1

Learning Shallow Context-Free Languages under Simple Distributions

PIETER W. ADRIAANS

In this paper I present the EMILE 3.0 algorithm[1]. It can learn shallow context-free grammars efficiently. It does so under circumstances that, from a perspective of complexity, come resonably close to the conditions under which human beings learn a language. A language is shallow in its descriptive length if all the relevant constructions we need to know to learn it have a complexity that is logarithmic in the descriptive length of the grammar as a whole. I claim that natural languages are shallow and that shallowness in itself is an interesting general constraint in the context of formal learning theory. I also introduce the concept of separability. A language is separable if the validity of rules can be tested by means of memberbership queries using simple substitutions. Although I do not believe that human languages are strictly separable I do think it is an important aspect of efficient human communication. I illustrate the structure of the algorithm by means of an extensive example, and I indicate how a careful analysis of the bias involved in natural language learning may lead to versions of EMILE 3.0 that can be implemented in real-life dialogue systems. I claim that the EMILE approach could serve as a valuable metamodel for evaluating clustering approaches to language learning. The results of EMILE are mainly theoretical. They

[1]The original acronym stands for Entity Modeling Intelligent Learning Engine. It refers to earlier versions of EMILE that also had semantic capacities. The name EMILE is also motivated by the book on education by J.-J. Rousseau. In order to distinguish several versions of EMILE I started recently to give them version numbers. Versions 1 and 2 are discussed in Adriaans, 1992a.

could potentially also be used to develop more efficient variants of other approaches to language learning.

1 Introduction

Suppose that you are a systems analist and you have to design a new computer system for a company. The CIO tells you: 'We make wuzzles. Our wuzzles are shipped all over the world'. You probably do not know what wuzzles are (I don't), but from those two sentences you might get at least a rough idea. Firstly the word 'wuzzle' is a noun. Wuzzles are probably physical objects, not too big, they can be moved, etc. Based on this example we can make several observations. Exchange of information can be seen as language learning. We learned a partial definition of the word 'wuzzle'. The definition contains syntactic as well as semantic information. Apparently the partial sentences 'We make ...' and 'Our ...are shipped all over the world' are strong enough to determine the meaning of words we substitute in the gaps. We interpret the unknown in terms of the known.

Currently researchers from different backgrounds frequently rediscover the fact that by applying forms of cluster analysis to text one can identify semantic and syntactic clusters. Various ML techniques can be used in this context: genetic algorithms Lankhorst, 1994, neural networks Finch & Chater, 1992, Honkela *et al.*, 1995, n-gram search, Brown, 1991. What is lacking however is a unifying framework to evaluate these ad hoc results.

I claim that Adriaans, 1992a does provide such a framework and that the EMILE approach could serve as a valuable metamodel to evaluate clustering approaches to language learning. I want to stress that, although the work in this paper is essentially theoretical, my central research theme is practical: how can it be that our formal analysis of human language acquisition is so complex while the process itself in reality seems to be so efficient? With this question in mind I set out to analyse the bias involved in human communication. This paper contains some of the results. It combines theories like PAC-learning, categorial grammars and Kolmogorov complexity. It presents a conceptual framework to language learning that seems to be interesting enough to develop more deeply, although a lot of questions still remain open.

Work that is relevant for our approach is of course Gold's results on identification in the limit (Gold, 1967), with the application of Kanazawa to categorial grammars (Kanazawa, 1994), Valiants ideas on PAC-learning (Valiant, 1984), Angluin's work on membership- and equivalence queries (Angluin, 1992), the work of Buszkowski and Penn on learning categorial grammars from bracketed trees (Buszkowski, 1989). Ideas on learning from characteristic samples (Sakakibara, 1990, Sakamoto,1995). The ideas of Li and Vitányi on learning under simple distributions provide us with a tool that is fundamental for the proof of the results presented in this paper (Li & Vitányi, 1997, Li & Vitányi, 1991). Also important is the proof of Pentus that context-free and categorial grammars are equivalent (Pentus, 1993). Context-free grammars consist of sets of rules. Categorial grammars consist of a dictionary that assigns types to words. Pentus' result allows us to travel from one representation to another. One advantage is that the *occurrence* of a word in a specific syntactic role in a sentence in a sufficiently rich sample of a language suffices to deduce its syntactic type.

There are several other constraints to be found in the literature that make context-free grammars efficiently learnable (e.g. k-locality, Abe, 1988). A new approach to language learning are the socalled stochastic context-free grammars (Charniak, 1993). There are some interesting hardness results. Gold, 1978, showed that finding a DFA with a minimum number of states consistent with a given finite sample of positive and negative examples is NP-hard[2]. Pitt and Warmuth, 1989, show that merely finding an approximate solution is intractable (assuming $P \neq NP$).

The learning context presupposed by EMILE 3.0 is one in which we have a dialogue between a teacher and a pupil. The teacher generates grammatically correct sentences (positive examples). The pupil can ask yes/no questions concerning the grammatical validity of new sentences (membership queries: positive as well as negative examples). We say that a language can be learned efficiently when the amount of examples, yes/no questions and computing and storage power the pupil needs to learn a grammar are polynomially related to the length of a description of the grammar. Informally, the circumstances under which EMILE 3.0 learns a context-free language efficiently are:

- Each grammatical construction in the language can be illustrated[3] with a relatively small example sentence. (The length of description of the sentence must be logarithmic in the length of the description of the grammar).
- The teacher gives examples randomly, but in such a way that simpler sentences have a much higher probability to occur than complex sentences.

[2] For an analysis of this result see Pitt, 1989.

[3] The word 'illustrated' is used in a technical sense to be explained later. Essentially the construction is used in the generation of the sentence.

- The learning algorithm must be able to check the grammatical validity of example sentences by means of membership queries.

I argue that these constraints are quite reasonable for natural dialogues. The first constraint tells us that at first we only have to analyse the simple sentences that the teacher gives us. We can postpone the analysis of longer sentences, because we know for certain that they are built using grammatical rules that already occur in simple sentences.

The second constraint tells us that we will get all the necessary examples in time. We only need a polynomial number of example sentences in order to have the right building blocks to start the learning process.

The third constraint states that we need both positive and negative examples. Although there is some discussion, I believe that children get negative information about syntactic constructions in various implicit ways[4]. It is possible to develop efficient variants of Emile that learn from positive data alone. Given the sytematic behavior of the teacher we can deduce the ungrammaticality of constructions from the fact that they do not occur in the sample.

In practice there is a large number of potential mechanisms to speed up the learning process considerably. Note that EMILE 3.0 employs a rather strong definition of grammar induction. The original grammar and the learned grammar must be weakly equivalent: i.e. recognise exactly the same set of sentences as grammatical. For natural languages this is probably too rigid. It means that somebody only knows English when he knows the whole grammar by heart. If we loosen our definition to learning subgrammars the efficiency of the learning process increases considerably.

Here I will present the EMILE 3.0 algorithm, give an overview of the proof and discuss potential applications in real life systems, including the possibility to develop psychologically plausible models of language acquisition. In particular the perspective of a 'stepping stone' approach to language acquisition looks promising: a small core grammar is learned initially using computationally intensive techniques, using this core grammar the rest of the language (dictionary) can be learned in linear time.

2 The EMILE 3.0 Algorithm: A Simple Example

Before I discuss technical details, I will illustrate the basic algorithm by means of a simple example based on a context-free grammar. I introduce the slash notation used by categorial grammar. The operator $\backslash$ enables us to transform the rule $S \to ab$ into $a\backslash S \to b$ and vice versa. The

[4] I discuss this issue briefly in the evaluation.

operator / enables us to transform the rule $S \rightarrow ab$ into $S/b \rightarrow a$ and vice versa. These operators help us to isolate and manipulate parts of rules that would otherwise be inaccessible.

The algorithm consists of the following stages:

1. Selection of a sample of sentences.
2. Generation of the first order explosion.
3. Cross-check of expressions in different contexts.
4. Clustering of expressions and contexts into types.
5. Induction of general types from specific cases.
6. Rewriting of rules.

The detailed description of the algorithm will be illustrated by the following example-input:

John loves Mary
Mary walks

I will suppose this sample to be characteristic, i.e. it contains an example of every possible construction of the grammar in its simplest form. In subsequent sections I will study the exact conditions under which we can draw a characteristic sample of a language in polynomial time.

We start by forming the *first order explosion* of the sentences in the example. This means that for each sentence we examine the number of ways in which this sentence can be split up in subexpressions and contexts. Another way of looking at this, is to consider all ways in which we can place two brackets '(' and ')', such that the sentence is split up in three parts α, β and γ. β is the subexpression and α and γ form the context. α and γ may be empty. For each division $S \rightarrow \alpha\beta\gamma$ we move the context to the other side of the functor: $\alpha\backslash S/\gamma \rightarrow \beta$. This results in the following first order explosion for the example:

$$
\begin{array}{c}
S/\textsf{loves Mary} \rightarrow \textsf{John} \\
S/\textsf{Mary} \rightarrow \textsf{John loves} \\
S \rightarrow \textsf{John loves Mary} \\
\textsf{John}\backslash S/\textsf{Mary} \rightarrow \textsf{loves} \\
\textsf{John}\backslash S \rightarrow \textsf{loves Mary} \\
\textsf{John loves}\backslash S \rightarrow \textsf{Mary} \\
S/\textsf{walks} \rightarrow \textsf{Mary} \\
S \rightarrow \textsf{Mary walks} \\
\textsf{Mary}\backslash S \rightarrow \textsf{walks}
\end{array}
$$

Given a set of possible subexpressions and contexts we can now verify substitutions of all expressions in each context, using an *oracle*. This

	John	John loves	John loves Mary	loves	loves Mary	Mary	Mary walks	walks
S/loves Mary	s	·	·	·	·	s	·	·
S/Mary	·	s	·	·	·	·	·	·
S	·	·	s	·	·	·	s	·
John\\S/Mary	·	·	·	s	·	·	·	·
John\\S	·	·	·	·	s	·	·	s
John loves\\S	s	·	·	·	·	s	·	·
S/walks	s	·	·	·	·	s	·	·
Mary\\S	·	·	·	·	s	·	·	s

FIGURE 1 Substitution-matrix for the enriched first order explosion.

means that a candidate sentence is presented to a speaker of the language, who in turn specifies whether or not this is a valid sentence. Figure 1 shows a matrix of expressions and contexts, where each entry indicates whether the sentence is part of the language (s), or not part of the language (·). Observe that some of the entries in the substitution-matrix already occurred in the sample, and that several combinations of expressions and contexts may form the same sentence. The entries that contain an s form the *enriched first order explosion*:

S/loves Mary $\rightarrow$ John
S/Mary $\rightarrow$ John loves
S $\rightarrow$ John loves Mary
John\\S/Mary $\rightarrow$ loves
John\\S $\rightarrow$ loves Mary
John loves\\S $\rightarrow$ Mary
S/walks $\rightarrow$ Mary
S $\rightarrow$ Mary walks
Mary\\S $\rightarrow$ walks
S/loves Mary $\rightarrow$ Mary
John\\S $\rightarrow$ walks
John loves\\S $\rightarrow$ John
S/walks $\rightarrow$ John
Mary\\S $\rightarrow$ loves Mary

The next step is to cluster context rules into types. Expressions that can be substituted into the same contexts belong to the same type. For example

S/loves Mary $\rightarrow$ John
S/loves Mary $\rightarrow$ Mary

	John	Mary	John loves	John loves Mary	Mary walks	loves	loves Mary	walks
S/loves Mary	s	s	·	·	·	·	·	·
John loves$\backslash S$	s	s	·	·	·	·	·	·
S/walks	s	s	·	·	·	·	·	·
S/Mary	·	·	s	·	·	·	·	·
S	·	·	·	s	s	·	·	·
John$\backslash S$/Mary	·	·	·	·	·	s	·	·
John$\backslash S$	·	·	·	·	·	·	s	s
Mary$\backslash S$	·	·	·	·	·	·	s	s

FIGURE 2 Re-ordered substitution-matrix for the enriched first order explosion.

$$\begin{array}{c} \textsf{John loves}\backslash S \rightarrow \textsf{John} \\ \textsf{John loves}\backslash S \rightarrow \textsf{Mary} \\ S/\textsf{walks} \rightarrow \textsf{John} \\ S/\textsf{walks} \rightarrow \textsf{Mary} \end{array}$$

implies that John and Mary can be substituted in the same context, so:

$$A \rightarrow \textsf{John} \mid \textsf{Mary}$$

At the same time the contexts S/loves Mary, John loves$\backslash S$ and S/walks belong to the same type:

$$\begin{array}{c} S/\textsf{loves Mary} \rightarrow A \\ \textsf{John loves}\backslash S \rightarrow A \\ S/\textsf{walks} \rightarrow A \end{array}$$

Thus, to generate a set of types from the enriched first order explosion, we can cluster expressions on the basis of equivalent contexts. For the matrix in figure 1 this corresponds to re-ordering the rows and columns such that rectangles of s's can be found along the diagonal in the matrix[5]. Each rectangle corresponds to one type. The resulting matrix for our example is shown in figure 2. This corresponds to the following list of rules:

$$\begin{array}{c} S/\textsf{loves Mary} \rightarrow A \\ \textsf{John loves}\backslash S \rightarrow A \\ S/\textsf{walks} \rightarrow A \\ S/\textsf{Mary} \rightarrow B \end{array}$$

[5]Ambiguous grammars may cause s's to appear outside these rectangles. We need more sophisticated clustering techniques for ambiguous languages, but that does not affect the complexity in a fundamental way.

$$S \to C$$
$$\mathsf{John}\backslash S/\mathsf{Mary} \to D$$
$$\mathsf{John}\backslash S \to E$$
$$\mathsf{Mary}\backslash S \to E$$
$$A \to \mathsf{John} \mid \mathsf{Mary}$$
$$B \to \mathsf{John\ loves}$$
$$C \to \mathsf{John\ loves\ Mary} \mid \mathsf{Mary\ walks}$$
$$D \to \mathsf{loves}$$
$$E \to \mathsf{loves\ Mary} \mid \mathsf{walks}$$

We now have basic lexicon-rules such as $A \to \mathsf{John}$, and more complex rules that are expressed in the lexicon, such as $S/\mathsf{loves\ Mary} \to A$ and $B \to \mathsf{John\ loves}$. Our next step is to express the complex rules in terms of the basic lexical types. By making this induction-step we generalise from specific types to more general types.

Each word that occurs in the complex rules is substituted by the most specific type this word belongs to. For example $S/\mathsf{loves\ Mary} \to A$ becomes $S/DA \to A$. Finally we flip the contexts on the left side of rules over to the right side, so we get $S \to ADA$. This results in the final set of rules for the example:

$$S \to ADA$$
$$S \to BA$$
$$S \to C$$
$$S \to AE$$
$$A \to \mathsf{John} \mid \mathsf{Mary}$$
$$B \to AD$$
$$C \to ADA|AE$$
$$D \to \mathsf{loves}$$
$$E \to DA| \mathsf{\ walks}$$

By making the appropriate substitutions, we can reduce the grammar to:

$$S \to BA$$
$$S \to AE$$
$$A \to \mathsf{John} \mid \mathsf{Mary}$$
$$B \to AD$$
$$D \to \mathsf{loves}$$
$$E \to DA| \mathsf{\ walks}$$

Now although this example might at first sight seem to be somewhat cumbersome, I will show in the subsequent sections that the EMILE 3.0 algorithm learns shallow grammars efficiently under simple distribution.

2.1 PAC-learning

In his 1984 paper Valiant introduced a completely new learning concept that since then has been named "probably approximately correct" (PAC) learning (Valiant, 1984). The basic idea of PAC-learning is that we can minimize the chance of learning something that is wrong without being completely sure that we are right. Let $\Sigma = 1, 0$ be the Boolean alphabet and let Σ^* be the set of all strings of finite length on Σ. A *concept* f is any subset of Σ^*. A formal definition of PAC-learning is (Natarajan, 1991):

Definition 1.1 *An algorithm A is a* probably approximately correct *(PAC) learning algorithm for a class of concepts F if*

- *A takes as input $\epsilon \in (0, 1]$, $\delta \in (0, 1]$, and $n \in N$ where ϵ and δ are error- and confidence parameters, respectively, and n is the length parameter.*
- *A may call EXAMPLE, which returns examples for some $f \in F$. The examples are chosen randomly according to an arbitrary and unknown probability distribution P on P on $\Sigma^{\leq n}$[6].*
- *For all concepts $f \in F$ and all probability distributions P on $\Sigma^{\leq n}$, A outputs a concept $g \in F$, such that with probability at least $(1 - \delta)$, $P(f \Delta g) \leq \epsilon$)*

Here N is the set of natural numbers and $(0, 1]$ is the half open interval from 0 to 1. Σ^n is the set of all strings of length n in Σ^*, and $\Sigma^{\leq n}$ is the set of all strings of length at most n. For any two sets f and g, $f \Delta g$ denotes the symmetric difference $(f - g) \cup (g - f)$.

Valiant applied his theory to Boolean concept learning. Subsequent investigations showed that this technique actually has a much broader field of application (Li & Vitányi, 1991). The pure PAC-paradigm is too general for the grammar induction problem. Firstly we need membership queries to test our hypotheses during the learning process. Secondly, we will restrict ourselves to samples, not taken under any distribution, but under a special class of simple distributions. In the following I will deal with two complexity issues: sample complexity and time complexity. Firstly I will show that it is possible to take with high probability a chracteristic sample of a shallow grammar under simple distributions in polynomial time, then I will show that this sample can be transformed to a grammar in polynomial time.

[6] The restriction to examples not longer than n is not essential (Natarajan, 1991, pg. 34-36).

3 Complexity bounds

In the following I will make extensive use of the theory of Kolmogorov complexity. This theory allows us to formulate an a priori complexity measure for objects. In a nutshell the Kolmogorov complexity of an object is the length of the shortest program that generates this object on a universal Turing machine. We include a short presentation of the relevant ideas behind Kolmogorov complexity in this paper to make it self contained.

Suppose that an abstract speaker has given us a number of examples and we have a learning algorithm that distinguishes certain regularities in this set; i.e., the algorithm is beginning to learn something. In such a case we want to have a measure of the probability that the observed regularities are not merely produced by chance and are caused by a predefined structure of the grammar. This enables us to fix threshold values to activate interactive learning routines. The algorithm will only try to learn something if it is worth doing so. A related question is the a priori estimation of the size of the sample space. How many examples do we have to ask before we have enough material to start our analysis? In this paragraph I will describe a surprisingly consistent and perspicuous solution for this problem arising from research in complexity theory. I think that this approach based on the work of Li and Vitányi can be of great value for researchers in the field of language learning.

Using results of Solomonoff and Levin we can define an a priori probability of a finite binary string[7]. Based on this analysis we can define a criterion for efficient learnability. Let S be an arbitrary discrete sample space. We call a function $w : S \rightarrow [0,1]$ a *semi-probability density* if:

$$\sum_{x \in S} w(x) \leq 1$$

We call it a probability density over the space S if this relation is an equality.

We consider real-valued functions. A function $f : N \rightarrow R$ is enumerable iff the set $\{(x,y) | y \leq f(x) \text{ and } x, y \in Q\}$, is recursively enumerable. Here Q represents the rational numbers, R represents real numbers, N represents natural numbers. This is enumerability from below. If $f(x)$ is also enumerable from above, then we call it recursive. (Both are an extension of the normal case in recursive function theory). Enumerable is also called 'semicomputable'. A recursively enumerable semi-measure μ on N is called universal if it multiplicatively dominates every other enu-

[7] Our discussion of simple distributions is very elementary. For details see Li & Vitányi, 1997, Li & Vitányi, 1991.

merable semi-measure μ' i.e. $\mu(x) \geq c\mu'(x)$ for a fixed positive constant c independent of x. We state without proof:

Theorem 1.1 (Levin) *There is a universal enumerable semi-measure.*

We fix a universal semi-measure $\mathbf{m}$(x). The semi-measure $\mathbf{m}$(x) converges to 0 slower than any positive recursive function which converges to 0. Of course $\mathbf{m}$(x) itself is not recursive.

Consider only Turing machines such that the set of halting programs is in prefix-form. The descriptive complexity of a string x relative to a Turing machine T and a binary string y is defined as the shortest program that gives output x on input y:

$$K_T(x|y) = \min\{|p| : p \in \{0,1\}^*, T(p,y) = x\}$$

There is a universal Turing machine U, such that for each Turing machine T there is a constant c_T, such that for all x and y, we have $K_U(x|y) \leq K_T(x|y) + c_T$. This definition is invariant up to a constant with respect to different universal Turing machines. Hence we fix a reference universal Turing machine U, and drop the subscript U by setting $K(x|y) = K_U(x|y)$. We define:

Definition 1.2 *The Kolmogorov complexity of a binary string x is $K(x) = K(x|\epsilon)$. That is the shortest program that produces x on an empty input string. that is the shortest program that produces x on an empty input string.*

Now let this $K(x)$ be the prefix variant of Kolmogorov complexity first proposed by Levin (See Levin, 1974, Gács, 1974).

Definition 1.3 (Solomonoff, Levin) *The universal a priori probability $P_U(x)$ of a binary string x is*

$$P_U(x) = \sum_{U(p)=x} 2^{-|p|}$$

This is the sum of the probabilities of all the programs that generate x on a universal Turing machine on an empty input string. One might be tempted to think that this summation easily can be greater than one. This is not the case by Kraft's inequality. Programs that produce a certain string x are extremely rare. We now give without proof a theorem that relates all these concepts with each other:

Theorem 1.2 (Levin)

$$-\log \mathbf{m}(x) = -\log P_U(x) + O(1) = K(x) + O(1)$$

The probability that an arbitrary speaker selects a certain binary string decreases exponentially with the complexity of this string. The importance for a theory of learning is apparent from the following completeness result of Li and Vitányi:

Theorem 1.3 *A concept class C is learnable under $\mathbf{m}(x)$ iff C is also learnable under any arbitrary simple distribution $P(x)$ provided the samples are taken according to $\mathbf{m}(x)$.*

A distribution is *simple* if it is dominated by a recursively enumerable distribution (and hence by $\mathbf{m}$). Since the distribution $\mathbf{m}$ is not constructive we can in reality never sample according to $\mathbf{m}$. However, learnability according to $\mathbf{m}$ is important because of the fact that the linguistic behaviour of a cooperative teacher or a group of methodical speakers will always be dominated by $\mathbf{m}$. If a speaker has a limited amount of time to generate examples he will prefer simple examples to complex ones. We cannot simulate $\mathbf{m}$ but real world situations are dominated by $\mathbf{m}$. In practice a proof of learnability according to $\mathbf{m}$ is a strong indication for learnability in natural situations[8].

Definition 1.4 *Two grammars G and $G\prime$ are weakly equivalent iff $L(G) = L(G\prime)$.*

Definition 1.5 *A* characteristic sample *S_c of a language $L(G)$ generated by a grammar G is a finite set of positive and negative example sentences such that there is a recursive function f that constructs a grammar $G\prime$ from S_c that is weakly equivalent to G.*

Definition 1.6 *A* positive characteristic sample *contains only positive examples.*

Definition 1.7 *Let n be any parameter associated with a grammar G. A language is* shallow in *n if it has a characteristic sample S_c satisfying the following conditions:*

- *$|S_c|$ is polynomial in n.*
- *For each $s \in S_c$ we have $K(s) \leq c \log n$, where $K(s)$ is the prefix Kolmogorov complexity of s.*
- *A grammar $G\prime$ weakly equivalent to G can be constructed from from S_c in time polynomial to n using membership queries.*

The parameter n in this definition can relate to any intrinsic property of the grammar, i.e. its descriptive length, the number of rules, the number of categories or the number of states of an equivalent Turing machine. The most intuitive interpretaion of shallowness is of course related to the descriptive length of grammars and we will use the notion

[8]In the following I prove that we can learn an exact grammar from a sample that is with high probability representative, i.e. we do 'exact' PAC-learning. A real PAC version of the proof can probably be developed but is currently not available.

of shallowness in this sense in the rest of this paper. Lemma 1 states that we can take a representative sample of a shallow language under $\mathbf{m}$ in polynomial time. This was proven in Li & Vitányi, 1997 and we include the argument to make our treatement self contained.

We can use the length of a sentence as a measure for its complexity. For context-free languages simplest means shortest in this context and the shallowness condition could be restated as $|s| \leq c\log(K(G))$.

Lemma 1 *Let $L(G)$ be a language generated by a grammar G that is shallow in a parameter n. A characteristic sample of $L(G)$ can be drawn with high probability under $\mathbf{m}$ in time polynomial in n.*

Proof. Sample $n^{c'}$ sentences where we choose c' large enough to satisfy the following argument.

Claim With probability greater than $1 - n^c/e^n$ the sample will be characteristic for $L(G)$.

Let s be a sentence with a prefix Kolmogorov complexity of at most $c\log(n)$. Therefore

$$\mathbf{m}(s) \geq 2^{(-c\log(n))} \geq n^{(-c-1)}$$

for large enough n. This is the probability that s will be sampled. Let E be the event that s does not occur in $n^{c'}$ examples. Then:

$$\Pr(E) \leq (1 - n^{(-c-1)})^{n^{c'}} \leq e^{-n}$$

for c' large enough.There are only n^c possible sentences s such that $K(s) \leq c\log(n)$ hence the probability that all those sentences are sampled is at least $1 - n^c/e^n$. □

Note that this means that a shallow language can be PAC-learned if there exists a deterministic polynomial time algorithm to analyse the characteristic sample into a grammar. In the following I will show that such an algorithm exists for shallow context-free languages.

It seems reasonable to suppose that, for natural languages with their enormous lexica, we have for every grammatical rule an example with a length logarithmic in the length of the lexicon. A rough estimate shows that we would need an interactive analysis of a sample under $\mathbf{m}$ of about 5 million sentences to learn a language with 50.000 words (Adriaans, 1992a). For natural speakers this is not an unsurmountable barrier. It is interesting to note that most programming languages with their simple grammars with deep nesting of syntactic constructions are not shallow and therefore probably not learnable under $\mathbf{m}$. In other words: to learn a programming language we need an extremely cooperative teacher.

4 A syntactic learning algorithm

4.0.1 Introduction

Buszkowski, 1989 described an algorithm to deduce a categorial grammar from linguistic data by unification. Their method presupposes that we know the syntactic structure of the expressions. For real life learning situations this is an unrealistic situation. Here I present a class of clustering learning algorithms to overcome this drawback.

In Adriaans, 1992a I gave a proof of the learnability of categorial grammars. This proof was complicated due to the fact that at that time the equivalence of context-free and categorial grammars was still an open question. Here I will give a much simpler version of the proof. I will prove efficient learnability of shallow grammars with no lexical ambiguity and a special quality called separability. I will illustrate how these two constraints can be removed in order to get a learnability result for shallow grammars in general. In the proof I will make use of the theory of categorial grammars to analyse the internal structure of groups of expressions.

4.0.2 Context-free grammars

We will stick to the following notational conventions:

- Roman lowercase letters denote individual words of a language, x, y, z. The expressions $\overline{x}$, $\overline{y}$, $\overline{z}$ will denote sequences of words.
- Greek lowercase letters will indicate labels in rewrite rules: α, β, γ. The expressions $\overline{\alpha}$, $\overline{\beta}$, $\overline{\gamma}$ will denote sequences of labels and words.
- Roman capitals will represent types, i.e. sets of expressions that can be substituted for each other salva beneformatione: A, B, C.
- The symbols m, n, N, S, σ and P are reserved symbols.

Suppose we want to learn a context-free grammar $G =< N, \Sigma, P, \sigma >$ in Chomsky normal form. Any context-free grammar can be transformed in to Chomsky normal form[9] so this does not restrict us in any way. G is defined in the following way:

- Σ is a finite set of *terminals*
- N is a finite set of *non-terminals*, such that $N \cap \Sigma = \emptyset$
- $\sigma \in N$ is the *start symbol*
- P is a finite set of *production rules* of the form $\alpha \rightarrow \beta \bullet \gamma$ and $\delta \rightarrow t$ ($\alpha, \beta, \gamma, \delta \in N$ and $t \in \Sigma$).

[9] For a proof see Hopcroft,1979 pg. 92-93.

It will prove to be useful to treat a grammar as a calculus and to introduce a formal notion of derivation in a grammar G:

$$\frac{\alpha \rightarrow \overline{\beta}}{\alpha \overset{*}{\Rightarrow}_G \overline{\beta}}$$

$$\frac{\alpha \overset{*}{\Rightarrow}_G \overline{\beta} \bullet \gamma \bullet \overline{\epsilon} \quad \gamma \overset{*}{\Rightarrow}_G \overline{\delta}}{\alpha \overset{*}{\Rightarrow}_G \overline{\beta} \bullet \overline{\delta} \bullet \overline{\epsilon}}$$

$\alpha \in N, \overline{\beta}, \overline{\gamma}, \overline{\delta}, \overline{\epsilon} \in (\Sigma \cup N)^*$. $\overline{\beta}, \overline{\epsilon}$ may be empty.

4.0.3 Semantics for context-free grammars

This calculus has a set theoretical interpretation. It will prove useful to have the following definitions. The subsets of a free monoid are subject to the following operations:

$$\begin{aligned} A \bullet B &= \{x.y \in M \mid x \in A \ \& \ y \in B\} && (1) \\ C/B &= \{x \in M \mid \forall y_{\in B}(x.y \in C)\} && (2) \\ A \backslash C &= \{y \in M \mid \forall x_{\in A}(x.y \in C)\} && (3) \end{aligned}$$

A, B and C are subsets of M. We read: A *times* B, C *over* B and A *under* C. One can easily verify that the following statements hold for all A, B and C in M:

$$\begin{aligned} A \bullet B \subseteq C \quad &\text{if and only if} \quad A \subseteq C/B && (4) \\ A \bullet B \subseteq C \quad &\text{if and only if} \quad B \subseteq A \backslash C && (5) \end{aligned}$$

We have

$$(A \bullet B) \bullet C = A \bullet (B \bullet C) \qquad (6)$$

We can now describe a semantics for our syntactic calculus.

Definition 1.8 *Let* $[\alpha]^G$ *be the semantic value of the expression* α *with respect to grammar G:*

- $[\alpha \bullet \beta]^G = [\alpha]^G \bullet [\beta]^G$.
- For each $t \in \Sigma$ we have $[t]^G = \{t\}$.
- For each $\delta \in N$ we have $[\delta]^G = \{\overline{x} | \delta \overset{*}{\Rightarrow}_G \overline{x}, \overline{x} \in \Sigma^*\}$.
- $[\sigma]^G = \{\overline{x} | \sigma \overset{*}{\Rightarrow}_G \overline{x}, \overline{x} \in \Sigma^*\} = L(G)$ is the set of sentences.
- For each $\alpha \overset{*}{\Rightarrow}_G \beta \bullet \gamma$ in P we have $[\alpha \overset{*}{\Rightarrow}_G \beta \bullet \gamma]^G = [\beta]^G \bullet [\gamma]^G \subseteq [\alpha]^G$
- For each $\delta \overset{*}{\Rightarrow}_G t$ in P we have $[\delta \overset{*}{\Rightarrow}_G t]^G = \{t\} \subseteq [\delta]^G$

The following lemma shows that our rule of derivation presented in 4.0.2 is actually sound:

Lemma 2 *If* $[\beta]^G \bullet [\gamma]^G \bullet [\epsilon]^G \subseteq [\alpha]^G$ *and* $[\delta]^G \subseteq [\gamma]^G$ *then* $[\beta]^G \bullet [\delta]^G \bullet [\epsilon]^G \subseteq [\alpha]^G$

In this way I have associated the notion of derivability with the notion of set inclusion of sets of expressions. Every valid derivation in a context-free grammar corresponds with a set inclusion of expressions. The converse is not true, not every set inclusion corresponds with a valid derivation.

4.0.4 Additional constraints and definitions

We can now discuss the additional separability constraint, used in our algorithm. Basically separability allows us to do simple tests to check whether an expression belongs to a type or not. A language is separable if there exist simple paradigms to check constructions.

Definition 1.9 *We say that a grammar has* context separability *if for each each non-terminal* $\eta \in N$ *there is at least one sentence* $\overline{u} \bullet v \bullet \overline{w}$ *(or* $\overline{u} \bullet v$ *or* $v \bullet \overline{w}$*)*[10] *in* $L(G)$ *such that only expressions of type* η *can be substituted salva beneformatione. We will call* $u\backslash\sigma/w$ *a* characeristic context *of* η*.*

Definition 1.10 *We say that a grammar has* expression separability *if for each non-terminal* $\eta \in N$ *there is at least one expression* v *that belongs unambiguously to type* η*, i.e., it can not be substituted in a context that is characteristic for an expression other than* η*. We will call* v *a* characteristic expression *of* η*.*

Definition 1.11 *We say that a grammar* G *is* lexically unambiguous *if for every word* $w \in \Sigma$ *there is exactly one arrow* $\delta \rightarrow w$ *in the grammar.*

Definition 1.12 *A* positive characteristic sample *of a context-free language in Chomski normal form* $L(G)$ *will be a finite set* $S_c \subseteq L(G)$ *such that:*

- *for each rule* $\alpha \rightarrow \beta \bullet \gamma$ *(*$\alpha, \beta, \gamma \in N$*) in* P *there is a sentence* $s \in S_c$ *such that* $\alpha \rightarrow \beta \bullet \gamma$ *is used in the construction of* s *and there are no sentences shorter (i.e. less complex) than* s *involving rule* $\alpha \rightarrow \beta \bullet \gamma$*.*
- *for each rule* $\delta \rightarrow t$ *(*$\delta \in N, t \in \Sigma$*) in* P *there is a sentence* $s \in S_c$ *such that* $\delta \rightarrow t$ *is used in the construction of* s *and there are no sentences shorter (i.e., less complex) than* s *involving rule* $\delta \rightarrow t$*.*

---

[10] For readability reasons we will make frequent use of ruleschemata like sentence $\overline{u} \bullet v \bullet \overline{w}$ in which $\overline{u}$ or $\overline{w}$ may be empty. The same holds for expressions like $\overline{\alpha}\backslash\beta/\overline{\gamma}$ in which $\overline{\alpha}$ or $\overline{\gamma}$ may be empty. The use of these schemata is supposed to be clear from the context.

This is in fact a weak notion of a positive characteristic sample because it only deals with the occurence of rules. In the context of this paper where we use grammars in Chomsky normal form this has the advantage that it becomes easy to prove the equivalence of the learned grammar and the original grammar. In fact simplification is a kind of overfitting to the notion of context-free grammars. In the context of categorial grammars, for example, the notion of the occurrence of rules does not make sense, since this grammatical formalism is lexicalistic. In the original definition in Adriaans, 1992a I referred to occurrence of constructions, thus allowing for stacking of rules in the sample sentences. Since the notion of construction is still vague a more general notion of characteristic sample has been introduced in the paragraph on complexity bounds. We now state the central theorem of the paper.

Definition 1.13 *A membership query for a grammar G is a function that is defined as follows:* $\texttt{member}(x) = 1$ *iff* $x \in L(G)$, $\texttt{member}(x) = 0$ *iff* $x \notin L(G)$.

Theorem 1.4 *Suppose that we have a characteristic sample S_c with m sentences of at most n words of a shallow context-free grammar G in Chomsky normal form with no lexical ambiguity and context- and expression separability and we can do membership queries. We can construct a grammar $G\prime$ that is weakly equivalent to G in time polynomial in m and n.*

Note that this theorem refers to the use of membership queries. Strictly spoken membership queries are not necessary when we are sampling under simple distributions. The current implementation of EMILE uses membership queries to speed up the learning process. It is easy to adapt EMILE to a situation where memberhip queries are not available. Such an implementation would take a lot more time to gather the right samples, but would still learn in polynomial time.

4.0.5 The EMILE 3.0 algorithm

In order to prove this theorem I describe the EMILE 3.0 algorithm in detail. I use pseudo SQL to describe the data manipulations.

The *explode* routine takes the sample and splits it into a set of expressions and contexts. Basically the output is a two dimensional matrix indicating which combinations of expressions and contexts lead to valid sentences.

`Routine: Explode`
`Input: A positive characteristic sample` $S_c \subseteq L(G)$
`with` m `sentences of at most` n `words`

```
Output:  partly filled table MATRIX
:begin
create table MATRIX(EXPRESSION, CONTEXT,SENTENCE-YN)
for all $\overline{x} \in S_c$ do
$l = |\overline{x}|$
for $i = 0$ to $l - 1$ do
for $j = 1$ to $l - i$ do
$p := \overline{x}.subarray([0, i])$
$q := \overline{x}.subarray([i, i + j])$
$r := \overline{x}.subarray([i + j + 1, l])$
insert $(q, p\backslash\sigma/r, 1)$ into MATRIX
od
od
od
:end
```

Because we want to prove that the grammar that we have learned is really equivalent it is useful to check all possible combinations of sentences and expressions. This is done by the following routine.

```
Routine:  Enrich
Input:  partly filled table MATRIX
Output:  completely filled table MATRIX
:begin
For all $x$ in (select EXPRESSION from MATRIX) do
For all $y\backslash\sigma/z$ in (select CONTEXT from MATRIX) do
// If the combination does not yet occur in MATRIX
If (select EXPRESSION from MATRIX
where EXPRESSION $= x$ and CONTEXT $= y\backslash\sigma/z$) = $\emptyset$
then
ANSWER = member($x \bullet y \bullet z$)
insert ($x, y\backslash\sigma/z$,ANSWER) into MATRIX
od
od
:end
```

The following routine clusters sets of expressions into groups that can be substituted for each other and then clusters sets of contexts that fit with these goups of expressions.

```
Routine:  Cluster
Input:  :  completely filled table MATRIX
```

```
Output:  Two tables with new type assignments,
LEFTTYPE for expressions and RIGHTTYPE for contexts
:begin
create table LEFTTYPE(EXPRESION, TYPE)
create table RIGHTTYPE(TYPE, CONTEXT)
Type# = 1
For all x in (select CONTEXT from MATRIX)
do

// If type has not yet been created
If not exist u in (select TYPE from LEFTTYPE)
such that
(select EXPRESSION from LEFTTYPE where TYPE = u)
=
(select EXPRESSION from MATRIX
where CONTEXT = x and SENTENCE-YN = 1)
then

// Cluster expressions in some context x
For all y in
(select EXPRESSION from MATRIX
where SENTENCE-YN= 1 and CONTEXT = x)
do
insert (y, Type#) into LEFTTYPE
od

// Cluster contexts that fit same expressions
For all z in (select CONTEXT from MATRIX)
do
if
(select EXPRESSION from MATRIX
where SENTENCE-YN = 1 and CONTEXT = z)
=
(select EXPRESSION from LEFTTYPE where TYPE = Type#)
then
insert (Type#,z) into RIGHTTYPE
od

// Adjust type counter
Type# = Type# +1
od
```

```
:end
```

We can create a rough lexicon for the grammar immediately.

```
Routine:  Create lexicon
Input:  Table LEFTTYPE
Output:  table LEXICON-RULES
:begin
create table LEXICON-RULES(RULE)
For all (x, α) in
(select EXPRESSION, TYPE from LEFTTYPE
where |EXPRESSION| = 1)
do
insert (α → x) into LEXICON-RULES
od
:end
```

This concludes the general part of EMILE that is valid for the whole class of shallow context-free grammars. In the next steps of the algorithm we will use the non-ambiguity and the separability constraints, so these results will be less general. Later I will argue that these constraints are not essential for the complexity of the algorithm in general. Since the grammar may contain ambiguous contexts, e.g. $\sigma \rightarrow \alpha \bullet \beta$ and $\sigma \rightarrow \gamma \bullet \beta$ this rough lexicon may be ambiguous too. In this case both expressions of types α and γ can be substituted in context σ/β. This will cause a new label in `LEXICON-RULES`, the extension of which is the union of type α and γ. In an unambiguous grammar it is undesirable to have this redundancy. Fortunately it is easy to clean up `LEXICON-RULES` and retain only simple type assignments. We remove all supersets from `LEXICON-RULES`. Note that since we have only ambiguous contexts supersets are the only problem, since all words in the original grammar are unambiguously assigned to a type.

```
Routine:  Clean Lexicon
Input:  table LEXICON-RULES
Output:  clean table LEXICON-RULES
:begin
For all π → q, ρ → q and π → s
such that ρ → s does not occur in LEXICON-RULES
Delete all rules of the form (π → ...) from LEXICON-RULES
:end
```

Note that LEFTTYPE and RIGHTTYPE may contain ambiguous types and contexts. Since we have *context separability* and *expression separability* and the sample is characteristic, we must be able to identify characteristic expressions and contexts in the sample. This is done with the following routines. First for contexts:

```
Routine:  Create characteristic context
Input:  table LEFTTYPE, RIGHTTYPE, MATRIX
Output:  table CHARACTERISTIC-CONTEXT
:begin
create table CHARACTERISTIC-CONTEXT(CONTEXT,TYPE)
For all α in (select TYPE from LEFTTYPE) do
NEXPR_α = (select EXPRESSION from LEFTTYPE
where not(TYPE = α))
EXPR_α = (select EXPRESSION from LEFTTYPE
where TYPE = α)
for all β\σ/γ in select CONTEXT from RIGHTTYPE do
if (δ̄, β̄\σ/γ̄, 1) occurs in MATRIX for each δ̄ in EXPR_α
and none in NEXPR_α
then
insert (β̄\σ/γ̄, α) into CHARACTERISTIC-CONTEXT
od
od
:end
```

Then for expressions:

```
   Routine:  create characteristic expression
Input:  table LEFTTYPE, RIGHTTYPE, MATRIX
Output:  table CHARACTERISTIC-EXPRESSION
:begin
create table CHARACTERISTIC-EXPRESSION(EXPRESSION, TYPE)
For all α in (select TYPE from LEFTTYPE)
do
NCONT_(α) = (select CONTEXT from CHARACTERISTIC-CONTEXT
where not (TYPE = α)
CONT_(α) = (select CONTEXT from CHARACTERISTIC-CONTEXT
where (TYPE = α)
For all β̄ in (select EXPRESSION from LEFTTYPE)
do
If (β̄, γ̄\σ/δ̄, 1) occurs in MATRIX for each
γ̄\σ/δ̄ in CONT_(α) and none in NCONT_(α)
```

```
then insert (β̄,α) into CHARACTERISTIC-EXPRESSION
od
od
:end
```

Since the lexicon is now unambiguous, for each terminal there is only one entry in LEXICON-RULES. This justifies the following routine:

```
Routine:  Convert-from-lexicon(s̄,γ̄)
Input:  table LEXICON-RULE, a string s̄ ∈ Σ*
Output:  A corresponding string γ̄ ∈ N*
:begin
l = |s̄|
for i = 1 to l
(select α → x from LEXICON-RULES
where x = s_i)
γ_i := α
:end
```

We can create a rough set of rules:

```
Routine:  Create rule
Input:  tables RIGHTTYPE and LEFTTYPE
Output:  table RULES
:begin
create table RULE(RULE)
for all (p̄,α) in
(select EXPRESSION , TYPE from LEFTTYPE
where |EXPRESSION| > 1)
convert-from-lexicon(p̄,π̄)
insert (α → π̄) into RULES
for all (ρ,s̄\σ/t̄) in (select TYPE, CONTEXT from RIGHTTYPE)
convert-from-lexicon(s̄,π̄)
convert-from-lexicon(t̄,τ̄)
insert (σ → π̄ • ρ • τ̄) into RULES
:end
```

The combination LEXICON-RULES and RULES will be called a *proto grammar*. We must clean this proto grammar to find the original rules. This is done via a controlled substitution process that is in fact a form of inverse derivation. Since such a process of inverse derivation is inductive we need to check the soundness of each step. Fortunately the separability conditions help us to do this. We need the following routine:

```
Routine:  Testvalid(π → ᾱ)
Input:  tables CHARACTERISTIC-CONTEXT,
CHARACTERISTIC-EXPRESSION, rule
Output:  true, false
:begin
Select a characteristic context x̄\σ/ȳ
for all π from CHARACTERISTIC-CONTEXT
Form ā by substituting for each α_i in ᾱ
a corresponding characteristic expression a_i from
CHARACTERISTIC-EXPRESSIONS
Check MEMBER(x̄ • ā • ȳ)
:end
```

The cleaning process itself takes the following form:

```
Routine:  Shorten rules
Input:  table RULES filled with rules
Output:  table SHORTRULES filled with short rules
:begin
create table SHORTRULES(RULE)
For all π → ᾱ in RULES such that |ᾱ| > 2
do
l := |ᾱ|
For i = 1 to l − 2
do
β̄ := ᾱ.subarray([0, i])
γ̄ := ᾱ.subarray([i + 1, l])
For all δ such that δ → β̄ ∈ RULES
For all η such that η → γ̄ ∈ RULES
If Testvalid(π → δ • η) =true then
do
Insert( π → δ • η) into SHORTRULES
od
od
For all α → β̄ in RULES such that |β̄| = 2 and
Testvalid(α → β) =true
do
insert α → β̄ into SHORTRULES
od
:end
```

Theorem 1.5 *The EMILE 3.0 algorithm is polynomial in m, the number of sentences in the original sample, and n, the maximal length of sentences in the original sample.*

Proof. It is clear that the number of elements in a first order explosion is equal to the number of ways in which we can place two brackets '(' and ')' grammatically in a sentence in such a way that there is at least one word between them[11]. We could also say that the first order explosion gives us the set of different sub-expressions and contexts of the sample space. Now there are n ways to place the first bracket and if the first bracket is before the k^{th} word then there are $(n-k)$ ways to place the second bracket: there are $1/2(n^2-n)$ ways to place brackets. We introduce a new constant $d = 1/2(n^2-n)$ that gives the upper bound of the total number of subexpressions that we can find in our sample. Clearly d is polynomial in m and n. The EMILE 3.0 algorithm simply consists of a sequence of successive for-next loops, all of which are bounded by m, n or d. Since there is no backtracking EMILE 3.0 is polynomial in m and n. A rough upper bound for the complexity of the total algorithm is $O(d^3)$. □

The maximum number of substitutions in the matrix is d^2. In practice however the number of substitutions will be considerably smaller than this maximum, because similar expressions may occur in several sentences, especially in languages with relatively small lexicons. There is a variety of routines that one can develop to speed up the learning process, but the rough description of the algorithm that I have given here is sufficient to understand the complexity. We have to prove that the algorithm does live up to our expectations. This is done by showing that the grammar that is learned by the algorithm actually contains the original grammar, plus a number of other rules that are harmless because they can be derived from the original grammar.

Theorem 1.6 *The tables* `LEXICONRULES` *and* `SHORTRULES` *contain a grammar $G\prime$ that is weakly equivalent to the original grammar G.*

Proof. We have to prove that $[\sigma]^G = [\sigma]^{G\prime}$ which means $\{\overline{x} | \sigma \overset{*}{\Rightarrow}_G \overline{x}\} = \{\overline{x} | \sigma \overset{*}{\Rightarrow}_{G\prime} \overline{x}\}, \overline{x} \in \Sigma^*$.

Now suppose that $G =< N, \Sigma, P, \sigma >$ and $G\prime =< N\prime, \Sigma\prime, P\prime, \sigma >$ and that $G\prime$ has been created by the EMILE 3.0 algorithm on the basis of a characteristic sample of G.

We first prove that $G \subseteq G\prime$:

[11] The n^{th} order explosion of a sentence gives all the possible ways to place n pairs of brackets.

- $\Sigma = \Sigma\prime$ because of the fact that the sample is characteristic i.e. each word occurs at least once.
- $N \subseteq N\prime$ (for a suitable relabeling of $N\prime$). Our sample is characteristic. For every $\alpha \in N$ the sample will contain at least one $\overline{\beta} \bullet \overline{\gamma} \bullet \overline{\delta}$ such that $\sigma \overset{*}{\Rightarrow}_G \overline{\beta} \bullet \overline{\gamma} \bullet \overline{\delta}$ and $\alpha \overset{*}{\Rightarrow}_G \overline{\gamma}$. Our sample is strongly seperable and it will contain at least one characteristic context $\overline{u}\backslash\sigma/\overline{w}$ that accepts only expressions of type α. Since the cluster algorithm checks all expressions against all contexts it will have found $\overline{\gamma}$ as an expression and $\overline{u}\backslash\sigma/\overline{w}$ as a context of type α such that: $\{\overline{x}|\texttt{LEFTTYPE}(\overline{x},\alpha)\} \subseteq \{\overline{x}|\alpha \overset{*}{\Rightarrow}_G \overline{x}\}$. Note that the converse $N\prime \subseteq N$ is not true. The algorithm will create new labels that do not exist in the original grammar.
- All the rules of the form $\delta \rightarrow \{t\}$ occurring in G are in `LEXICONRULES`. The sample S_c is characteristic. This means that $\{t\}$ occurs in the sample. There will be at least one sentence of the form $\overline{u} \bullet \overline{v} \bullet \overline{w}$ in which only expressions of type δ can be substituted. The sentence $\overline{u} \bullet t \bullet \overline{w}$ will be tested for membership. This will lead to a context $\overline{u}\backslash\sigma/\overline{w}$ and an expression $\{t\}$. The cluster algorithm will create a label δ such that $\overline{u}\backslash\sigma/\overline{w} \rightarrow \delta$ and $\delta \rightarrow \{t\}$.
- All the rules of the form $\alpha \rightarrow \beta \bullet \gamma$ occurring in G are in `SHORTRULES`. Since the sample is characteristic there will be at least one rule $\alpha \rightarrow \overline{\pi} \bullet \overline{\rho}$ with $\beta \rightarrow \overline{\pi}$ and $\gamma \rightarrow \overline{\rho}$. The **`shorten rule`** algorithm will make the substitutions to form $\alpha \rightarrow \beta \bullet \gamma$.

Conversely: $G\prime \subseteq G$:
Suppose that there is a sentence s that is valid in $G\prime$ but not in G. This sentence must have been constructed using a rule that is not valid in G. All the rules of the form $\delta \rightarrow \{t\}$ are valid in G because G and $G\prime$ are unambiguous and share the same lexicon. But also all the rules of the form $\alpha \rightarrow \beta \bullet \gamma$ are valid, because their validity has been tested explicitly before they were inserted in `SHORTRULES`.

□

The EMILE 3.0 algorithm as discussed above learns shallow unambiguous context-free grammars with context- and expression separability. A natural question is whether these constraints are necessary.

4.0.6 The nature of shallowness

The shallowness constraint seems to be essential for our results. How natural is shallowness? Are natural languages shallow and if so, why? There are a number of perspectives that one could have on shallowness:

- Complexity: we can efficiently sample all relevant constructions under all simple distributions.

- Syntax: shallowness favors languages with large lexica and relatively few constructive rules, or, expressed differently, a small set of types of rich cardinality.
- Efficiency: the creation of simle types with high cardinality enables the realisation of very complex structures with simple means.
- Natural language: indeed natural languages seem to be shallow. If the lexicon of an average natural language has 50.000 words than a counterexample would consist of a syntactical construction that could not be expressed in less than ca. 16 words[12].
- Evolution: shallowness seems to find a parallel in the concept of evolutionary development. First an abundance of simple building blocks needs to exist before new, more complex constructions are introduced. Complex organisms that result from an evolutionary development seem to have a shallow architecture. In this context there might be relation between shallowness and the concept of *logical depth* of a string discussed in Li & Vitányi, 1997.
- Learnability: last but not least it seems to be the case that shallow structures are relatively easy to learn.

Thus the concept of shallowness ties together a number of notions that seem to be essential for the understanding of natural language: evolutionary development, efficiency and learnability. Further research is necessary to see whether this view leads to fundamentally new insights, but as sweeping statement it is appealing.

It is important to note that I do not claim that semantic structures are shallow. Syntax needs to be efficient and learnable, but there is no limit to the complexity of the semantic concepts that we can express using language. So the syntax that we use to decribe a puzzle may be easy to learn, while it may be impossible or very hard to solve the puzzle itself.

4.0.7 Separability is non-essential

The separability constraints are, given the shallowness constraints, non-essential. The main reason for the introduction of these constraints in this paper is the fact that they make it easier to construct the proof that the grammar we learn is weakly equivalent with the original grammar. The only thing we really need are polynomial validity tests for expressions and contexts. It is easy to generalize the algorithm in such a way that context- and expression separability are not needed. We have to replace the notion of a *characteristic expression* for a certain non-

[12]At the 1997 CSLI workshop I offered thousand dollars for such a counter example in any known language. Nobody has claimed this money yet. The offer is still open.

terminal $\delta \in N$ with the notion of a *characteristic sample* for δ. This is a set of expressions $D_c \subseteq \{\overline{x} | \delta \stackrel{*}{\Rightarrow}_G \overline{x}\}$ such that all the expressions of D_c can be substituted for any expression of type δ, but for all type other than δ the set D_c will contain one or more expressions that cannot be substituted. Along the same lines one can replace the notion of a *characterstic context* with that of a *set of characteristic contexts* for a non-terminal δ. In such a *set of characteristic contexts* only expressions of type δ can be substituted in all contexts. If we adapt the notion of a characteristic sample in such a way that it facilitates the construction of characteristic samples and sets of characteristic contexts for each non-terminal than we can learn shallow context-free grammars without separability. The price we have to pay for this is an increased (but still polynomial) complexity of the `Testvalid` routine.

4.0.8 While sampling under simple distributions membership queries are not necessary

The current version of Emile uses membership queries to get both positive and negative examples of grammatical constructions. Note however that the complexity of the positive and negative examples that we need for shallow languages is bounded by $c \log n$. Lemma 1 tells us that we can sample all those constructions in time polynomial in $|G|$. Therefore we can simply build up our contexts and expressions matrix with positive examples alone. After some time the systematic non-occurrence of groups of sentences will be an indication of their ungrammaticality. This insight froms the basis of an incremental version, EMILE 4.0, that can learn from text alone. It is currently under development.

4.0.9 The nonambiguity constraint is non-essential

The only two places where we use the non-ambiguity constraint is in the `Clean Lexicon` and the `Convert-from-lexicon` routines. In the previous paragraph we showed that for shallow grammar we can always construct efficient validity tests. This means that the `Clean Lexicon` routine is not essential. It is just something that is nice to have if one wants to create a clean grammar at the end of the process. The `Convert-from-lexicon` routine is a different case. In the present version of the algorithm we assume a simple one-to-one correspondence between words and non-terminals, but natural language is certainly more complex, allowing for real lexical ambiguity. In order to deal with this we have to develop a more advanced `convert-from-lexicon` routine. Given the fact that we have efficient ways of testing the validity of a type assignment this is not very difficult. We just collect all candidate type assignments in `LEXICONRULES` for a rule in the protogrammar and test each of them. We even can allow for an occasional number of candidate type assignments that is exponential in the length of the rule given that the length of the rule is bounded by $log|G|$. But, a language that would have consistently an exponential number of candidate readings for even simplest sentences is not shallow, so the concept of shallowness excludes extreme forms of ambiguity.

4.0.10 Natural languages are only mildly non-separable and mildly ambiguous

This shows that using the clustering techniques of EMILE we can learn shallow languages efficiently under simple distributions. Ambiguity is allowed. I do not believe, however, that natural languages have an extreme form of ambiguity in which all words belong to almost all types. This seems in contradiction with the notion of efficient communication[13]. The analysis of EMILE shows, in my opinion, that the notion of separability is much more important for language learning than that of lexical ambiguity. Although it is probably not true that natural languages are strictly separable, in the sense that there is a simple definitive validity test for every expression, separability seems to be extremely important for efficient communication and learnability. It is my conjecture that natural languages are, if not completely separable, only mildly non-separable, i.e. the cases where no simple validity test are available are exceptions. The 'wuzzle' example at the beginning of this paper illustrates this.

4.0.11 Problems of the EMILE approach

The proof presented in this paper shows that context-free grammars for natural languages can be learned efficiently under conditions that could come close to natural conditions. Yet there is still a lot of meat left over for a critical analysis of my approach. An extensive critical analysis of the EMILE approach was made by Dörnenburg, 1997. I discuss some points briefly:

- EMILE is polynomial, but still way too complex to be of any practical interest. Especially the required number of negative membership queries is ridiculous. This is essentially true. In almost all cases the algorithms described in this paper do not come close to the most efficient procedure to learn a grammar. In fact the procedure described here could be a *worst case batch procedure*. As I have shown in the examples it is very well possible to make op-

[13]The Newspeak that is described by George Orwell in '1984' seems an impossible vehicle for human communication: "Any word in the language (in principle this applied even to very abstract words such as *if* or *when*) could be used either as verb, noun, adjective or adverb."

timalisation decisions in an on line version of the algorithm. In the current versions of EMILE I have mainly concentrated on the proof of the existence of polynomial sample- and time complexity algorithms. If we have these upper bounds we can gradually work our way downward to implementations of learning algorithms that are really efficient and powerful. There is still a lot of work to be done here, but there is also an abundance of optimization strategies possible (For a discussion see Dörnenburg, 1997).

- It has been noted in a number of places in the literature on child language acquisition that parents do not in fact provide reliable indicators of the well-formedness of their children's utterances, responding instead to the truth of the utterances (See for example Marcus, 1993). I still believe however that children get implicit negative information. In Adriaans, 1992a I introduced a domain theory for language learning that showed that the notions of syntactic and semantic validity are interrelated and that it is possible to combine syntactic and semantic learning into one EMILE-like algorithm. In this light the remark that parents react to the truth of an utterance and not to its syntactic validity seems to miss the point. Obviously the process is in reality much more complex and involved than the algorithms that are described in this paper.
- Some critics of the shallowness approach have argued that it is unrealistic to suppose that the complexity of the grammar is known in advance. Since in reality we never know this when we start the learning process we would not be able to determine the right sample size. In practical situations this is less of a problem if we note that shallow grammars tend to have large lexica. When we start to sample a language we will collect new lexical elements quickly. Later the introduction of new elements will slow down to stop completely after some time. When no new lexical elements have occurred for a certain period we may assume that we have seen all relevant examples with high probability. An exact statistical analysis of this phenomenon is outside the scope of this paper. There is some empirical evidence that this approach to learning G while sampling works in practice, but further research is necessary (Adriaans *et al.*, 1993).
- Some critics say the result is of little practical significance because we can never sample according to $\mathbf{m}$ in the real world. Although this is true it is not as harmful as one might think. Real world distributions in teaching situations are dominated by $\mathbf{m}$. Any teacher that uses a method to select examples will select according

to a simple distribution dominated by **m**. In all those real life situations EMILE would work.

- Some critics have remarked that the grammars learned by EMILE are sloppy and have an abundance of rules (Dörnenburg, 1997). The grammars parse the original language but do not give an indication of the real parts of speech of a language. This is true and it is a real problem for the current version of EMILE. In fact EMILE 3.0 learns a grammar that is *functionally complete* for the original sample. This means that it gives a parsing for any valid bracketing of sentences in the original sample. Functional completeness is a nice quality of categorial grammars but undesirable for real life systems. I believe that, when one is sampling under simple distributions, a statistical analysis of the initial substitution matrix may lead to the identification of real parts of speech. A suggestion that might be interesting is to do away with syntactic rules altogether and use the clustered matrix as a means to parse any given sentence. In this way EMILE would move in the direction of connectionistic approaches.

An extensive discussion of other approaches to natural language learning in relation to our solution would lead beyond the scope of this paper. It is important to notice that the EMILE approach is not in competition with other techniques like n-gram search, connectionistic approaches and stochastic context-free grammars. All these techniques can be analysed in the context of sampling under **m**, separability and shallowness. It is clear that n-gram search will never learn the complete class of shallow grammars. The same holds probably for connectionistic approaches (Finch & Chater, 1992; Honkela *et al.*, 1995). There are various interesting approaches using stochastic grammars and it is not unconcievable that a stochastic algorithm might learn the whole class of shallow grammars under simple distributions efficiently. There is an interesting parallel between the use of inside and outside probabilities and my notion of expressions and contexts (Charniak, 1993). A number of researchers have tackled the problem of learning grammars from bracketed corpora (Buszkowski, 1989, Sakakibara, 1990). It is clear that this is a much easier task than learning from unbracketed samples since one only has to guess the labels. In reality native speakers certainly get some structural information about utterances (intonation, stress etc.) but the majority of the material that we use to learn a language is unbracketed so these efforts have limited practical significance.

5 Conclusion

I believe that the approach presented here has some advantages over other proposals in the literature both from a mathematical and a psychological perspective. The elegant theory behind categorial grammars makes it easy to analyse the consequences of learning algorithms. It provides us with a foundation of the simple idea of learning by substitution that has great psychological appeal. The application of Kolmogorov complexity in this context has led to the insight that human languages might be shallow. My conjecture would be that this holds indeed for all human languages and I would be curious to see some counter examples. Any syntactical construction that could not be illustrated in a sentence of less than say 16 words would constitute such a counter example.

It would be interesting to analyse the actual first and second language acquisition processes in the EMILE context, especially the notion of separability (Radford, 1990). It seems that children go through phases where they use characteristic contexts that are actually non grammatical to learn type assignments of classes of words: 'Want ice-cream', 'Want doll'. The context seems to act as a monolitic whole without syntactic structure that facilitates one stage of the learning process. In a later stage the contexts are analysed into their correct syntactic constituents 'I want an ice-cream'. This could account for the chaotic structure of the first language acquisition process.

A possible approach in this respect is the use of ontologies. An ontology in this case is a set of substitution contexts that gives us validity tests for the set of semantic/syntactic types. There is empirical evidence that young children use such contexts (Radford, 1990). Learning an ontology with EMILE is quite cumbersome, but once an ontology is acquired the rest of the acquisition of the lexicon can be learned in time linear to the size of the lexicon. The type assignment for each expression can be determined easily by substitution in one or more appropriate contexts. The wuzzle example at the beginning of this paper illustrates this. We do not feel the need to substitute the word 'wuzzle' in all possible contexts to make the correct type assignment. This leads to a psychological model of language acquisition where children first learn an ontology (= basic grammar) on a shallow subset of the language and then shift to extremely efficient learning with ontologies.

Another interesting application of separability is the deciphering of unknown texts. When Champollion identified the cartouches on the obelisk of Philae as devices to distinguish royal names this gave him the first clue to the understanding of hieroglyphic inscriptions. It is a nice example of characteristic contexts, the cartouche as device to identify royal names, and characteristic expressions, the royal names of Cleopatra and Ptolemaios. It would be a nice idea to make sparse substitution matrices of undeciphered texts to see what type of grammatical contructions could be corroborated or falsified. Inscriptions that one could think of are Cretan Linear A, the famous Phaistos disk (Godart, 1995) and the Pictish ogham found in Scotland (Sermon, 1992).

Consequently there are a number of ways in which this research could be extended:

- a further analysis of the concept of shallowness in relation to languages of various complexity (e.g. first order logic). How fundamental is shallowness? Are there non-shallow languages that are PAC-learnable? The concept of shallowness has a much broader area of application in the context of learning under simple distributions. There is a whole class of shallow languages that we can learn efficiently.
- It would be interesting to compare the concept of shallowness to other constraints that make grammars learnable, like k-locality. Does locality imply shallowness of vice versa?
- a deeper analysis of the bias involved in natural language learning in relation to other classes of learning algortihms e.g. connectionistism, stochastic context-free grammars, ILP.
- an efficient implementation of the EMILE primitives in a relational database architecture (e.g. for textmining purposes).
- an incremental version of EMILE. An interesting approach to this problem would be to use a clustered version of the matrix to parse sentences directly without the construction of a complete grammar. New material could then directly be incorporated in the matrix. This will allow the usage of an initial grammar containing the basic constructs of the language, which can be tuned to capture the details of specific words and exceptions.
- Application of EMILE to real life problems of a different nature: musical composition, pattern recognition, deciphering messages, inscriptions etc.

Given these additional results it may some day in the not too distant future be possible to realise dialogical systems that really learn a subset of an existing natural language in a dialogue.

Acknowledgements

I thank my colleagues at Syllogic for their support of this type of rather fundamental non-commercial research. I thank the anonymous referees for their valuable suggestions and criticisms. I thank Arno Knobbe for the first implementation of EMILE 3.0 and for a lot of stimulating discussions. Other persons that contributed to this paper in various forms are Paul Vitányi, Erik Dörnenburg, Peter van Emde Boas and Johan van Benthem.

References

Abe, N. *Learnability and locality of formal grammars,* in Proceedings of the 26th Annual meeting of the Association of computational linguistics, 1988.

Adriaans, P.W. *Language Learning from a Categorial Perspective.* PhD thesis, University of Amsterdam, 1992.

Adriaans, P.W. *Bias in Inductive Language Learning*, in Proceedings of the ML92 Workshop on Biases in Inductive Learning, Aberdeen, 1992.

Adriaans, P.W., Janssen, S., Nomden, E. *Effective identification of semantic categories in curriculum texts by means of cluster analysis*, in workshop-notes on Machine Learning Techniques for Text Analysis, Vienna, 1993.

Adriaans, P.W., Knobbe, A.K. *EMILE: Learning Context-free Grammars from Examples*, in Proceedings of BENELEARN'96

Angluin, D. *Computation learning theory: survey and selected bibliography*, In Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992.

Brown,P.F., V.J.D. Pietra, P.V. DeSouza, J.C. Lai end R.L. Mercer, *Class-based n-gram models of natural language*, Computational linguistics 18, 4, 467-479.

Buszkowski, W., *Generative Power of categorial Grammars* in R.T. Oehrle, E. Bach and D. Wheeler (eds.), *Categorial Grammars and Natural Language Structures*, D. Reidel Publishing Company, Dordrecht and Boston, 1988.

Buszkowski, W. and G. Penn, *Categorial Grammars Determined from Linguistic Data by Unification*, The University of Chicago, Technical Report 89-05, June 1989.

Charniak, E. *Statistical Language Learning*, MIT Press, Cambridge, 1993.

Dörnenburg, E., *Extension of the EMILE algorithm for inductive learning of context-free grammars for natural languages.* Masters thesis, university of Dortmund, 1997.

Finch, S., Chater, N. *Automatic methods for finding linguistic categories*, in Artificial Neural Networks, Vol. 2, Elseviers Science Publishers, 1992.

Gács, P. *On the symmetry of algorithmic information*, Soviet Math. Dokl. 15, 1974.

Godart, L., *The Phaistos Disc, The enigma Of an Eegean Script.* Itanos Publications, 1995.

Gold, E. M., *Language Identification in the Limit*, in Information and Control, Vol. 10, pp 447-474. Academic Press, 1967.

Gold, E. M., *Complexity of automaton identification from given data.* Information and Control, 37:302-320, 1978.

Honkela, T., Pulkki, V., Kohonen, T. *Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map*, in Proceedings of the International Conference on Artificial Neural Networks 1995.

Hopcroft J. and J. Ullman, *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, MA, 1979.

Kanazawa, M. *Learnable Classes of Categorial Grammars*, PhD thesis, University of Stanford, 1994.

Lambek, J., *Categorial and Categorical Grammars*, in R.T. Oehrle, E. Bach, D. Wheeler (eds.), *Categorial Grammars and Natural Language Structures*, Reidel, Dordrecht and Boston, 1988.

Lankhorst, M.M. *Grammatical inference with a genetic algorithm*, in Proceedings of the 1994 EUROSIM Conference on Massively Parallel Applications and Development, pp 423-430, Elsevier, 1994.

Levin, L. A. *Laws of information conservation (nongrowth) and aspects of the foundation of probability theory,* Problems Inform. Transmission, 10, 1974.

Li, M. and Vitányi, P.M.B., *An Introduction to Kolmogorov Complexity and its Applications*,second ed. Springer Verlag, NY, 1997.

Li, M. and Vitányi, P.M.B., *Learning simple concepts under simple distributions*, SIAM Journal of Computing, pg. 911-935, 1991.

Marcus, G., *Negative evidence in language acquisition.* Cognition, vol. 46, pg. 53-85, 1993.

Natarajan, B., *Machine Learning, A theoretical approach.* Morgan Kaufmann Publishers, San Mateo, CA, 1991.

Oehrle, R.T., Bach, E., Wheeler, D. (eds.), *Categorial Grammars and Natural Language Structures*, D. Reidel Publishing, 1988.

Pentus, M. *Lambek Grammars are Context Free*, in IEEE Symposium on Logic in Computer Science, 1993.

Pitt, L., *Inductive inference, dfa's, and computational complexity.* Proc. Analogical and Inductive inference (K.P. Jantke, Ed.), Lecture notes in Artificial Intelligence 397, pg. 18-44, 1989.

Pitt, L., Warmuth, M.K., *The minimum DFA consistency problem cannot be approximated within any polynomial*, in Proceedings of the Twenty-first Annual ACM symposium on Theory of Computing, Seattle, Washington, May 1989.

Radford, A., *Syntactic Theory and the Acquisition of English Syntax.* Blackwell Publishers, 1990.

Sakakibara, Y., *Learning context-free grammars from structural data in polynomial time. Theoretical Computer Science*, vol. 76, pg 223-242, 1990.

Sakamoto, H, *Language learning from membership queries and characteristic examples.* Proc, 6th International Workshop on Algorithmic Learning Theory, (K.P. Jantke, T. Shinohara, T. Zeugmann, Eds.), Lecture Notes in Artificial Artelligence 997, pg 55-65, 1995.

Sermon, R., *Pictish Ogham inscriptions catalogue*. unpublished, 1992.

Valiant, L.G., *A Theory of the Learnable*, in Communications of the ACM, Vol. 27, No. 11., pp 1134-1142, 1984.

Yokomori, T., *Learning context-free languages efficiently, A report on recent results in Japan*, Lecture notes on artificial intelligence, 397, 1989.