of a finite-state automaton recognising the same language may be bounded by an expression of the order $t^{q^{q^q}}$, where q is the number of states, and t the size of the alphabet of the push-down automaton. Stearns concludes that[1]

> If this bound cannot be improved significantly, then it would appear profitable in some cases to maintain a pushdown design for a recogniser even if a finite-state design is possible.

In light of these observations, it may be difficult to choose between the *interpreter* or *compiler* approach, or still other approaches that may be designed (for instance finite-state automata augmented with a finite number of registers (Krauwer and des Tombe, 1981; Beesley, 1998)). Minimised finite-state automata are an appropriate normal form for regular languages; but they are only 'minimal' in the sense that there isn't a smaller automaton *in that representation.* For example, suppose you have a very complicated language X. Furthermore, you want to have an automaton for the language union(aXa, bXb). The minimal

[1]refer to Valiant (1975) for later results. Also note that the result applies to *deterministic* push-down automata. For general push-down automata no such upper-bound exists (Ullian, 1967).

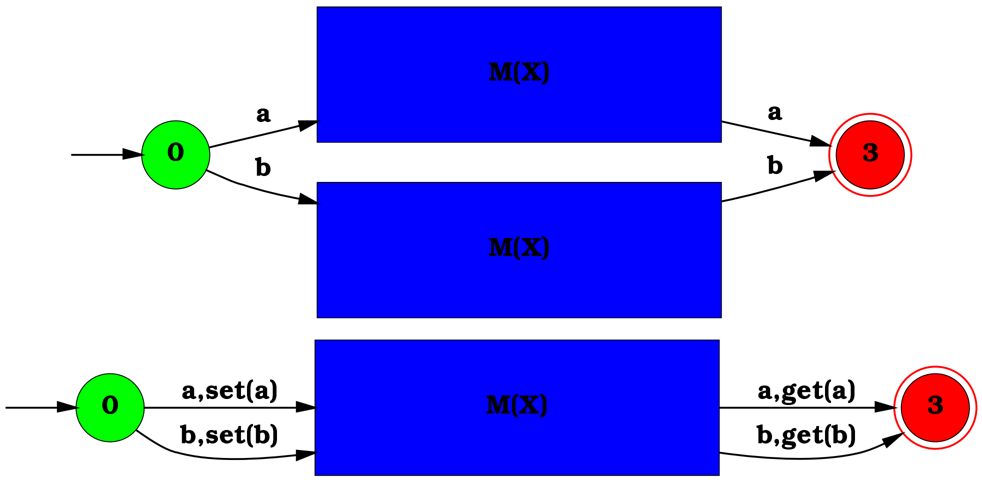


Figure 2.1: Two representations of the language union(aXa, bXb) where X is some given language and $M(X)$ is its corresponding automaton. In the classical representation, two copies of $M(X)$ are required. If a register is used to keep track of the first input symbol, then a single copy of $M(X)$ suffices. Here we use the operation set(x) to set the register, and get(x) to check the value of the register.

automaton for that language will essentially need to have a copy of the automaton for X in it, as in the first automaton of figure 2.1. However, if you had a single global memory cell containing the value of the first letter you read, then you would only need one copy of X, as in the second automaton of figure 2.1. So even if the two representations are formally equivalent, their sizes may vary dramatically.

As a starting point, the minimal deterministic finite-state automata constructed in the *compiler* approach may provide a *normal-form*; a representation that we can use to compare different approaches.

2.3 Research Questions in Finite-state Language Processing

2.3.1 Finite-State Calculus

In order to experiment with finite-state techniques, it is very important to have available an implementation of the Finite-state Calculus, i.e., implementations of all the important finite-state operations such as union, concatenation, Kleene closure, difference, intersection, complementation, etc. An operation such as union takes two finite-state languages l_1 and l_2 (expressed as regular expressions or as finite-state automata) and produces the union of the sentences in these languages (this is defined as $l_1 \cup l_2 = \{s | s \in l_1 \vee s \in l_2\}$). Such regular operations are crucial building blocks of approximation algorithms and other finite-state language processing algorithms.

Furthermore, such an implementation should make available various tools for performing operations such as determinisation and minimisation of automata. A number of implementations exist (Karttunen, 1995; Kiraz, 1997; Evans, Kiraz, and Pulman, 1996; Mohri and Sproat, 1996). The sources of these implementations are not available however. Since we expect that fruitful research in this area will lead to extensions and/or alternative implementations it is important that an implementation of the finite-state calculus is available that can be easily extended.

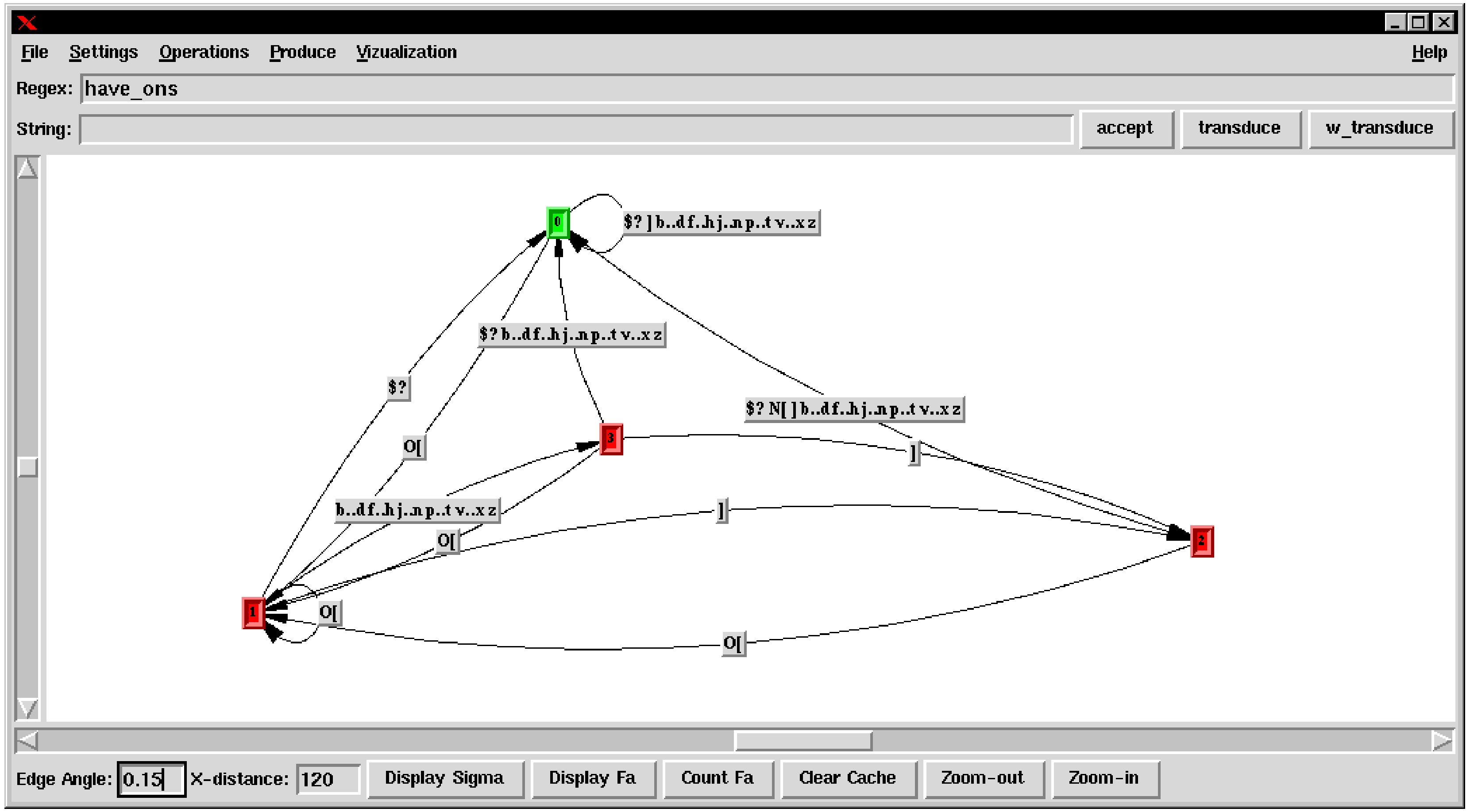
Therefore it is proposed that the FSA Utilities toolkit (van Noord, 1997; van Noord, 1998) is adopted, and actively maintained and developed as part of this project. The toolkit consists of a collection of utilities to manipulate regular expressions, finite-state automata and finite-state transducers. Manipulations include automata construction from regular expressions, determinisation (both for finite-state acceptors and finite-state transducers), minimisation, composition, complementation, intersection, Kleene closure, etc. Furthermore, various visualisation tools are available to browse finite-state automata. An illustration of the tool-box is given in figure 2.2. Some of the functionality of the toolkit is available through the World Wide Web. A first demonstration version was developed by the University of Karslruhe (http://i12www.ira.uka.de/Visualisierung.endlicher.Automaten/). Peter Kleiweg has developed a demonstration version of some of the regular expression features of the toolkit (http://www.let.rug.nl/~vannoord/fsademo/).

2.3.2 Methods of Finite-state Approximation

Many of the existing finite-state approximation techniques are defined on the basis of a specific parsing algorithm restricted with some mechanism to limit its recognising power to finite-state languages. For instance, whereas the approach of e.g. Pereira and Wright (1997) is based on an LR parser, Johnson (1998) proposes a finite-state approximation technique based on a left-corner parser. The result can be seen as an explicit compilation of the parser in Resnik (1992) into a finite-state automaton, by providing a maximum depth to the stack size. An alternative approach is described in Evans (1997).

As far as we know, the finite-state approximation approaches mentioned before have not yet been applied in practice; on the other hand, the finite-state parsing approaches known as *chunking* have already been applied successfully. In these latter approaches exemplified in publications such as Abney (1995); Chanod and Tapanainen (1996); Grefenstette (1996) and Roche (1997), a grammar is defined by a sequence of levels (or *stages*). Each level consists of a number of 'grammar rules' which apply to a given string. The result of a level is then passed on to the next level. Lower levels are used to create phrases. These phrases are then combined into more extensive analyses by later levels. Since rules do not apply recursively at a level, and there is only a finite number of levels, the resulting *cascade* is finite-state.

The drawback, from our perspective, of the finite-state chunking method is that such finite-state grammar rules are defined directly as such, and are not derived in some way from a more powerful grammatical model. We will therefore investigate whether it is possible to obtain finite-state approximations by automatically dividing the grammar rules of a given



Regex: have_ons

String:

Edge Angle: 0.15 X-distance: 120

Figure 2.2: Illustration of the FSA Utilities toolkit. In this example the regular expression operator 'have_ons' and its associated finite-state automaton is displayed. The 'have_ons' operator is part of an implementation in finite-state calculus (based on Karttunen (1998)) of the syllabification analysis in Optimality Theory (Prince and Smolensky, 1993). The 'have_ons' operator represents the constraint that syllables must have onsets.

grammar into such a finite-state cascade approximating the original grammar. In order to be able to do this, static grammar analysis techniques will be employed, in combination with certain limits on recursive rule application.

2.3.3 Qualitative Evaluation of Finite-state Approximation

Finite-state approximation techniques can be evaluated by comparing the language of the underlying grammar, and the language defined by the approximation. Some techniques always produce a *superset* of the language of the input grammar (such techniques are useful in applications where the finite-state approximation is used as a pre-processor); other techniques always yield a *subset* of the language of the input grammar (for instance in approaches which are psycholinguistically motivated). For some techniques neither characterisation holds.

Furthermore, techniques can be characterised by evaluating their results for certain sub-classes. For instance, an interesting result would be if a certain approximation technique would be *exact* i.e., always find an *equivalent* finite-state automaton, for cases in which the input describes a regular language. However, it turns out that this is impossible; it is an unsolvable problem in general to transform a given context-free grammar generating a regular language into an equivalent finite-state automaton (Ullian, 1967).

Other sub-cases are for instance the left-linear and right-linear grammars. The technique described in Pereira and Wright (1997) is exact in these cases. The case of left-linear and right-linear grammars is generalised somewhat in Nederhof (1997) to the strongly regular grammars. His method is exact for those grammars.

Another interesting question is to compare a given finite-state approximation with the psycholinguistic observations discussed in the previous chapter. For instance, what kind of center embedding is allowed in the approximation? How does this compare with human abilities? Similar questions can be asked with respect to other types of construction which are difficult for humans such as crossing dependencies. Obviously, constructions humans have no difficulty with should be treated without problems.

2.3.4 Experimental Evaluation of Finite-state Approximation

Apart from a qualitative evaluation of finite-state approximation techniques it is worth evaluating how such approximations work out in practice, i.e. for realistic grammars on realistic corpora. As far as we know such an empirical evaluation has not been performed before. Therefore, we propose to set up experiments for a number of different approximation techniques, in order to compare their effects.

Such an empirical, quantitative evaluation will start out by selecting an appropriate source grammar and an appropriate corpus. Hopefully some of the sub-corpora developed in the corpus initiative 'Corpus Gesproken Nederlands' will be useful for this purpose (cf. section 4.4.1). Implementations of a number of existing approximation techniques will be required. These techniques can then be applied to the grammar and the resulting finite-state automata can be compared. Such a comparison should not only take into account the loss of accuracy that results from the approximation, but will also take into account computational issues, such as the size of the resulting automaton, the difficulty of the production of the automaton, and other properties of the automaton.

2.3.5 Approximation of constraint-based grammars

Most finite-state approximation techniques assume that the input grammar is a context-free grammar. Preliminary experiments with feature-based grammars are described in Black (1989). Furthermore, certain approximation techniques generalise to feature-based grammars in a natural way (Johnson, 1998). In other cases the assumption is that a constraint-based grammar is first approximated by a context-free grammar, which can then be approximated by a finite-state grammar in turn.

Each of these methods have in common that distinct complex categories in the input grammar are mapped to the same category in the approximation. This leads to an approximation which will allow sentences not allowed by the input grammar (the *superset* case). One important research question in the proposed project is the question how approximations of constraint-based grammars can be defined which yield an interesting *subset* of the sentences allowed by the input grammar.

2.3.6 Finite-state Approximation and Interpretation

At first sight, the concept of finite-state approximation of e.g. context-free grammars seems to ignore an important aspect of such grammars. Context-free grammars not only characterise the grammatical sentences of a language, but also assign structural descriptions to such sentences. These structural descriptions are crucial for the interpretation of sentences.

Different approaches can be identified with respect to this issue. It has been suggested (Pereira and Wright, 1997) that the original grammar should still be used for the purpose of interpretation. In such a set-up the finite-state approximation is used as a quick filter to rule out (possibly many) impossible analyses. For a few succeeding analyses the original grammar is then applied to obtain interpretations.

In Nederhof (1998) this approach is extended by letting the finite-state approximation produce a finite-state *transducer* rather than a recogniser. The transducer produces a table, in linear time, which can then be used (in a second phase) to recover all parse-trees with respect to the original grammar.

The drawbacks of these methods is that the original grammar is still crucial for language understanding. We envision a more elegant approach in which a finite-state transducer is constructed as a finite-state approximation; this transducer should directly produce structures which can be used for language interpretation. There should be no need to refer to the input grammar in the semantic interpretation component. Some interesting ideas are presented in Krauwer and des Tombe (1981).

2.3.7 Finite-state Approximation in Language Generation

Finite-state approximation is a technique which has been exclusively applied to language understanding. An interesting question is whether it is fruitful to regard language generation as a finite-state process as well. If it is a fundamental property of human language that its processing is limited by finite-state means, then a natural question to ask is whether language production can be characterised by limitations of a similar nature.

2.3.8 Compact Formats for Finite-state Approximations

Deterministic finite-state automata (DFA) constitute only one formalisation of regular languages. The advantages of this particular formalisation are that it can be implemented extremely efficiently (in linear time, and independent of the size of the actual automaton), and that for each given deterministic finite-state automaton an equivalent but *minimal* automaton can be automatically computed. The drawback of deterministic finite-state automata as a formalisation of regular languages is that such DFA are in general not very *compact*. In fact, this is the reason that in many implementations of large DFA a somewhat less efficient but much more compact format is chosen (Daciuk, 1998; Kowaltowski, Lucchesi, and Stolfi, 1993).

Many other formalisations of regular languages exist. For instance, *regular expressions* are another very popular device to present regular languages. Such regular expressions can be implemented fairly efficiently as well (for instance, the regular expression matching in Perl is done without compilation to DFA), but there is no concept of a 'minimal' regular expression.

In Beesley (1998) finite-state automata are augmented with a finite number of registers. This allows for a much more compact representation. Moreover, efficiency of implementation is hardly affected. However, the drawback is that there is little known about the construction of a minimal representative for such augmented finite-state devices.

In Kowaltowski, Lucchesi, and Stolfi (1993) binary automata are introduced as a representation for regular languages. Again, very compact and fairly efficient representations are possible. The authors discuss a few heuristics to minimize such binary automata. No general algorithms are known though.

An important question therefore is whether there exist a representation of finite automata which combines efficiency and compactness with the possibility to construct minimal representatives automatically.

Chapter 3

Grammar Specialisation

3.1 Motivation

Humans are very good at *disambiguation*; natural language users quickly discard ridiculous readings of a given sentence by taking into account the context and situation of the utterance. In fact, humans typically are unaware of the alternative readings of an utterance. Only in special circumstances (for instance in jokes) the ambiguity property of natural language becomes obvious. A computational theory of language use therefore will have to explain how it is that the appropriate reading in a given situation is selected from the space of possible readings provided by the grammar.

We propose to investigate techniques which are capable to augment the knowledge of language (modelled in the form of a grammar) with a theory of how this knowledge of language is applied in a given situation. We refer to such techniques as *grammar specialisation* in order to stress the fact that we take a linguistic grammar (the model of the knowledge of language) as an important point of departure for such techniques.

Problems of disambiguation are often avoided in natural language processing by focusing on a limited domain. In such a domain, utterances can often be understood unambiguously. The grammar used in such an approach is heavily tuned towards the domain. This tuning is performed by human experts. It is unclear what the relation is between this tuned version of the grammar and the underlying, more general grammar. Moreover, such a tuned grammar is only useful for the intended application and cannot be used without major investments for other domains: the grammar is not *extendible.*

The investigation will focus on possibilities to perform such grammar tuning automatically. Such *automated grammar specialisation* techniques would constitute a potential answer to the question how the desiderata of suitability for a specific domain (on the one hand) and ease of portability (on the other hand) can be combined. Such techniques incorporate some elements from the data-driven parsing approach, yet it differs from pure data-driven parsing in that a general and abstract linguistically inspired grammar is still essential as the starting point of development of the parser.

The expected practical gain from such techniques is that the grammar can be made more general, i.e. suited for several domains, without loss of parsing accuracy for a specific domain of application. This is because the grammar can be automatically specialised for such an application, omitting from consideration all language phenomena that do not occur in corpora for the corresponding domain.

It may be worthwhile to explain more carefully the role of such specialisation techniques. We assume that a grammar provides for a set of *linguistically possible* meanings for a given utterance. This is the set of meanings that could be assigned to a given utterance if only linguistic constraints are taken into account. In real life, almost every utterance receives only a single meaning: the situation or context typically is such that the other readings do not make sense. Only in the case of jokes or other language games is ambiguity intended.

The purpose of grammar specialisation techniques is to formalise at least certain aspects of the interaction between what is linguistically possible, and what is appropriate in a given situation.

It may be argued that, rather than specialising the grammar, some extra-grammatical reasoning mechanism should be employed. In such an argument, it is assumed that knowledge of the situation and context is available, and that a given candidate meaning is somehow first checked for consistency with the available information. Apart from the consideration that the sophisticated knowledge representation and automated reasoning techniques that would be required are not yet readily available, there are a number of theoretical reasons why we believe that it is worthwhile to consider specialisation techniques (perhaps as a supplement to such reasoning techniques).

Firstly, in many cases not enough information will be available to base the relevant inference on. It is typically extremely hard to spell out precisely why a certain reading should be avoided in a certain context or situation, because typically not enough information is available to base the decision on. Therefore, the specialisation techniques that we propose to investigate are are a means to aid reasoning in circumstances where not enough information is available.

Secondly, it seems that it is quite unlikely that extra-grammatical inference techniques are employed to solve all disambiguation tasks. Such an hypothesis seems to predict that all of the linguistically possible readings of a given sentence are to be constructed, before inference filters most of these; that would lead to the expectation that we should be somehow conscious of the other possible readings. Consider the example:

(1) We saw the man with a telescope

People tend to be surprised if the alternative reading where *saw* is understood as the present tense of the verb *to saw* is explained to them, suggesting that these readings were never actually constructed. Moreover, given the fact that there can be exponentially many different readings for a given utterance, such a setup would also lead to efficiency problems (both from a practical and a theoretical point of view).

In the grammar specialisation approach that we propose here, the integration of disambiguation techniques into the grammar, in combination with standard best-first search techniques, will in general avoid the computation of all linguistically possible meanings, but instead return only the most plausible one.

Obviously, humans quickly adapt their language use to varying situations and domains. In an appropriate context our examples of ambiguous sentences might be understood differently. It is thus clear that disambiguation techniques should be able to model these *dynamic* aspects of natural language. In this proposal, however, we will not immediately concentrate on these dynamic aspects but we will start with the assumption that useful insights can already be gained with a simpler static approach. Given this limitation we will focus mostly on examples which can be explained under fairly stable domain characteristics; we will focus less on dynamic aspects such as pronoun resolution beyond sentence boundaries.

3.2 Previous Work in Grammar Specialisation and Disambiguation

A number of preliminary attempts have been carried out in this area. In the last few years, probabilistic disambiguation techniques have become increasingly popular. As a beneficial side-effect of the application of such probabilistic techniques, the *corpus-based evaluation* of parsing systems has become much more important. In the present proposal we want to combine such a corpus-based methodology with a solid theoretical linguistic basis.

In Grishman et al. (1984) a full coverage grammar is applied to a given corpus of utterances. After removing incorrect analyses this analysed corpus is then inspected to see which grammar rules are actually used. A specialised grammar is constructed which contains the subset of the rules used in the analysis of the corpus.

An obvious extension of this idea consists in adding probabilities to rules, depending on how often a particular rule is used in the analysis of the corpus. This is different from the inside-outside algorithm for stochastic context-free grammars (Baker, 1979) which is used to estimate probabilities from an unannotated corpus; but which has not been very successful in practice.[1] In Pereira and Schabes (1992) a more promising technique is described which adapts the inside-outside algorithm to *partially* bracketed corpora.

In Briscoe and Carroll (1993) a technique is described in which the actions in an LR parsing table are augmented with probabilities. As a further step, Rayner and Carter (1996) introduce a pruning technique which filters out very unlikely actions during the parsing process. Again, a corpus of analysed examples is used to determine what actions count as unlikely.

A promising alternative means of specialisation consists of the application of explanation-based generalization techniques to natural language parsing (Rayner, 1988; Rayner and Samuelsson, 1990; Samuelsson, 1994; Srinivas and Joshi, 1995). A full coverage grammar is used to analyse a given set of examples. From the analysed corpus a

[1]For instance, the technique is not even mentioned in Jelinek (1998).

specialised grammar is constructed which typically only analyses a subset of the language analysed by the general grammar. However, it does so very efficiently. Moreover, and more importantly, it often favours more likely analyses. If all goes well, the specialisation removes useless analyses, while retaining the appropriate ones.

Another approach towards disambiguation is the data-oriented approach described in Scha (1990) and Bod (1995). Preliminary experiments on the ATIS corpus of the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993) were very promising. In this model, a stochastic tree substitution grammar is created by taking into account (almost) all possible sub-trees of the trees present in an annotated corpus.

A very successful approach consists of the application of decision tree algorithms to parsing. For instance, very good parsing performance on the Penn Treebank Wall Street Journal corpus (Marcus, Santorini, and Marcinkiewicz, 1993; Marcuss et al., 1994) has been reported for these techniques (Jelinek et al., 1994; Magerman, 1995). These methods make heavy use of lexical information. In Black et al. (1997) a different system based on statistical decision tree modelling is described which is also capable of capturing linguistic dependencies. Remarkable results are presented on the ATR/Lancaster Treebank of General English (Black et al., 1996). The interesting aspect of this work is the central role played by a detailed and linguistically motivated grammar of English.

Very good results on the Penn Treebank Wall Street Journal corpus have also been reported in Collins (1996); Collins (1997). A number of lexicalised probabilistic models are compared. These models are sensitive to the *lexical head* of constituents. Moreover, probabilities over subcategorisation frames are incorporated; complement/adjunct distinctions are important, and *WH*-movement constructions are treated separately. Somewhat similar techniques are described in Eisner (1996) and Eisner (1997), expressed in terms of Dependency Grammar, where it is very natural to express lexical dependencies of a statistical nature.

Disambiguation of prepositional phrase attachment is the subject of a number of other experiments in which phrases with prepositional phrase attachments were extracted from the Penn Treebank Wall Street Journal corpus consisting of the sequence *verb noun-phrase prepositional-phrase*. Of these, only the verb, head noun of the first noun phrase, preposition and head noun of the noun phrase contained in the prepositional phrase were recorded. Thus, in these experiments the lexical heads are deemed important. Experiments with a variety of techniques have been reported, including a Decision Tree model and a maximum entropy model (Ratnaparkhi, Reynar, and Roukos, 1994); a transformation-based learning model (Brill and Resnik, 1994); a relatively simple back-off model (Collins and Brooks, 1995); and a number of models using memory-based learning techniques (Zavrel and Daelemans, 1997). The adequacy of these models is roughly between 80 and 84%. Collins and Brooks (1995) moreover performed an experiment suggesting that humans, if given the same pieces of evidence, get about 88% correct; for full sentences accuracy of 93% is obtained. These facts suggests, again, that the (head) words are of extreme importance.

Finally, Lee and Choi (1998) report results indicating that a language model which takes into account lexical dependencies between head words improves upon N-gram language models, for the purpose of determining the most probable continuation of an utterance.

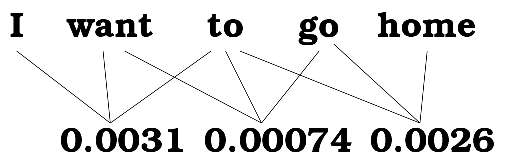


Figure 3.1: Trigrams capture dependencies between neighbouring words. Trigrams are unable to capture dependencies that are more than 2 words apart, such as the dependency between the subject *I* and the embedded verb *to go*.

3.3 Research Questions in Grammar Specialisation and Disambiguation

The central question is to what extent it is possible to adapt a linguistically motivated and general grammar of Dutch to a particular domain. Important evaluation criteria are how adequate the resulting, specialised, grammar is. Another important evaluation criterion is how much effort is required for the specialisation. For instance, many of the proposed specialisation techniques assume the existence of an annotated corpus of examples. In such cases, the evaluation should consider the required detail of annotation and the required amount of corpus material.

Apart from the importance of a corpus-based methodology, another important conclusion to be drawn from previous work is the importance of the actual words. For this reason, we propose to apply disambiguation techniques on lexical dependency structures rather than on syntactic parse trees. The focus on the actual words is now motivated as follows.

3.3.1 The Importance of Words for Disambiguation

In speech-recognition systems, a *language model* is responsible for the prediction of the 'next' word in an utterance. N-gram statistical models are almost exclusively used for this task. In such a model, the probability of the next word w is dependent only on the last $N-1$ words just seen. As a typical example, consider the case in which $N=3$. For such a *trigram* model, a large corpus is used to count the frequency of occurrence of all possible triples of words. If w_i, w_j were the last two words seen so far, then the probability that the next word is w_k is estimated to be the frequency of the trigram $\langle w_i, w_j, w_k \rangle$ divided by the frequency of the pair $\langle w_i, w_j \rangle$ (for low frequency counts often special arrangements are necessary). In the simple sentence:

(2) I want to go home

the probability that the word *home* follows after *I want to go* is thus estimated by the number of times *to go home* occurs in the corpus, divided by the number of times *to go* occurs in the corpus.

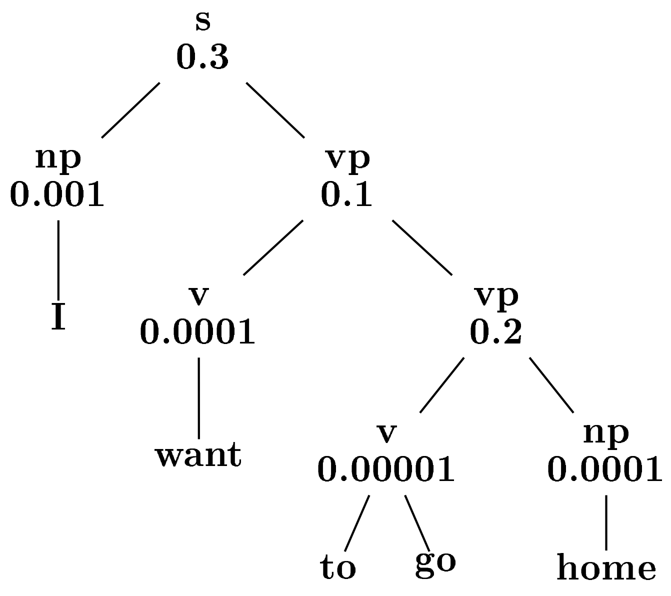


Figure 3.2: Parse-tree augmented with probabilities provided by a stochastic context-free grammars. Such models are unable to express the dependency between words such as *go* and *home*.

In practice, trigrams are much more accurate for the purpose of predicting the probability of a sentence than, for instance, stochastic context-free grammars which aim to model syntactic regularities (an observation which usually surprises syntacticians). Yet, it is immediately clear that many linguistically significant dependencies cannot be captured by such simple models. The success of simple models such as the trigram model strongly suggests that the actual words are extremely important.

The disappointing results of stochastic context-free grammars can be explained, because such stochastic context-free grammars are typically unable to express (statistical) dependencies between words. In stochastic context-free grammars, grammar rules are augmented with probabilities. Such probabilities can be automatically derived from a corpus. Simplifying matters somewhat, the probability of a rule such as $vp \rightarrow v, np$ is estimated by the number of times the rule $vp \rightarrow v, np$ occurs in the corpus divided by the number of times the category *vp* occurs in the corpus (i.e. the proportion of cases that this particular *vp* rule was used to derive a *vp*). An important limitation of such usage of stochastic context-free grammars is their inability to express (probabilistic) dependencies between words. In the example, there is no way the system can express the fact that *to go home* is a frequent expression in English. It is not surprising, therefore, that other techniques have been investigated which are capable to incorporate such lexical dependencies. Some of the most successful of these techniques have been discussed already in the previous section.

Note, however, that the lack of expressiveness with respect to lexical dependencies is not an inherent property of stochastic context-free grammars, but rather a property of their typical use. For instance, it is quite possible to envision stochastic context-free grammars

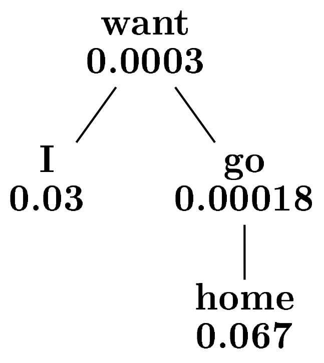


Figure 3.3: Dependency structure augmented with probabilities. In such dependency structures, probabilistic dependencies between dependent words can easily be expressed.

of lexical dependency structures in which each of the non-terminal nodes in the stochastic context-free grammar relates to a word in the input sentence. Obviously, the relation between input sentences and tree structure is different in such an approach. In such a set-up, some other grammatical device might produce such a dependency structure for which we can then compute its probability according to the stochastic context-free grammar (which defines all grammatical dependency structures and their associated probability). The example might for instance give rise to the dependency structure given in figure 3.3.

3.3.2 Disambiguation of Lexical Dependency Structures

A very promising line of research therefore consists of applying statistical techniques (such as stochastic context-free grammar) to lexical dependency structures, as opposed to traditional syntactic structures. Therefore, we propose to compare a number of probabilistic techniques which are sensitive to lexical dependencies.

In order to be able to do so, lexical dependency structures need to be defined. For instance, questions such as whether there should be a link in examples such as in figure 3.3 between the verb *go* and the matrix subject *I* should be answered. The construction of such lexical dependency structures is either performed explicitly by the grammar, or else can be straightforwardly derived from the structures the grammar derives. All modern grammatical theories exploit the notion *linguistic head* in one way or the other. Lexical dependency structures can be derived from such headed representations.

The following lists a number of approaches towards disambiguation:

- variants of stochastic CFG for lexical dependency structure
- maximum entropy models of lexical dependencies
- decision tree models of lexical dependencies
- inductive logic programming techniques for lexical dependencies

- data-oriented parsing of lexical dependency structures
- memory-based learning

For each of these approaches, an initial feasibility study will be conducted, in order to find out how such an approach can be combined with a given general grammar of Dutch. Based on this feasibility study a selection will be made from this list of a number of disambiguation techniques. The selected techniques will then be specified and implemented in detail, and carefully evaluated on an annotated corpus.

3.3.3 Evaluation of Disambiguation Techniques

An important aspect of the study of disambiguation techniques concerns their success on a human-annotated test-corpus. At the moment there is relatively little syntactically and/or semantically annotated corpus material available for Dutch. Hopefully, the recently initiated corpus initiative 'Corpus Gesproken Nederlands' (Levelt, 1998) will help to remedy this situation (cf. section 4.4.1). Furthermore, the test bank developed as an evaluation tool for the grammar development work (cf. section 4) can be used to perform a more qualitatively oriented evaluation.

Chapter 4

Grammar Development for Dutch

The proposed research aims at general answers to the above-mentioned questions of efficiency and disambiguation. In order to achieve such answers, we believe strongly in a methodology in which concrete proposals are developed and compared. For this reason we want to be able to apply and evaluate such concrete proposals on a specific grammar of Dutch.

Such a grammar of Dutch should describe a large fragment of the Dutch language, in such a way that it is able to treat the large majority of sentences of a given corpus of Dutch. The grammar should include a detailed and linguistically sophisticated treatment of constructions such as cross-serial dependencies (verb-clusters) and various types of nested dependencies (which potentially give rise to center-embedding, e.g. noun-phrases within adjectival phrases within noun-phrases), because these constructions are crucial for the qualitative evaluation of finite-state language processing techniques. The grammar should also provide a treatment of *government* and *headed projections*, in order to be able to use the grammar for disambiguation experiments for techniques which are based on lexical dependency structures. For the same reason, the grammar should treat various kinds of modification constructions including prepositional phrase attachments. Moreover, a large lexicon should be available in order to be able to experiment realistically with such disambiguation techniques.

In this section we describe a number of tasks aimed at the development of such a linguistically motivated grammar for Dutch.

4.1 Towards a linguistically motivated grammar for Dutch

Computational grammars for English, which are based, more or less directly, on linguistic theories are the Alvey (Briscoe et al., 1987; Grover, Carroll, and Briscoe, 1993) and CLE (Alshawi, 1992) grammars (both based on GPSG), XTAG (based on TAG (Group, 1998)), the Parallel Grammar Project at Xerox (based on LFG, for English, French and German (LFG ParGram, 1998)), and ERGO (Ergo, 1998) and ALE (Carpenter and Penn,

1998) grammars (both based on HPSG). Typically, such grammars account for phenomena such as subcategorisation, agreement, auxiliary inversion, copula and small-clause constructions, passives, long-distance dependencies, relative clauses, extra-position, PP-attachment, anaphoric binding, adverb placement, etc. The implemented grammars are intended as faithful, yet computationally feasible, implementations of a given linguistic theory. Linguistic processing of such systems requires, at least, the ability to handle complex data-structures (usually (typed) feature structures), under-specification and unification, highly under-specified syntactic schemata (in which the order and number of the daughters may be under-specified), (recursive) lexical rules, and, sometimes, the ability to handle default mechanisms of various kinds.

Below, we propose to develop a linguistically motivated computational grammar for Dutch, comparable in coverage to systems available for English. At the moment, there is no such grammar available for Dutch, although a non-trivial grammar fragment has been developed within the NWO Priority Programme Language and Speech Technology (TST).

Note that developing such a grammar will involve implementing accounts for a number of phenomena which do not occur, or occur to a lesser extent, in English, such as cross-serial dependencies, the word order differences between (verb-final) subordinate clauses and (verb-initial or verb-second) main clauses, and the relative free order of adjuncts in the "Mittelfeld".

We propose to take the TST fragment as a starting point for a more general grammar. Before we explain what is involved in the development of such a grammar, we will first motivate this choice as follows.

4.2 Motivation for TST Grammar

The TST grammar fragment has been developed within the NWO Priority Programme Language and Speech Technology. The immediate goal of the Programme is the development of a demonstrator of a public transport information system, which operates over ordinary telephone lines. This demonstrator is called OVIS, Openbaar Vervoer Informatie Systeem (*Public Transport Information System*). The language of the system is Dutch.

The TST grammar is based on Head-driven Phrase-structure Grammar (HPSG) (Pollard and Sag, 1994). HPSG is a linguistic theory which is an ideal candidate to base a computational grammar on, because it combines a sound linguistic base with a clear formalisation. The linguistic orientation can be inferred from the large number of publications treating many of the world's languages, paying attention to many of the phenomena that have been treated in competing linguistic theories over the last few decades. It is possible to get an idea of the existing body of work by browsing the electronic HPSG bibliography available from `http://www.dfki.de/lt/HPSG/hpsg_bib.html`. A further advantage of HPSG for the current proposal is the fact that Germanic languages, in particular German and Dutch, have been treated extensively.

HPSG combines this theoretical linguistic base with a clear formalisation. Although this does not imply that HPSG is a computational theory of language, or that HPSG grammars can be employed computationally *as is*, it does provide for many computational advantages. Moreover, HPSG has been the starting point for a number of other computational grammars.

The TST grammar uses a high-level formalism (with feature structures, types and inheritance) but is compiled into a Definite-Clause Grammar (DCG). There are a number of benefits for choosing DCG as the basic formalism. Firstly, DCG's provide for a balance between *computational efficiency* on the one hand and *linguistic expressiveness* on the other. Secondly, DCG's are a (simple) member of the class of declarative and constraint-based grammar formalisms. Such formalisms are widely used in linguistic descriptions for NLP. Finally, DCG's are straightforwardly related to context-free grammars. Almost all parsing technology is developed for CFG; porting this technology to DCG is usually possible (although there are many non-trivial problems as well).

Finally, the TST grammar is a successful combination of HPSG and DCG. Even if certain aspects of the grammar have been tuned to the domain of application in TST, it is fair to say that the basic architecture of the grammar is fully general. Moreover, the grammar was successfully applied in a formal evaluation on the TST task: 95% concept accuracy on (previously unseen) user utterances [1].

The development of a general grammar for Dutch based on this fragment will involve a re-implementation of certain parts of the grammar (e.g. the account of unbounded dependencies, PP-attachment, and verb clustering), so as to make it compatible with linguistic accounts, the addition of a number of syntactic construction types (such as passives and relative clauses) which are currently not in the fragment, and expansion of the lexicon, so as to cover the basic vocabulary of Dutch.

Below, we describe the current fragment in some detail, and then go on to describe the activities we foresee in developing a general grammar.

4.3 Current Status

The design and organisation of the TST grammar, as well as many aspects of the particular grammatical analyses, are based on Head-driven Phrase Structure Grammar (Pollard and Sag, 1994). The grammar currently covers the majority of verbal subcategorisation types (intransitives, transitives, verbs selecting a PP, and modal and auxiliary verbs), NP-syntax (including pre- and post-nominal 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).

Most, if not all, linguistic theories used in computational linguistics (including GPSG (Gazdar et al., 1985), HPSG (Pollard and Sag, 1994), LFG (Bresnan, 1982), and unification-based versions of TAG (Vijay-Shanker, 1993), Dependency Grammar (Hellwig, 1986), and Categorial Grammar (Zeevat, Klein, and Calder, 1987; Uszkoreit, 1986))

[1]The details of this evaluation are described in Bonnema, van Noord, and van Zanten (1998)

employ feature-structures to represent linguistic information, and use unification as the single operation to combine feature structures. Feature-structures may be under-specified, and, depending on the computational formalism used, arbitrary complex constraints may be used in the definition of such under-specified feature-structures. In these unification-based or constraint-based grammar formalisms, each word or phrase in the grammar is associated with a (possibly under-specified) feature-structure, in which phonological (or orthographic), syntactic, and semantic information may be bundled. Within Head-driven Phrase Structure Grammar (HPSG), such feature-structures are called *signs*.

The TST grammar makes use of 15 different types of sign, where each type roughly corresponds to a different category in traditional linguistic terminology. For each type of sign, a number of features are defined. For example, for the type NP, the features AGR, NFORM, CASE, and SEM are defined. These features are used to encode the agreement properties of an NP, (morphological) form, case and semantics, respectively.

Typical for lexicalist linguistic theories, such as HPSG and Categorial Grammar, is the fact that they define subcategorisation lexically, by means of features representing the list of elements for which they subcategorize. Such an encoding of valence makes it possible to capture significant generalisations at the level of phrase structure, thus leading, in principle, to a drastic reduction of the number of phrase structure rules that have to be postulated. A property which HPSG shares with GPSG, is the fact that it accounts for long-distance dependencies by means of feature-passing. The current implementation uses a (restricted) version of the account of long-distance dependencies proposed in Pollard and Sag (1994) and Sag (1997). The account of verb-initial and verb-second clauses follows the transformational grammar tradition (Koster, 1975), in that it assumes that verb-initial clauses are structurally similar to verb-final clauses, and by assuming that verbs in main clauses are linked to an empty element (a phonologically empty verbal sign in this case) occupying a clause-final position.

A restriction imposed by the current grammar-parser interface is that each rule must specify the category of its mother and daughters. A rule which specifies that a head daughter may combine with a complement daughter, if this complement unifies with the first element on SC of the head (i.e. a version of the categorial rule for functor-argument application) cannot be implemented directly, as it leaves the categories of the daughters and mother unspecified. Nevertheless, generalisations of this type do play a role in the grammar. We have adopted an architecture for grammar rules similar to that of HPSG, in which individual rules are classified in a hierarchy of *structures* (e.g. *head-complement* and *head-modifier* structures), which are in turn defined in terms of general *principles* (such as the HEAD FEATURE PRINCIPLE and the VALENCE PRINCIPLE).

The TST lexicon is a list of clauses associating a word (or sequence of words) with a specific sign. Constraint-based grammars in general, and lexicalist constraint-based grammars in particular, tend to store lots of grammatical information in the lexicon. This is also true for the TST grammar. A lexical entry for a transitive verb, for instance, not only contains information about the morphological form of this verb, but also contains the features SC and SUBJ for which quite detailed constraints may be defined. Furthermore, for all lexical signs it is the case that their semantics is represented by means of a feature-structure. This structure can also be quite complex. To avoid massive reduplication of identical information in the lexicon, multiple inheritance has been used extensively. This architecture should enable the construction of a lexicon of a much bigger size.

4.4 Development Plans

In van Noord et al. (1999), a detailed account of the syntactic coverage of the TST grammar is given. Although inspired by linguistic theory and designed as much as possible as a general grammar of Dutch, the current fragment is not a general, wide-coverage, grammar of Dutch. In particular, coverage in the lexical domain is limited, and several grammatical constructions are not taken into consideration (e.g. passives) or accounted for only to a certain extent (e.g. the grammar of Dutch verb clusters). The coverage of the grammar is quite satisfactory for the TST application, however. For instance, when evaluating the grammar on a corpus of 1000 transcribed test-sentences, we obtained a semantic concept accuracy of 95%. The evaluation results are presented in detail in Bonnema, van Noord, and van Zanten (1998).

Our development plans aim at building a general, wide-coverage, grammar of Dutch, taking the TST grammar as a starting point. The inheritance-based set-up of the lexicon and the rule set facilitates such a development. The resulting grammar will be used as a concrete test-case for experiments and evaluation in the grammar specialisation and grammar approximation projects.

Below, we give an overview of activities to be carried out as part of the grammar development effort.

4.4.1 Corpus Exploration

Grammar development can benefit enormously from the availability of (annotated) corpora (Skut et al., 1998). An annotated corpus can be used for various kinds of testing (ensuring coverage does not decrease from one version of the grammar to the next), debugging (spotting undesired derivations or rule interactions), and evaluation (measuring syntactic and lexical coverage). Both raw corpora, corpora labelled with part-of-speech tags, and tree-banks (corpora with syntactic annotation) can be used for this purpose. For Dutch, there are a limited number of corpora available for this purpose, such as the corpora of the *Instituut voor Nederlandse Lexicografie*, the Eindhoven (Uit den Boogaard) corpus (den Boogaart, 1975), the TST corpus (Scha et al., 1996), and the Parole corpus (Kruyt, 1995; Parole, 1998).

Furthermore, we expect the corpora within the project for a corpus of spoken Dutch (*Corpus Gesproken Nederlands*) (Levelt, 1998) to be very valuable in this respect. The project aims at the collection of 10 million words of spoken Dutch, annotated (among others) with part of speech and lexical information. Some parts of the corpus will moreover be syntactically annotated. Collaboration with the syntactic and semantic annotation efforts in this corpus initiative will therefore be important. The first results of the *Corpus*

Gesproken Nederlands project will be available in 1999. If these corpora are not suitable for our purposes, then steps will be taken to develop suitably annotated corpora in cooperation with other interested parties.

In addition to the exploration of such corpora, effort will be devoted to the construction of a more systematically constructed set of example sentences. Such *test suites* of considerable sizes already exist for English, German and French (Lehmann et al., 1996). The construction of such a test suite for Dutch will be a very useful evaluation tool both for the proposed project and for other efforts aimed at the construction of Dutch grammars and/or Dutch language technological applications.

4.4.2 Syntactic Coverage

It is clear that important syntactic constructions (such as passives and relative clauses) are missing from the grammar. Furthermore, evaluation on corpora will reveal that a number of other syntactic constructions are still missing in the grammar. These constructions will have to be incorporated in a linguistically motivated fashion, and in a way compatible with the overall architecture of the grammar.

4.4.3 Lexical Coverage

As a consequence of the fact that the TST grammar was used to interface with a speech recogniser (which, for the given task, can typically handle up to a few thousand words), the current lexicon is relatively small. To count as a wide-coverage grammar, it will be necessary to expand the lexicon dramatically. (The XTAG grammar for English, for instance, contains over 300.000 word forms.) Various resources can be used to facilitate lexicon development. The Dutch part of the Celex lexical database (CELEX) (Baayen, Piepenbrock, and van Rijn, 1993) contains morphosyntactic information for over 100.000 lemma's and over 300.000 word forms. Information about syntactic valence is not standardly included in this database, but is available (R. Piepenbrock, p.c.), and is also provided by the lexica developed as part of the projects RBN (Referentiebestand Nederlands) and Parole (Kruyt, 1995). Lexical semantic (conceptual) information is available to some extent in CELEX, and will be available in the EuroWordNet database (Vossen and Bloksma, 1998).

4.4.4 Automated lexical acquisition

Apart from using existing lexical resources, we hope to experiment with techniques for acquiring lexical information automatically. Given a syntactically analyzed corpus, it is relatively straightforward to collect data about the valence of verbs and other lexical items. However, syntactic annotation is not a prerequisite for obtaining this kind of knowledge. For instance, Brent (1991); Manning (1991); Brent (1993); de Lima (1997); Carroll and Rooth (1997) and Carroll, Minnen, and Briscoe (1998) investigate how one may obtain (statistical) lexical (valence) information from an untagged corpus, using no or very coarse grammar rules. Automated acquisition of lexical information is bound to be less precise than manually constructed lexica. However, the approach also has two distinct advantages. First, acquisition can be done for a given domain or application area, thus opening up the possibility of automatically tuning the lexicon for a given domain. Second, the lexical information which is obtained in this way is probabilistic, i.e., not only provides us with information about the possible subcategorisation frames for a given verb, but also tells us which of these frames occurs most frequently. This could be an important aid for disambiguation.

4.4.5 Linguistic Sophistication

The TST fragment contains an account of cross-serial dependency constructions, unbounded dependencies, and modifier attachment, but does not cover these phenomena in their full generality. Furthermore, the grammar format imposes rather strict conditions on the kind of rules that can be formulated. These limitations and constraints are partly a consequence of the application for which the grammar has been developed (which contains only relatively straightforward cases of the phenomena just mentioned) and partly a consequence of restrictions imposed by processing. In order to obtain a general, linguistically motivated, grammar, these phenomena will have to be dealt with in their full generality, and, consequently, certain constraints which are a consequence of processing considerations will have to be removed. It is obvious that this has implications for processing efficiency. Thus, it will become important to investigate how efficiency may be restored, either by approximation or specialisation. In other words, the Dutch grammar to be developed will provide an ideal test-case for the other two research areas.

Chapter 5

Towards a linguistically-informed search tool: lgrep

The proposal focuses on linguistic questions and therefore is part of humanities research, and uses many of the tools and techniques developed in the physical and technical sciences. We believe that in order for the proposed project to be conducted effectively, a methodology should be adopted in which a small number of applications are constructed.

One of the applications that we propose is a linguistically-informed search tool for text corpora (**lgrep**). This application concerns a tool which is capable of searching text corpora (including arbitrary texts on the Internet) on the basis of syntactic criteria. This application fits well with the approach of grammar approximation by finite-state techniques. Such a tool should be useful for (computational) linguists working with corpora, but also as an extension to traditional grammars as used by language learners, to be able to obtain example sentences of particular constructions upon request. Moreover, the tool will be useful for the research proposals discussed in the previous sections.

5.1 What is lgrep?

To start with a simple example, such a tool could be useful to search in text corpora for a particular reading of a given word. For instance, the Dutch word *bar* is ambiguous. It can be a noun (in which case the word means the same as in English), or it can be a degree adverb, as in

(1) a. bar slecht
   quite bad
  b. bar vervelend
   quite boring

In the latter case, *bar* is a negative polarity item. In order to collect example sentences of such negative polarity items (for instance in order to investigate the various contexts in which such negative polarity items can occur), a linguist now typically uses a tool to search for a given word. The resulting set of sentences will then need to be checked by the linguist in order to filter out all the unwanted sentences in which *bar* is used as a noun. Given that in this particular case the wrong examples are much more frequent than the useful examples, this is a time-consuming task. If the tool were to possess linguistic knowledge, as we propose here, it could withdraw the wrong examples itself.

As a much more complicated example, one could ask (in some appropriate format) for sentences in which a prepositional phrase argument has been extra-posed to the right of the verbal group. Note that in order to find appropriate examples the tool should not only be capable of recognising syntactic phrases such as root sentences and prepositional phrases, but the analysis should be deep enough to recognise the difference between prepositional phrases which function as adjuncts and as argument. The tool would then for example return a set of examples:

(2) a. Zou Allende de parlementaire weg verlaten, dan kan hij waarschijnlijk niet meer **rekenen** *op het leger.*
If Allende abolishes parliament, then he probably cannot rely on the army anymore.
b. De Ierse radio- en televisie-autoriteiten hebben medegedeeld dat zij **wachten** *op de beslissing van de Europese Omroep Unie* voor zij plannen zullen gaan maken voor de organisatie van het Eurosongfestival van volgend jaar.
Irish radio and television officials have announced that they will wait for a decision by the European Broadcast Association before making plans for the organisation of the Eurovision Song-contest of next year.
c. Zeker Nederlanders moeten **rekenen** *op een hoogteverschil van gemiddeld tweeduizend meter.*
The Dutch in particular should expect a difference in altitude of about 2000 meters.
d. Misschien vanwege de duizenden dichters, schrijvers en schilders, studenten en verliefden die hier komen en die allemaal **hopen** *op iets onzegbaars.*
Maybe because of the thousands of poets, writers, and painters and lovers who come here and all hope for something unsayable
e. Andere vrouwen zullen haar gezag moeten aanvaarden om te kunnen **rekenen** *op haar genegenheid.*
Other women will have to accept her authority in order to count on her sympathy.

Another example usage of the tool could be to identify examples of verb raising constructions in which an adjunct takes narrow scope, i.e. it is an adjunct modifying one of the verbs embedded in the verb cluster (cf. van Noord and Bouma (1994)):

(3) a. Dit vertelde onlangs Mamie Eisenhouwer, de weduwe van de 34e president van de Verenigde Staten, die *in haar periode in het Witte Huis* dezelfde moeilijkheden gehad schijnt te hebben als alle andere huisvrouwen overal op de wereld.
This was told by Mamie Eisenhouwer, the widow of the 34th president of the United States, who seems to have had the same difficulties during her stay in the White House as all other house wives around the world.
b. Een tweede belangrijke ingreep is het weglaten van de inleiding van het stuk

waarin de God Indra zijn dochter toestemming geeft naar de aarde te vertrekken, een inleiding die Strindberg overigens zelf *pas vier jaar na het stuk* schijnt te hebben geschreven om zijn publiek enige achtergrond te geven voor het plotselinge verschijnen van een godendochter.
A second important adaptation is the omission of the introduction of the piece in which the God Indra allows his daughter to leave for the earth, an introduction that Strindberg is said to have written only four years after the piece, in order to provide the audience with some background knowledge for the sudden appearance of the daughter of a God.

c. Zijn belangrijkste verzuim was dat hij een deel van zijn instructies *op Cape Kennedy* heeft laten liggen.
His most important neglect was that he left part of his instructions at Cape Kennedy.

d. Ook al een echte Wieringa, die zich niet *in een hoekje* wil laten drijven ?
Another real Wieringa, who doesn't want to be pushed in a corner?

It should be clear however that this tool only has limited knowledge of syntactic constructions (otherwise creating the tool would presuppose knowledge that the use of the tool seeks to discover). We envisage that the tool provides an extension of regular expressions capable of recognising matching syntactic brackets, major syntactic categories, and grammatical functions such as subject, (in)direct object and modifier.

The novel feature of this application (in contrast with tools such as `tgrep`) will be that it can search in text corpora which need not be syntactically annotated. This has the obvious advantages that much more corpus material is available (especially now that large amounts of text corpora are available through the Internet). A further possible advantage is that it might be easier to change linguistic analyses in a grammar, rather than in an annotated corpus. Of course, the challenge is to make this application fast enough for it to be of any practical use. Moreover, we believe that even if only a small fraction of the described functionality can be achieved, then this could be a useful tool for linguists working with large text corpora.

5.2 The construction of lgrep

The search tool we propose will be somewhat reminiscent of the UNIX tool `grep`. `grep` can be used to search in text files for lines matching a given regular expression. In **lgrep** there are two important differences:

- **lgrep** searches for sentences, rather than lines
- **lgrep** provides for a much richer regular expression language
- **lgrep** provides for an extendible regular expression language
- **lgrep** has facilities to integrate it with Internet search engines

The tool will thus tokenize a given text file into a series of sentences, rather than lines. The technology for this tokenization task exists, even if it is not perfect (for instance Grefenstette and Tapanainen (1994)). For this reason we will enable an architecture in which a useful default tokenization scheme can be augmented with user specified alternatives.

The regular expression language that **lgrep** should support will be much more powerful than the regular expression languages typically found in tools such as `grep` and Perl. For instance, an interface with a part-of-speech tagger is foreseen which will provide for the part-of-speech labels as nullary regular expression operators. For instance, the operator `noun` will denote all words with part-of-speech *noun*.

On top of this, the regular expression language might define more complex linguistic categories as regular expression operators such as `np`. As a very first approximation, `np` could be defined as `[det^,adj*, noun+]` (in the FSA Utilities notation; this expression denotes an optional determiner followed by any number (including zero) of adjectives, followed by one or more nouns). Ultimately, we hope to be able to extract the definitions of such operators by means of finite-state approximation techniques from a general grammar of Dutch.

The regular expression operators provided by the regular expression language should be considered default implementations of these. In order that the tool be easily adaptable to different linguistic insights and/or different needs it is of extreme importance that the regular expression operators can be redefined, and that new operators can be defined (typically in terms of existing regular expression operators).

Finally, **lgrep** will be integrated with Internet search engines in order to be able to regard the Internet as a large text corpus. We have already implemented a small tool which is capable to search for sentences containing some given word on arbitrary web sites. This works fully automatically as follows. Firstly a search query is sent to a search engine. The resulting pages are automatically scanned for relevant links. These links are visited and the corresponding pages are tokenised into sentences. The sentences are scanned for occurrences of the search query. A similar idea is described in van Oostendorp and van der Wouden (1998). We plan to integrate **lgrep** in a similar fashion in order to treat the Internet as a large text corpus. As a more ambitious task, we propose to investigate the possibilities of a search engine which is capable to search for linguistic patterns directly.

References

Abney, Steven. 1995. Partial parsing via finite-state cascades. In John Carroll, editor, *Workshop on Robust Parsing; Eight European Summer School in Logic, Language and Information*, pages 8–15.

Alshawi, Hiyan, editor. 1992. *The Core Language Engine.* ACL-MIT press, Cambridge Mass.

Baayen, R. H., R. Piepenbrock, and H. van Rijn. 1993. *The CELEX Lexical Database (CD-ROM).* Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.

Baker, J. K. 1979. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech Communication Papers Presented ath the 97th Meething of the Acoustical Society of America*, MIT Cambridge.

Beesley, Kenneth R. 1998. Constraining separated morphotactic dependencies in finite-state grammars. In *Finite-state Methods in Natural Language Processing*, pages 118–127, Ankara.

Black, A.W. 1989. Finite state machines from feature grammars. In *International Workshop on Parsing Technologies*, pages 277–285, Pittsburgh.

Black, E., S. Eubank, H. Kashioka, R. Garside, G. Leech, and D. Magerman. 1996. Beyond skeleton parsing: Producing a comprehensive large-scale general-english treebank with full grammatical analysis. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages pages 107–112, Copenhagen.

Black, Ezra, Stephen Eubank, Hideki Kashioka, and David Magerman. 1997. Probabilistic parsing of unrestricted english text, with a highly-detailed grammar. In Joe Zhou and Kenneth Church, editors, *Proceedings of the Fifth Workshop on Very Large Corpora*, Beijing and Hong Kong.

Bod, Rens. 1995. *Enriching Linguistics with Statistics: Performance Models of Natural Language.* Ph.D. thesis, University of Amsterdam.

Bonnema, Remko, Gertjan van Noord, and Gert Veldhuizen van Zanten. 1998. Evaluation results NLP components OVIS2. Technical Report 57, NWO Priority Programme Language and Speech Technology.

den Boogaart, P. C. Uit. 1975. *Woordfrequenties in geschreven en gesproken Nederlands.* Oosthoek, Scheltema & Holkema, Utrecht. Werkgroep Frequentie-onderzoek van het Nederlands.

Bouma, Gosse and Ineke Schuurman. 1998. De positie van het Nederlands in taal- en spraaktechnologie. De Nederlandse Taalunie.

Brent, Michael R. 1991. Automatic acquisition of subcategorization frames from untagged text. In *29th Annual Meeting of the Association for Computational Linguistics*, pages 209–214, Berkeley.

Brent, Michael R. 1993. From grammar to lexicon. *Computational Linguistics*, 19(2):243–262.

Bresnan, Joan, editor. 1982. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge Mass.

Brill, Eric. 1992. A simple rule-based part-of-speech tagger. In *Proceedings Third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy.

Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4):543–566.

Brill, Eric and Philip Resnik. 1994. A rule-based appraoch to prepositional phrase attachment disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, Kyoto.

Briscoe, Ted and John Carroll. 1993. Generalized probabilistic LR parsing. *Computational Linguistics*, 19(1).

Briscoe, Ted, Claire Grover, Bran Boguraev, and John Carroll. 1987. A formalism and environment for the development of a large grammar of english. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 703–708, Milan.

Carpenter, Bob and Gerald Penn, 1998. *The Attribute Logic Engine User Guide.* Version 3.1 Beta.

Carroll, Glenn and Mats Rooth. 1997. Valence induction with a head-lexicalized PCFG. www.ims.uni-stuttgart.de/~mats.

Carroll, John, Guido Minnen, and Ted Briscoe. 1998. Can subcategorisation probabilities help a statistical parser? In *Proceedings 6th Workshop on Very Large Corpora*, Montreal.

Chanod, Jean-Pierre and Pasi Tapanainen. 1996. A robust finite-state grammar for french. In John Carroll, editor, *Workshop on Robust Parsing*, Prague.

Chomsky, Noam. 1963. Formal properties of grammars. In R. Duncan Luce, Robert R. Bush, and Eugene Galanter, editors, *Handbook of Mathematical Psychology; Volume II.* John Wiley, pages 323–418.

Chomsky, Noam. 1964. On the notion 'rule of grammar'. In Jerry E. Fodor and Jerrold J. Katz, editors, *The Structure of Language; Readings in the Philosophy of Language.* Prentice Hall, pages 119–136.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax.* MIT press, Cambridge Mass.

Collins, Michael. 1996. A new statistical parser based on bigram lexical dependencies. In *34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, Madrid.

Collins, Michael and J. Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge MA.

Daciuk, Jan. 1998. *Incremental Construction of Finite-state Automata and Transducers, and their Use in the Natural Language Processing.* Ph.D. thesis, Technical University of Gdańsk.

Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345, Copenhagen.

Eisner, Jason M. 1997. An empirical comparison of probability models for dependency grammar. Technical report, CIS department, University of Pennsylvania. cmp-lg/9706004.

Ergo. 1998. English resource grammar online; a multi-purpose broad-coverage computational grammar of english. hpsg.stanford.edu/hpsg/ergo.html.

Evans, Edmund Grimley. 1997. Approximating context-free grammars with a finite-state calculus. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 452–459, Madrid.

Evans, Edmund Grimley, George Anton Kiraz, and Stephen G. Pulman. 1996. Compiling a partition-based two-level formalism. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, Copenhagen.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar.* Blackwell.

Grefenstette, Gregory. 1996. Light parsing as finite-state filtering. In *EACI 1996 Workshop Extended Finite-State Models of Language*, Budapest.

Grefenstette, Gregory and Pasi Tapanainen. 1994. What is a word, what is a sentence? problems of tokenization. Technical Report MLTT-004, Xerox Research Centre Europe, MLTT.

Grishman, Ralph, Ngo Thanh Nhan, Elaine Marsh, and Lynette Hirschman. 1984. Automated determination of sublanguage syntactic usage. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics (COLING)*, Stanford.

Gross, Maurice. 1989. *The use of Finite Automata in the Lexical Representation of Natural Language.* Lecture Notes in Computer Science. Springer Verlag, Berlin.

Gross, Maurice. 1997. Local grammars. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing.* MIT Press, Cambridge, pages 330–354.

Group, The XTAG Research, 1998. *A Lexicalized Tree Adjoining Grammar for English.* Institute for Research in Cognitive Science, University of Pennsylvania. cs.CL/9809024.

Grover, Claire, John Carroll, and Ted Briscoe. 1993. The ALVEY natural language tools grammar (4th release). Technical Report 284, Computer Laboratory, Cambridge University UK.

Hellwig, Peter. 1986. Dependency unification grammar. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING)*, pages 195–198, Bonn.

Huybrechts, Riny. 1984. The weak inadequacy of context-free phrase structure grammars. In Ger de Haan, Mieke Trommelen, and Wim Zonneveld, editors, *Van Periferie naar Kern.* Foris.

Jelinek, F., J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, and S. Roukos. 1994. Decision tree parsing using a hidden derivation model. In *Proceedings of the 1994 Human Language Technology Workshop (ARPA)*, pages 272–277.

Jelinek, Frederick. 1998. *Statistical Methods for Speech Recognition.* MIT Press.

Johnson, C. Douglas. 1972. *Formal Aspects of Phonological Description.* Mouton, The Hague.

Johnson, Mark. 1998. Finite-state approximation of constraint-based grammars using left-corner grammar transforms. In *COLING-ACL '98. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics. Proceedings of the Conference*, Montreal.

Johnson-Laird, Philip N. 1983. *Mental Models.* Harvard University Press.

Joshi, Aravind K. and B. Srinivas. 1994. Disambiguation of super parts of speech (supertags): Almost parsing. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, Kyoto, Japan.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Karlsson, Fred, Atro Voutilainen, Juha Heikkila, and Atro Anttila. 1995. *Constraint Grammar, A Language-independent System for Parsing Unrestricted Text.* Mouton de Gruyter.

Karttunen, Lauri. 1995. The replace operator. In *33th Annual Meeting of the Association for Computational Linguistics*, M.I.T. Cambridge Mass.

Karttunen, Lauri. 1998. The proper treatment of optimality theory in computational phonology. In *Finite-state Methods in Natural Language Processing*, pages 1–12, Ankara.

Kiraz, George Anton. 1997. Compiling regular formalisms with rule features into finite-state automata. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, Madrid.

Koskenniemi, Kimmo. 1983. Two-level morphology: a general computational model for word-form recognition and production. Technical Report 11, Department of General Linguistics, University of Helsinki.

Koster, Jan. 1975. Dutch as an SOV language. *Linguistic Analysis*, 1.

Kowaltowski, Tomasz, Cláudio L. Lucchesi, and Jorge Stolfi. 1993. Minimization of binary automata. Technical Report Relatório Téchnico DCC-22/93, Departamento de Ciência de Computa cão, Universidade Estudual de Campinas.

Krauwer, Steven and Louis des Tombe. 1981. Transducers and grammars as theories of language. *Theoretical Linguistics*, 8:173–202.

Kruyt, J. G. 1995. Nationale tekstcorpora in internationaal perspectief. *Forum der Letteren*, 36(1):47–58.

Lee, Seungmi and Key-Sun Choi. 1998. Automatic acquisition of language model based on head-dependent relation between words. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montreal.

Lehmann, Sabine, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, Hervé Compagnion, Judith Baur, Lorna Balkan, and Doug Arnold. 1996. Tsnlp—test suites for natural language processing. cmp-lg/9607018.

Levelt, Pim. 1998. Corpus gesproken Nederlands. zie ook www.elis.rug.ac.be/cgn/.

LFG ParGram. 1998. Parallel grammar project. www.parc.xerox.com/istl/groups/nltt/-pargram/.

de Lima, Erika F. 1997. Acquiring german prepositinal subcategorization frames from corpora. In Joe Zhou and Kenneth Church, editors, *Proceedings of the Fifth Workshop on Very Large Corpora*, Beijing and Hong Kong.

Linguistic Data Consortium. `tgrepdoc`*—documentation for* `tgrep`. University of Pennsylvania, Philadelphia.

Magerman, David M. 1995. Statistical decision-tree models for parsing. In *33th Annual Meeting of the Association for Computational Linguistics*, MIT Cambridge.

Manning, Chris. 1991. Automatic acquisition of a large subcategorization dictionary from corpora. In *29th Annual Meeting of the Association for Computational Linguistics*, pages 235–242, Berkeley.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 19(2).

Marcuss, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *Proceedings of the 1994 Human Language Technology Workshop (ARPA)*, pages 110–115.

Miller, George and Noam Chomsky. 1963. Finitary models of language users. In R. Luce, R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology. Volume 2.* John Wiley.

Mohri, Mehryar. 1996. On some applications of finite-state automata theory to natural language processing. *Natural Language Engineering*, 2:61–80. Originally appeared in 1994 as Technical Report, institut Gaspard Monge, Paris.

Mohri, Mehryar, Michael Riley, Don Hindle, Andrej Ljolje, and Fernando Pereira. 1998. Full expansion of context-dependent networks in large vocabulary speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98)*, Seattle.

Mohri, Mehryar and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz.

Nederhof, Mark-Jan. 1997. Regular approximations of CFLs: A grammatical view. In *International Workshop on Parsing Technologies*, pages 159–170.

Nederhof, Mark-Jan. 1998. Context-free parsing through regular approximation. In *Finite-state Methods in Natural Language Processing*, pages 13–24, Ankara.

Nederlands, Referentiebestand. 1998. Referentiebestand nederlands (rbn). www.taalunie.org/%5F%5F/werkt/rbn.html.

van Noord, Gertjan. 1997. FSA Utilities: A toolbox to manipulate finite-state automata. In Darrell Raymond, Derick Wood, and Sheng Yu, editors, *Automata Implementation.* Springer Verlag, pages 87–108. Lecture Notes in Computer Science 1260.

van Noord, Gertjan. 1998. The treatment of epsilon moves in subset construction. In *Finite-state Methods in Natural Language Processing*, Ankara. cmp-lg/9804003.

van Noord, Gertjan and Gosse Bouma. 1994. Adjuncts and the processing of lexical rules. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, pages 250–256, Kyoto. cmp-lg/9404011.

van Noord, Gertjan and Gosse Bouma. 1997. Hdrug, A flexible and extendible development environment for natural language processing. In *Proceedings of the EACL/ACL workshop on Environments for Grammar Development, Madrid.*

van Noord, Gertjan, Gosse Bouma, Rob Koeling, and Mark-Jan Nederhof. 1999. Robust grammatical analysis for spoken dialogue systems. *Journal of Natural Language Engineering.* To appear; 48 pages.

Oflazer, Kemal. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89.

van Oostendorp, Marc and Ton van der Wouden. 1998. Corpus internet. In *Nederlandse Taalkunde.* to appear.

Parole. 1998. Le-parole, preparatory action for linguistic resources organisation for language engineering. project summary. www2.echo.lu/langeng/en/le2/le-parole/le-parole.html.

Pereira, Fernando and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *30th Annual Meeting of the Association for Computational Linguistics*, Newark, Delaware.

Pereira, Fernando C. N. and Michael D. Riley. 1996. Speech recognition by composition of weighted finite automata. cmp-lg/9603001.

Pereira, Fernando C. N. and R. N. Wright. 1991. Finite-state approximation of phrase structure grammars. In *29th Annual Meeting of the Association for Computational Linguistics*, Berkeley.

Pereira, Fernando C. N. and Rebecca N. Wright. 1997. Finite-state approximation of phrase-structure grammars. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing.* MIT Press, Cambridge, pages 149–173.

Pollard, Carl and Ivan Sag. 1994. *Head-driven Phrase Structure Grammar.* University of Chicago / CSLI.

Prince, Alan and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Technical Report TR-2, Rutgers University Cognitive Science Center, New Brunswick, NJ. MIT Press, To Appear.

Pulman, Steve. 1986. Grammars, parsers, and memory limitations. *Language and Cognitive Processes*, 1(3):197–225.

Ratnaparkhi, A., J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the 1994 Human Language Technology Workshop (ARPA).*

Rayner, Manny. 1988. Applying explanation-based generalization to natural language processing. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, Kyoto.

Rayner, Manny and David Carter. 1996. Fast parsing using pruning and grammar specialization. In *34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz.

Rayner, Manny and Christer Samuelsson. 1990. Using explanation-based learning to increase performance in a large NL query system. In *Proceedings DARPA Speech and Natural Language Workshop.* Morgan Kaufmann.

Resnik, Philip. 1992. Left-corner parsing and psychological plausability. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, Nantes.

Roche, Emmanuel. 1997. Parsing with finite-state transducers. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing.* MIT Press, Cambridge, pages 241–281.

Roche, Emmanuel and Yves Schabes. 1995. Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, 21(2).

Rood, C.M. 1996. Efficient finite-state approximation of context free grammars. In A. Kornai, editor, *Extended Finite State Models of Language*, Proceedings of the ECAI'96 workshop, pages 58–64, Budapest University of Economic Sciences, Hungary.

Rosenkrantz, D. J. and P. M. Lewis-II. 1970. Deterministic left corner parsing. In *IEEE Conference of the 11th Annual Symposium on Switching and Automata Theory*, pages 139–152.

Sag, Ivan. 1997. English relative clause constructions. *Journal of Linguistics.* to appear.

Samuelsson, Christer. 1994. *Fast Natural-Language Parsing Using Explanation-Based Learning.* Ph.D. thesis, Swedish Institute of Computer Science, Kista.

Scha, Remko. 1990. Taaltheorie en taaltechnologie; competence en performance. In *Computertoepassingen in de Neerlandistiek.* Landelijke Vereniging van Neerlandici (LVVN Jaarboek), Almere.

Scha, Remko, Remko Bonnema, Rens Bod, and Khalil Simaán. 1996. Disambiguation and interpretation of wordgraphs using data oriented parsing. Probabilistic natural language processing in the NWO priority programme on language and speech technology. October 1996 deliverable. Technical Report 31, NWO Priority Programme Language and Speech Technology. Chapter 4.

Shieber, Stuart M. 1985. Evidence against the non-context-freeness of natural language. *Linguistics and Philosophy*, 8.

Silberztein, Max D. 1997. The lexical analysis of natural language. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing.* MIT Press, Cambridge, pages 175–203.

Skut, Wojciech, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A linguistically interpreted corpus of german newspaper text. In *ESSLLI-98 Workshop on Recent Advances in Corpus Annotation*, Saarbrücken. cmp-lg/9807008.

Srinivas, B. and A. Joshi. 1995. Some novel applications of explanation-based learning to parsing lexicalized tree-adjoining grammars. In *33th Annual Meeting of the Association for Computational Linguistics*, MIT Cambridge.

Stearns, R.E. 1967. A regularity test for pushdown machines. *Information and Control*, 11:323–340.

Tapanainen, Pasi. 1997. Applying a finite-state intersection grammar. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing.* MIT Press, Cambridge, pages 311–327.

Ullian, J.S. 1967. Partial algorithm problems for context free languages. *Information and Control*, 11:80–101.

Uszkoreit, Hans. 1986. Categorial unification grammar. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING)*, Bonn.

Valiant, L.G. 1975. Regularity and related problems for deterministic pushdown automata. *Journal of the ACM*, 22(1):1–10.

Vijay-Shanker, K. 1993. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4).

Vossen, Piek and Laura Bloksma. 1998. Categories and classifications in eurowordnet. In *Proceedings of First International Conference on Language Resources and Evaluation*, Granada.

Voutilainen, Atro. 1997. Designing a (finite-state) parsing grammar. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing.* MIT Press, Cambridge, pages 283–310.

Voutilainen, Atro and Pasi Tapanainen. 1993. Ambiguity resolution in a reductionist parser. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht.

Zavrel, Jakub and Walter Daelemans. 1997. Memory-based learning: Using similarity for smoothing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 436–443, Madrid.

Zeevat, Henk, Ewan Klein, and Jo Calder. 1987. Unification categorial grammar. In Nicholas Haddock, Ewan Klein, and Glyn Morrill, editors, *Categorial Grammar, Unification Grammar and Parsing.* Centre for Cognitive Science, University of Edinburgh. Volume 1 of Working Papers in Cognitive Science.