Acyclic Context-sensitive Grammars

Erik Aarts*

Research Institute for
Language and Speech
Trans 10
3512 JK Utrecht
The Netherlands

Dept. of Mathematics
and Computer Science
Plantage Muidergracht 24
1018 TV Amsterdam
The Netherlands

Abstract

A grammar formalism is introduced that generates parse trees with crossing branches. The uniform recognition problem is NP-complete, but for any fixed grammar the recognition problem is polynomial.

1 Introduction

In this article we propose a new type of context-sensitive grammars, the *acyclic* context-sensitive grammars (ACSG's). Acyclic context-sensitive grammars are context-sensitive grammars with rules that have a "real rewrite part", which must be context-free and acyclic, and a "context part". The context is present on both sides of a rule. The motivation for introduction of ACSG's is that we want a formalism which

- allows parse trees with crossing branches
- is computationally tractable
- has a simple definition.

We enrich context-free rewrite rules with context and this leads to a simple definition of a formalism that generates parse trees with crossing branches. In order to gain efficiency, we add a restriction: the context-free rewrite part of the grammar must be acyclic. Without this restriction, the complexity would be the same as for unrestricted CSG's (PSPACE-complete). With the acyclicity restriction we get the same results as for *growing* CSG's (Dahlhaus and Warmuth 1986, Buntrock 1993), i.e., NP-completeness for uniform recognition and polynomial time for fixed grammars.

Possible applications are in the field of computational linguistics. In natural language one often finds sentences with so-called *discontinuous constituents* (constituents separated by other material). ACSG's can be used to describe such constructions. Most similar attempts (Pereira 1981, Johnson 1985, Bunt 1988, Abramson and Dahl 1989) allow an arbitrary distance between two parts of a discontinuous constituent. This is not allowed in ACSG's. For unbounded dependencies like *wh-movement* we either have to extend the formalism (allow arbitrary context) or introduce the *slash-feature*.

The acyclicity of the grammar does not seem to form a problem for the generative capacity necessary to describe natural language. Acyclicity is closely related to the *off-line parsability constraint* (Johnson 1988). Constituent structures satisfy the off-line parsability constraint iff

- it does not include a non-branching dominance chain in which the same category appears twice, and

*The author was sponsored by project NF 102/62-356 ('Structural and Semantic Parallels in Natural Languages and Programming Languages'), funded by the Netherlands Organization for the Advancement of Research (NWO).

- the empty string ϵ does not appear as the righthand side of any rule.

The off-line parsability constraint has been motivated both from the linguistic perspective and computationally. Kaplan and Bresnan (1982) say that "vacuously repetitive structures are without intuitive or empirical motivation" (Johnson 1988). ACSG's satisfy the off-line parsability constraint: they have no cycles and no ϵ-rules.

The goal of designing a formalism that is computationally tractable is only achieved partially. We show that the uniform recognition problem is NP-complete. For any fixed grammar, however, the recognition problem is polynomial (in the sentence length).

The definition of ACSG is simple because it is a standard rewrite grammar. Derivability is defined by successive string replacement. This definition is, e.g., simpler than the definition of Discontinuous Phrase Structure Grammar (Bunt 1988). The main difference between DPSG and ACSG is that in ACSG constituents are "moved" when a context-sensitive rule is applied. In DPSG trees with crossing branches are described "static": the shape of a tree is described with node admissibility constraints.

The structure of this article is as follows. First we define acyclic CSG's, growing CSG's and quasi-growing CSG's formally. Then we present some results on the generative power of these classes of grammars and on their time complexity. We end with a discussion on the uniform recognition problem vs. the recognition problem for fixed grammars.

2 Definitions

2.1 Context-sensitive Grammars

Definition 2.1 *A* grammar *is a quadruple* $\langle V, \Sigma, S, P \rangle$*, where* V *is a finite set of symbols and* $\Sigma \subset V$ *is the set of terminal symbols.* Σ *is also called the* alphabet. *The symbols in* $V \setminus \Sigma$ *are called the nonterminal symbols.* $S \in V \setminus \Sigma$ *is a start symbol and* P *is a set of production rules of the form* $\alpha \rightarrow \beta$*, with* $\alpha, \beta \in V^*$*, where* α *contains at least one nonterminal symbol.*

Definition 2.2 *A grammar is* context-free *if each rule is of the form* $Z \rightarrow \gamma$ *where* $Z \in V \setminus \Sigma$*;* $\gamma \in V^*$*. A* cycle *in a context-free grammar is a set of symbols* $a_1, \ldots, a_n$ *with* $a_i \rightarrow a_{i+1} \in P$ *for all* $1 \leq i < n$ *and* $a_1 = a_n$*.*

Definition 2.3 *A grammar is* context-sensitive *if all rules are of the form* $\alpha \rightarrow \beta$*, with* $|\alpha| \leq |\beta|$*.*

Definition 2.4 Derivability *(*$\Rightarrow$*) between strings is defined as follows:* $u\alpha v \Rightarrow u\beta v$ $(u, v, \alpha, \beta \in V^*)$ *iff* $(\alpha, \beta) \in P$*. The transitive closure of* $\Rightarrow$ *is denoted by* $\stackrel{+}{\Rightarrow}$*. The reflexive transitive closure of* $\Rightarrow$ *is denoted by* $\stackrel{*}{\Rightarrow}$*.*

Definition 2.5 *The* language *generated by* G *is defined as* $L(G) = \{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}$*.*

Definition 2.6 *A derivation of a string* δ *is a sequence of strings* $x_1, x_2, \ldots, x_n$ *with* $x_1 = S$*, for all* i $(1 \leq i < n)$ $x_i \Rightarrow x_{i+1}$ *and* $x_n = \delta$*.*

2.2 Labeled Context-sensitive Grammars

Context-sensitive grammars have just been described as grammars with rules of the form $\alpha \rightarrow \beta$ for which $|\alpha| \leq |\beta|$. There is an alternative definition where *context* is used. In this definition, rules are of the form $\alpha Z \beta \rightarrow \alpha\gamma\beta$ $(Z \in V, \alpha, \beta, \gamma \in V^*)$. In this format, α and β are *context*, and $Z \rightarrow \gamma$ is called the *context-free backbone*. It is known that the two formats are weakly equivalent (Salomaa 1973, pp. 15,82). We introduce here a third form, which is a kind of a mix between the other two. Instead of having context on the outside of the rule, we can also have context inside. Suppose we have the rule "A rewrites to B C in the context D". In a standard context-sensitive grammar, this can only be expressed as D A $\rightarrow$ D B C or as A D $\rightarrow$ B C D. We are going to allow that the D is in between the B and the C. A possible rule is D A $\rightarrow$ B D C. In the rules of the form $\alpha Z \beta \rightarrow \alpha\gamma\beta$ it is not always clear which symbols are the context and which symbols form the context-free backbone. E.g., in the grammar rule

A A $\rightarrow$ A B A it is not clear what the context is. We are going to indicate the context with brackets. The rule A A $\rightarrow$ A B A can be written as [A] A $\rightarrow$ [A] B A or as A [A] $\rightarrow$ A B [A] (the context is between brackets). In the new form, where the context can be "scattered", brackets are not enough. Therefore we introduce labels for the context symbols. The labels left and right of the arrow must be the same of course. The rule [A] A $\rightarrow$ [A] B A is written now as A_1 A $\rightarrow$ A_1 B A and A [A] $\rightarrow$ A B [A] is written as A A_1 $\rightarrow$ A B A_1. The rule D A $\rightarrow$ B D C can be written as D_1 A $\rightarrow$ B D_1 C.

This can be formalized as follows.

Definition 2.7 *An* acyclic context-sensitive grammar *G is a quadruple $\langle V, \Sigma, S, P\rangle$, where V and Σ are, again, sets of symbols and terminal symbols. V_l and Σ_l (labeled symbols and terminals) denote $\{\langle a, k\rangle \mid a \in V, 1 \leq k \leq K\}$ and $\{\langle a, k\rangle \mid a \in \Sigma, 1 \leq k \leq K\}$ respectively, where K depends on the particular grammar under consideration. $S \in V \setminus \Sigma$ is a start symbol and P is a set of production rules of the form $\alpha \rightarrow \beta$, with $\alpha \in V_l^* V V_l^*, \beta \in (V \cup V_l)^*$. The left hand side contains exactly one unlabeled symbol, the right hand side at least one. For all production rules it holds that $|\alpha| \leq |\beta|$. The labeled symbols in a rule are called the context.*

There are three conditions:

- *If we leave out all members of V_l (the context) from the production rules we obtain a context-free grammar. This is the context-free backbone.*
- *If we leave out all members of V (the backbone) from the production rules we obtain a grammar that has permutations only. This is the context part. All context symbols in the rules should have different labels.*
- *If we remove all labels in the rules, i.e., we replace all symbols $\langle a, b\rangle$ by a, we get an ordinary context-sensitive grammar G'. We define that $L(G) = L(G')$.*

2.3 Acyclic Context-sensitive Grammars

Acyclic context-sensitive grammars, or ACSG's, are context-free grammars with an "acyclic contextfree backbone". This is formalized as follows.

Definition 2.8 *An* acyclic context-sensitive grammar *is a labeled context-sensitive grammar that fullfills the following condition:*

- *If we leave out all members of V_l from the production rules we must obtain a finitely ambiguous context-free grammar, i.e. a grammar for which $|\alpha| = 1$ and $|\beta| \geq 1$ and that contains no cycles.*

An example of a rule of an acyclic CSG is (pairs of $\langle N, L\rangle$ are written as N_L):

$$A_1\ B_2\ M\ C_3\ D_4 \rightarrow C_3\ K\ B_2\ A_1\ L\ M\ D_4$$

The context-free backbone is M $\rightarrow$ K L M. The context part is A_1 B_2 C_3 D_4 $\rightarrow$ C_3 B_2 A_1 D_4. The rule without labels is A B M C D $\rightarrow$ C K B A L M D. Another example is given after the definition of growing context-sensitive grammars.

2.4 Growing Context-sensitive Grammars

The definition of growing CSG's (GCSG's) is pretty simple: the lefthand side of a rule must be shorter than the righthand side. For the precise definition we follow Buntrock (1993):

Definition 2.9 *A context-sensitive grammar $G = \langle V, \Sigma, S, P\rangle$ is growing if*

1. $\forall(\alpha \rightarrow \beta) \in P : \alpha \neq S \Rightarrow |\alpha| < |\beta|$, *and*
2. *S does not appear on the right hand side of any rule.*

We also define grammars which are growing with respect to a weight function. These were introduced in (Buntrock 1993).

Definition 2.10 *We call a function* $f : \Sigma^* \to N$ *a* weight function *if* $\forall a \in \Sigma : f(a) > 0$ *and* $\forall w, v \in \Sigma^* : f(w) + f(v) = f(wv)$.

Now we can define *quasi-growing* grammars:

Definition 2.11 *Let* $G = \langle V, \Sigma, S, P \rangle$ *be a grammar and* $f : V^* \to N$ *a weight function. We call* G *quasi-growing if* $f(\alpha) < f(\beta)$ *for all productions* $(\alpha \to \beta) \in P$.

Quasi-growing context-sensitive grammars (QGCSG's) are grammars that are both quasi-growing and context-sensitive.

3 An Example

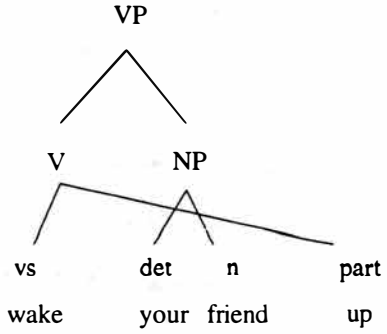
This section contains an example based on natural language of how ACSG's work. It is taken from Bunt (1988). Assume we have the following grammar. It has one rule that is not context-free.

VP	→	V	NP		vs	→	wake
$V\ NP_1$	→	vs	NP_1	part	det	→	your
NP	→	det	n		n	→	friend
					part	→	up

Table 1: An example grammar

Suppose we have the sentence: "Wake your friend up". This is a VP:
VP ⇒ V NP ⇒ vs NP part ⇒ vs det n part ⇒* wake your friend up

The corresponding parse tree is:



We see that we have a context-free backbone which allows us to draw parse trees (or structure trees) and that the scattered context causes branches in the trees to cross.

4 Properties of Acyclic Context-sensitive Grammars

4.1 Generative Power of Acyclic CSG's

In this section we discuss the relation ACSG's between and GCSG's and we sketch their position in the Chomsky hierarchy.

Theorem 4.1 *ϵ-free CFL $\subset$ ACSL*

Proof: if the cfl is generated by an acyclic cfg without empty productions we do not have to do anything. This cfg *is* an acyclic csg. If the cfg contains cycles we can remove them. A cycle can be removed by introduction of a new symbol. This symbol rewrites to any member of the cycle. Any cfg with empty productions can be changed into a cfg without empty productions that generates the same language. There's one exception here: languages containing the empty string can not be generated. Therefore acyclic context-sensitive grammars generate all cfl's that do not contain the empty word. □

Theorem 4.2 *ACSL* $\neq$ *CFL*

Proof: ACSG's are able to generate languages that are not context-free. One example is the language $\{a^n b^{2^n} c^n \mid n \geq 1\}$. This language is generated by the grammar:

S	$\rightarrow$	A B B C	A	$\rightarrow$	a
B X_1	$\rightarrow$	X_1 B B	B	$\rightarrow$	b
B C_1	$\rightarrow$	X B B C C_1	C	$\rightarrow$	c
A_1 X B_2	$\rightarrow$	A_1 A B_2			

Table 2: Grammar for $\{a^n b^{2^n} c^n \mid n \geq 1\}$

A derivation of " a a b b b b c c " is:

S $\Rightarrow$ A B B C $\Rightarrow$ A B X B B C C $\Rightarrow$ A X B B B B C C $\Rightarrow$ A A B B B B C C
$\stackrel{*}{\Rightarrow}$ a a b b b b c c.

We see that together with an A an X is generated. This X is sent through the sequence of B's in the middle. When the X meets the C's on the right-hand side it is changed into a C. While the X travels through the B's, the number of B's is doubled. This is different from an ordinary CSG. In an ordinary CSG it is possible to have a travelling X that does not double the material it passes (with a rule like X B $\rightarrow$ B X). □

Lemma 4.3 *ACSL* $\subset$ *QGCSL*

Suppose we have an acyclic context-sensitive grammar $G = (\langle V, \Sigma, S, P \rangle)$. We construct a QGCSG $G' = (\langle V, \Sigma, S, P' \rangle)$ with weight function g that generates the same language as the ACSG as follows. P' is obtained by removing all labels from P. G and G' generate the same language by definition (the weight function is irrelevant for the generative capacity).

It remains to show that we can construct a function g such that $\forall \alpha \rightarrow \beta \in P' : g(\alpha) < g(\beta)$. First we construct a graph as follows. For every unlabeled symbol in the ACSG there is a vertex in the graph. There is an arc from vertex T to vertex U iff the unary rule $T \rightarrow U$ is in the context-free backbone of the grammar. With $|V|$ we denote the cardinality of a set V. The maximal number of vertices in the graph is $|V|$.

We introduce a counter i that is initialized to $|V| + 1$. Assign the weight i to all vertices that are not connected to any other vertex and remove them. Now search the graph for a vertex without incoming edges. The weight of this symbol is i. Remove the vertex. We increment i by 1 and search for the next vertex without incoming edges. We repeat this untill the graph is empty. The algorithm is guaranteed to stop with an empty graph because the graph is acyclic. Observe that $\forall x \in V : |V| < g(x) \leq 2|V|$.

We have to prove now that $\forall \alpha \rightarrow \beta \in P' : g(\alpha) < g(\beta)$. We consider two cases:

- $|\alpha| = |\beta|$. In this case the context-free backbone of the ACSG rule is unary. The weight of the context symbols is irrelevant because they occur both in α *and* in β. The context-free backbone is of the form $A \rightarrow B$. We know that $g(A) < g(B)$ because A was removed earlier from the graph than B and therefore has a lower weight.

- $|\alpha| < |\beta|$. Again, the weight of the context symbols is irrelevant because they are equal both left and right of the arrow. The context-free backbone is of the form $Z \rightarrow \gamma$, with $|\gamma| > 1$. We know that $g(Z) \leq 2|V|$ and that $g(\gamma) > 2|V|$, hence $g(Z) < g(\gamma)$. □

Lemma 4.4 *QGCSL* $\subset$ *GCSL*

The proof is in Buntrock and Loryś (1992) and is repeated here. For any QGCSG G we construct a GCSG G' and a homomorphism h such that $L(G') = h(L(G))$. Then we introduce a GCSG G'' with $L(G'') = L(G)$. We cite Buntrock and Loryś (1992):

> Let L be generated by a quasi-growing grammar $G = \langle V, \Sigma, S, P\rangle$ with weight function f. Without loss of generality we can assume that S does not appear on the right hand side of any production and that $f(S) = 1$. Let $c \notin V$ and let h be a homomorphism such that $h(a) = ac^{f(a)-1}$ for each $a \in V$. Then $G' = \langle V \cup \{c\}, \Sigma, S, P'\rangle$, where $P' = \{h(\alpha) \to h(\beta) : (\alpha \to \beta) \in P\}$ is a growing context-sensitive grammar and $L(G') = h(L(G))$.

GCSL's are closed under inverse homomorphism (Buntrock and Loryś 1992). Therefore there exists a GCSG G'' that recognizes $L(G)$. □

Theorem 4.5 *ACSL* $\subset$ *GCSL*

Follows immediately from Lemma's 4.3 and 4.4.

It is not known whether GCSL $\subset$ ACSL.

Buntrock and Loryś (1992) show that GCSL $\neq$ CSL. We get the following dependencies:

CFL $\subset$ ACSL $\subseteq$ GCSL $\subset$ CSL

4.2 Complexity of Acyclic CSG's

We formally introduce the following problems:

UNIFORM RECOGNITION FOR ACYCLIC CONTEXT-SENSITIVE GRAMMAR
INSTANCE: An acyclic context-sensitive grammar $G = (V, \Sigma, S, P)$ and a string $w \in \Sigma^*$.
QUESTION: Is w in the language generated by G ?

UNIFORM RECOGNITION FOR GROWING CONTEXT-SENSITIVE GRAMMAR
INSTANCE: A growing context-sensitive grammar $G = (V, \Sigma, S, P)$ and a string $w \in \Sigma^*$.
QUESTION: Is w in the language generated by G ?

RECOGNITION FOR ACYCLIC CONTEXT-SENSITIVE GRAMMAR G
INSTANCE: A string $w \in \Sigma^*$.
QUESTION: Is w in the language generated by G ?

RECOGNITION FOR GROWING CONTEXT-SENSITIVE GRAMMAR G
INSTANCE: A string $w \in \Sigma^*$.
QUESTION: Is w in the language generated by G ?

We can prove two theorems:

Theorem 4.6 *There is a polynomial time reduction from UNIFORM RECOGNITION FOR ACYCLIC CONTEXT-SENSITIVE GRAMMAR to UNIFORM RECOGNITION FOR GROWING CONTEXT-SENSITIVE GRAMMAR*

Theorem 4.7 *for every G there is a G', such that RECOGNITION FOR ACYCLIC CONTEXT-SENSITIVE GRAMMAR G is polynomial time reducible to RECOGNITION FOR GROWING CONTEXT-SENSITIVE GRAMMAR G'*

Proof of both theorems. Consider the proofs of Lemma's 4.3 and 4.4. These proofs show how we can find for every ACSG G a GCSG G' and a homomorphism h such that $L(G') = h(L(G))$. We know that $w \in L(G)$ iff $h(w) \in L(G')$. The reduction is polynomial time. This can be seen as follows. Computing a weight function for an ACSG costs kwadratic time. The weights are linear in $|G|$. Computation of h and G' from the weight function is linear. The total reduction is quadratic. □

Theorem 4.8 *The problem RECOGNITION FOR ACYCLIC CONTEXT-SENSITIVE GRAMMAR G is in P for all G.*

We know from (Dahlhaus and Warmuth 1986) that RECOGNITION FOR GROWING CONTEXT-SENSITIVE GRAMMAR G' is in P for all G'. The theorem follows from this fact and from Theorem 4.7. Aarts (1991) conjectured that the fixed grammar recognition problem was in P. An algorithm was given there without an estimate of the time complexity.

Theorem 4.9 *The problem UNIFORM RECOGNITION FOR ACYCLIC CONTEXT-SENSITIVE GRAMMAR is NP-complete.*

This is proved in Aarts (1992) (but ACSG is defined differently here) and in Aarts (1995).

Theorem 4.10 *The problem UNIFORM RECOGNITION FOR GROWING CONTEXT-SENSITIVE GRAMMAR is NP-complete.*

From Theorems 4.6 and 4.9 it follows that the uniform recognition problem for GCSG is NP-hard. The problem is also in NP because we can guess derivations in GCSG and derivations in GCSG are very short (their length is smaller than the size of the input). □

This result is not new, however. The problem was put forward in an article from Dahlhaus and Warmuth (1986) and has been solved three times (Buntrock and Loryś 1992).

5 Discussion

In the introduction we said that we wanted to introduce a formalism that allows parse trees with crossing branches, which has a simple definition, and which was computationally tractable. The last goal has been achieved only partially. The uniform recognition problem is NP-complete. If the grammar is fixed, then the recognition problem is in P. An aside here: the complexity is $\mathcal{O}(n^k)$ where k depends on the grammar. There is no fixed k.

An interesting question is which of the two problems is more relevant: the uniform recognition problem or the fixed grammar recognition problem. Barton Jr., Berwick and Ristad (1987) argue that the fixed grammar problem is not interesting because it is about languages, and not about grammars. The definitions they use are the following. The universal recognition problem is:

> Given a grammar G (in some grammatical framework) and a string x, is x in the language generated by G?

This problem is contrasted with the fixed language recognition problem:

> Given a string x, is x in some independently specified set of strings L?

If we use the notation of Garey and Johnson (1979), we see that there are in fact not two but three different ways to specify recognition problems. Suppose we have a definition of context-sensitive grammars and the languages they generate. Then we can define the universal or uniform recognition problem as follows.

UNIFORM RECOGNITION FOR CONTEXT-SENSITIVE GRAMMARS
INSTANCE: A context-sensitive grammar $G = (V, \Sigma, S, P)$ and a string $w \in \Sigma^*$.
QUESTION: Is w in the language generated by G ?

There are two forms of the fixed language problem. Suppose we have a grammar ABCGRAM that generates the language $\{a^n b^n c^n | n \geq 1\}$. The following two problems can be defined.

RECOGNITION FOR CONTEXT-SENSITIVE GRAMMAR ABCGRAM
INSTANCE: A string $w \in \Sigma^*$.
QUESTION: Is w in the language generated by G ?

MEMBERSHIP IN $\{a^n b^n c^n | n \geq 1\}$
INSTANCE: A string $w \in \Sigma^*$.
QUESTION: Is w in $\{a^n b^n c^n | n \geq 1\}$?

The complexity of RECOGNITION FOR CONTEXT-SENSITIVE GRAMMAR ABCGRAM is identical to the complexity of MEMBERSHIP IN $\{a^n b^n c^n | n \geq 1\}$ because any algorithm that solves the first problem solves the second too and vice versa. The difference lies in the way the problems are specified. Barton Jr. et al. (1987) only consider the type of problems where languages are specified in some other way than by giving a grammar that generates the language. They argue that this type of problems is not interesting because the grammar has disappeared from the problem, and that it is more interesting to talk about families of languages/grammars than about just one language. However, we see that the grammar does not necessarily disappear in the fixed language recognition problem. If the grammar is part of the specification, we can talk about the fixed language problem in the context of families of grammars. We do this by abstracting over the fixed grammar. E.g., we can try to prove that *for every G*, RECOGNITION FOR CONTEXT-SENSITIVE GRAMMAR G is PSPACE-complete. Abstracting from the grammar is different from putting the grammar as input on the tape of a Turing machine, as is done in the uniform recognition problem.

If we want to answer the question which of the two problems is more relevant for us we first have to answer the question: "Relevant for what?". Complexity analysis of grammatical formalisms serves at least two purposes. First, it helps us to design algorithms that can deal with natural language efficiently. Secondly, it can judge grammar formalisms on their psychological relevance.

It is hard to say whether the uniform problem or the fixed grammar problem is more relevant for efficient sentence processing. An argument in favor of the uniform recognition problem is the following. Usually, grammars are very big, bigger than the sentence length. They have a strong influence on the runtime of an algorithm in practice. Therefore we can not simply forget the grammar size as is done in the fixed language recognition problem. On the other hand, we can argue in favor of the fixed language problem as follows. In many practical applications the task an algorithm has to fullfill is the processing of sentences of some given language (e.g. a spelling checker for English). Practical algorithms have to decide over and over whether strings are in the language generated by some fixed grammar. Only in the development phase of the application the grammar changes. This argues for the relevance of the fixed grammar problem. It is not clear which of the two problems is more relevant in the design of efficient algorithms.

The psychological relevance is a very hard subject (shortly discussed in Barton Jr. et al. (1987, pp. 29,74)). Humans can process natural language utterances fast (linear in the length of the sentence). It seems that we have to compare this with the complexity of the fixed language problem. We follow the same line of reasoning more or less that was used in the previous paragraph. The problem that is solved by humans is the problem of understanding sentences of one particular language (or two, or three). People can not change their "built-in grammar" at will. Barton Jr. et al. (1987)) remark that natural languages must be acquired, and that in language acquisition the grammar changes over time. This change is a very slow change, however, compared to the time needed for sentence processing.

Our conclusion is that both the uniform and the fixed language problem are interesting. Which of the two is more important depends on what perspective one takes.

References

Aarts, Erik: 1991, Recognition for Acyclic Context-Sensitive Grammars is probably Polynomial for Fixed Grammars, *ITK Research Memo no. 8*, Tilburg University.

Aarts, Erik: 1992, Uniform Recognition for Acyclic Context-Sensitive Grammars is NP-complete, *Proceedings of COLING '92*, Nantes, pp. 1157–1161.

Aarts, Erik: 1995, *Computational Complexity of Grammar Formalisms (?)*, PhD thesis, Research Institute for Language and Speech, University of Utrecht. to appear.

Abramson, H. and Dahl, V.: 1989, *Logic grammars*, Springer, New York–Heidelberg–Berlin.

Barton Jr., G. Edward, Berwick, Robert C. and Ristad, Eric Sven: 1987, *Computational Complexity and Natural Language*, MIT Press, Cambridge, MA.

Bunt, Harry: 1988, DPSG and its use in sentence generation from meaning representations, *in* M. Zock and G. Sabah (eds), *Advances in Natural Language Generation*, Pinter Publishers, London.

Buntrock, Gerhard: 1993, Growing Context-sensitive Languages and Automata, *Report no. 69*, University of Würzburg, Computer Science.

Buntrock, Gerhard and Loryś, Krzysztof: 1992, On growing context-sensitive languages, *Proceedings of 19^{th} International Colloquium on Automata, Languages and Programming*, Vol. 623 of *Lecture Notes in Computer Science*, Springer, pp. 77–88.

Dahlhaus, Elias and Warmuth, Manfred K.: 1986, Membership for Growing Context-Sensitive Grammars Is Polynomial, *Journal of Computer and System Sciences* **33**, 456–472.

Garey, Michael R. and Johnson, David S.: 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, CA.

Johnson, Mark: 1985, Parsing with discontinuous constituents, *Proceedings of the 23^{rd} Ann. Meeting of the ACL*, pp. 127–132.

Johnson, Mark: 1988, *Attribute-Value Logic and the Theory of Grammar*, Vol. 16 of *CSLI Lecture Notes*, CSLI, Stanford.

Kaplan, R. and Bresnan, J.: 1982, Lexical-Functional Grammar: a Formal System for Grammatical Representation, *in* J. Bresnan (ed.), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, MA, pp. 173–281.

Pereira, Fernando: 1981, Extraposition Grammars, *Computational Linguistics* **7**(4), 243–256.

Salomaa, Arto: 1973, *Formal languages*, Academic Press.