Sublogarithmic Ambiguity

Klaus Wich
E-mail: wich@informatik.uni-stuttgart.de

Institut für formale Methoden der Informatik, Universität Stuttgart,
Universitätsstr. 38, 70569 Stuttgart.

Abstract. So far the least growth rate known for a divergent inherent ambiguity function was logarithmic. This paper shows that for each computable divergent total non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ there is a context-free language L with a divergent inherent ambiguity function g below f. This proves that extremely slow growing divergent inherent ambiguity functions exist. For instance there is a context-free language L with infinite inherent ambiguity below $\log^*$.

1 Introduction

A context-free grammar G is unambiguous if it does not have two different derivation trees for any word. A context-free language is unambiguous if it is generated by an unambiguous context-free grammar. Context-free grammars and languages are ambiguous if they are not unambiguous. Ambiguous context-free languages are also called inherently ambiguous. The existence of ambiguous context-free languages is shown in [12]. Ambiguous context-free grammars and languages can be distinguished by their degree of ambiguity, that is, by the least upper bound for the number of derivation trees which a word can have. There are examples for k-ambiguous languages for each $k \in \mathbb{N}$ [9]. But even languages with infinite degree of ambiguity exist [14, 5]. They can be distinguished by the asymptotic behaviour of their ambiguity with respect to the length of the words. There is an efficient algorithm which computes for each cycle-free context-free grammar G a constant k with the promise that the ambiguity of G is either $\mathcal{O}(n^k)$ or $2^{\Theta(n)}$[17]. However which alternative is the case is undecidable [15]. Languages with ambiguity $2^{\Theta(n)}$ and with ambiguity $\Theta(n^k)$ for each $k \in \mathbb{N}$ are presented in [10].

The class of languages with polynomially bounded ambiguity ($PCFL$) has some interesting properties. For instance [17] shows that $PCFL$ is the closure of the class of unambiguous context-free language under a restricted form of projection. Combined with [13, Theorem 3] one easily obtains that each language $L \in PCFL$ can be recognised in $\mathcal{O}(\log n)$ time on a CREW PRAM. Polynomially bounded ambiguity is also important in the field of random generation [4].

Infinite sublinear ambiguity is interesting for two reasons: Firstly from the theoretical point of view it is interesting to know whether there is a gap between constant ambiguity and the "lowest" possible divergent ambiguity as is the case between polynomially bounded and exponential ambiguity. Secondly it

is well known that the parsing algorithm of Early [6, 1] parses general context-free grammars in $\mathcal{O}(n^3)$ time while unambiguous context-free grammars are parsed in $\mathcal{O}(n^2)$ time. The proof which shows the speed up for unambiguous grammars can easily be generalised to $\mathcal{O}(n^2 \cdot \hat{d}_G(n + k_G))$ time for an arbitrary reduced context-free grammar G where $\hat{d}_G$ is the ambiguity function of G and $k_G \in \mathbb{N}$ is a constant only depending on G.[1] Thus Earley's algorithm parses languages with sublinear ambiguity faster than in cubic time. Languages with inherent square-root and logarithmic ambiguity can be found in [16], respectively. In [18] the following theorem has been shown:

Theorem 1.1. *The set of ambiguity functions for cycle-free context-free grammars and the set of inherent ambiguity functions coincide.*

Thus in order to prove the existence of infinite but sublogarithmic inherent ambiguity it suffices to present a cycle-free context-free grammar with the corresponding ambiguity.

It is well known that each recursively enumerable language is the homomorphic image of the intersection of two context-free languages [2, 7, 8]. In this paper we use similar ingredients. For each Turing machine M we construct context-free grammars generating languages of the form $(\mathcal{L}\{\#\})^*\mathcal{F}(\{\#\}\mathcal{R})^*$, where F is an unambiguous context-free subset of M's configurations, while $\mathcal{L}$ and $\mathcal{R}$ generate pairs of configurations separated by a "#" symbol, such that the right configuration is obtained in one step of M from the left one. Moreover exactly one of the configurations of such a pair is written in reverse. In case of $\mathcal{L}$ the right configuration is reversed, while for $\mathcal{R}$ it is the left one. It turns out that the ambiguity function of these grammars is dominated by the ambiguity of the words which lie in $\mathcal{F}(\{\#\}\mathcal{R})^* \cap (\mathcal{L}\{\#\})^*\mathcal{F}$. A word w within this set represents a segment of a computation of the Turing machine M which starts and ends in a configuration belonging to $\mathcal{F}$. Moreover the ambiguity of w is the number of times a configuration which belongs to F and which is preceded by an even number of configurations occurs in w. Since we are free in the choice of the underlying Turing machine M and the unambiguous context-free set $\mathcal{F}$ we have a strong tool to design very low divergent ambiguity functions. Roughly speaking, it is sufficient to find candidates for M and $\mathcal{F}$, such that computations of M containing many configurations in $\mathcal{F}$ cannot be to short.

2 Preliminaries

If not stated otherwise Σ is an arbitrary finite non-empty alphabet in the sequel. The symbol # is not in Σ. The empty word is denoted by ε. For $w \in \Sigma^*$ the *reversal* w^R of w is defined by $\varepsilon^R = \varepsilon$ and $(au)^R = u^R a$ for $u \in \Sigma^*$ and $a \in \Sigma$. The length of a word $w \in \Sigma^*$ is denoted by $|w|$. For all $n \in \mathbb{N}$ we define $\Sigma^n := \{w \in \Sigma^* \mid |w| = n\}$, $\Sigma^{\leq n} := \cup_{i \in \{0,\ldots,n\}} \Sigma^i$, and $\Sigma^{<n} := \Sigma^{\leq n} \setminus \Sigma^n$. A

[1] The constant k_G can be omitted if $\exists c \in \mathbb{N} : \forall n \in \mathbb{N} : \hat{d}_G(n+1) \leq c \cdot \hat{d}_G(n)$. An ambiguity function which violates this condition is not known.

context-free grammar is a quadruple $G = (N, \Sigma, P, S)$ where N and Σ are two disjoint alphabets of *nonterminals* and *terminals*, respectively, $P \subseteq N \times (N \cup \Sigma)^*$ is a finite set of *productions*, and $S \in N$ is the *start symbol*. The reader is assumed to be familiar with derivation trees and the definition of the language generated by G as defined in [8]. A context-free grammar G is *cycle-free* if a nonterminal cannot reproduce itself in a non-void derivation.

Let $G = (N, \Sigma, P, S)$ be a context-free grammar. The ambiguity of a word w is its number of derivation trees. The *ambiguity power series* of G is a function $d_G : \Sigma^* \to \mathbb{N}$ which maps each word to its ambiguity. This function is not well defined if a single word has infinite ambiguity. Let $G = (N, \Sigma, P, S)$ be a context-free grammar with a well defined ambiguity power series. The *ambiguity function* $\hat{d}_G : \mathbb{N} \to \mathbb{N}$ of G is $\hat{d}_G(n) = \max\{d_G(w) \mid w \in \Sigma^{\leq n}\}$, i.e., it maps each $n \in \mathbb{N}$ to the ambiguity of the most ambiguous word of length up to n. The grammar G is *unambiguous* if its ambiguity function is bounded by 1.

The grammar G is *reduced* if it does not contain useless symbols, i.e., symbols which does not occur in any derivation tree. Useless symbols can easily be eliminated. This elimination preserves the ambiguity power series. Therefore we do not need to consider non-reduced context-free grammars. It is easily seen that a reduced context-free grammar has a well defined ambiguity power series if and only if it is cycle-free.

A context-free language is unambiguous if it is generated by some unambiguous context-free grammar. It is ambiguous otherwise. For each context-free language L and each $n \in \mathbb{N}$ there is a context-free grammar G such that all the words of length up to n are generated unambiguously, i.e., $\hat{d}_G(n) \leq 1$. In case L is ambiguous to increase n one has to switch to context-free grammars with larger and larger pumping constants. But for an ambiguous language it may be possible to specify the least length of a word with a given ambiguity up to a constant depending only on the the pumping constant of the used context-free grammar. This leads us to the following definition taken from [18]:

Definition 2.1. *Let L be a context-free language and $f : \mathbb{N} \to \mathbb{N}$ a function. The language L is* f-ambiguous *if*

1. *there is a context-free grammar G such that $L = L(G)$ and $f = \hat{d}_G$ and*
2. *for each context-free grammar G such that $L = L(G)$ there exists a $c \in \mathbb{N}$ such that $f(n) \leq \hat{d}_G(cn)$ for all $n \in \mathbb{N} \setminus \{0\}$.*

A function $f : \mathbb{N} \to \mathbb{N}$ is an *inherent ambiguity function* if there is a context-free language L such that L is f-ambiguous. If L is f-ambiguous then L is f'-ambiguous for all non-decreasing functions f' such that f' agrees with f for all but a finite number of arguments. Note that the question whether each context-free language has an inherent ambiguity function is open. Now we have introduced all the notions needed to properly understand Theorem 1.1, which occurred in the introduction.

The reader is assumed to be familiar with single tape Turing machines as defined in [8]. Whenever we refer to a Turing machine in this paper we mean this type. Whether the Turing machine is deterministic or nondeterministic or

whether the tape is single or both sided infinite does not matter here. But it may help to think of deterministic machines. A *configuration* of a Turing machine consists of the tape content, the state of the Turing machine and the position of the head. It is denoted by a word consisting of the shortest string which represents the coherent portion of the tape which covers all the non blank cells and the position of the tape head. The head is denoted immediately to the left of the tape cell the machine reads in the next step. For each Turing machine M the corresponding set of configurations is denoted by ID_M. The relation $\vdash_M$ contains all the pairs of configurations $(id_a, id_b) \in ID_M \times ID_M$ where id_b is obtained from id_a by a single step of M.

3 Block Correlation Languages

Our aim is to find for each computable divergent non-decreasing function f an inherent ambiguity function g which falls below f. As a tool to design divergent ambiguity functions with a very slow growth rate we introduce block correlation languages.

Definition 3.1. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation. Then*

$$\mathcal{L}(R) := \{u\#v^R \mid (u,v) \in R\} \text{ and } \mathcal{R}(R) := \{u^R\#v \mid (u,v) \in R\}.$$

The relation R is (unambiguous) context-free *if $\mathcal{L}(R)$ and $\mathcal{R}(R)$ are both (unambiguous) context-free languages.*

It can be shown that $\mathcal{L}(R)$ is (unambiguous) context-free if and only if $\mathcal{R}(R)$ is (unambiguous) context-free. But instead to prove this statement for arbitrary R it is easier to check both languages for the relations considered below.

Definition 3.2. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be an unambiguous context-free relation, and $\mathcal{F} \subseteq \Sigma^*$ an unambiguous context-free language. The* block correlation language *over the relation R and the set $\mathcal{F}$ is defined by:*

$$L(R,\mathcal{F}) := \big(\mathcal{L}(R)\#\big)^* \mathcal{F} \big(\#\mathcal{R}(R)\big)^*.$$

If $L(R,\mathcal{F})$ is a block-correlation language then $\mathcal{F}$ is called the corresponding language of free blocks.

Definition 3.3. *A* canonical grammar *for a block correlation language $L(R,\mathcal{F})$ over a relation $R \subseteq \Sigma^* \times \Sigma^*$ and a set $\mathcal{F} \subseteq \Sigma^*$ is a context-free grammar:*

$$G := \big(\{S,A\} \dot\cup N_{\mathcal{L}} \dot\cup N_{\mathcal{R}} \dot\cup N_{\mathcal{F}},\ \Sigma \cup \{\#\},\ P \cup P_{\mathcal{L}} \cup P_{\mathcal{R}} \cup P_{\mathcal{F}},\ S\big).$$

where $G_{\mathcal{L}} := (N_{\mathcal{L}}, \Sigma \cup \{\#\}, P_{\mathcal{L}}, S_{\mathcal{L}})$, $G_{\mathcal{R}} := (N_{\mathcal{R}}, \Sigma \cup \{\#\}, P_{\mathcal{R}}, S_{\mathcal{R}})$, and $G_{\mathcal{F}} := (N_{\mathcal{F}}, \Sigma, P_{\mathcal{F}}, S_{\mathcal{F}})$ are unambiguous context-free grammars generating $\mathcal{L}(R)$, $\mathcal{R}(R)$, and $\mathcal{F}$, respectively. Moreover P is defined by:

$$P := \{S \to S_{\mathcal{L}}\#S,\ S \to S_{\mathcal{F}}A,\ A \to A\#S_{\mathcal{R}},\ A \to \varepsilon\}.$$

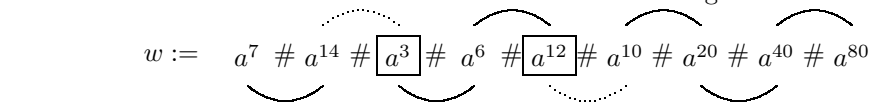
Note that $G_{\mathcal{L}}$, $G_{\mathcal{R}}$, and $G_{\mathcal{F}}$ have pairwise disjoint sets of nonterminals. Moreover the nonterminal sets does not contain the symbols S and A, respectively. This is expressed by the dot on the union symbols.

Let G be a canonical grammar which generates a block correlation language $L(R, \mathcal{F})$ with all the sets and symbols named as above and let $G' := (\{S, A\}, \{S_\mathcal{L}, S_\mathcal{R}, S_\mathcal{F}, \#\}, P, S)$. Obviously the grammar G' is unambiguous and generates the regular language $(S_\mathcal{L}\#)^* S_\mathcal{F}(\#S_\mathcal{R})^*$. A given derivation tree ρ of G generating a word w can always be trimmed in a unique way to obtain a derivation tree ρ' of G'. Let α be the frontier of ρ'. If we know α we can retrieve ρ' since G' is unambiguous. Moreover, except for the first and last symbol, each occurrence of $S_\mathcal{L}$, $S_\mathcal{R}$ and $S_\mathcal{F}$ in α is immediately preceded and followed by a "#" symbol. Furthermore each string of terminals generated by $S_\mathcal{L}$ or $S_\mathcal{R}$ contains exactly one "#" symbol and a terminal string generated by $S_\mathcal{F}$ never generates a "#" symbol. Therefore each occurrence of the symbols $S_\mathcal{L}$, $S_\mathcal{R}$ and $S_\mathcal{F}$ in α can be uniquely matched with the infix of w it generates. This is sufficient to complete the remainder of ρ uniquely since the grammars $G_\mathcal{L}$, $G_\mathcal{R}$, and $G_\mathcal{F}$ are unambiguous. Thus if we know w and α we can uniquely retrieve the whole derivation tree ρ. But this does not mean that G is necessarily unambiguous since w does not determine α in general. We can only deduce the length of $\alpha \in (S_\mathcal{L}\#)^* S_\mathcal{F}(\#S_\mathcal{R})^*$, but there may be several permissible position for $S_\mathcal{F}$. We consider the "#" symbols as markers factorising w into blocks. Thus $S_\mathcal{F}$ generates exactly one block called the *free block* in the sequel. The free block is preceded and followed by strings of the form $\big(\mathcal{L}(R)\#\big)^*$ and $\big(\#\mathcal{R}(R)\big)^*$, respectively. (In particular this implies that the free block is preceded and followed by an even number of blocks.) The number of derivation trees for w coincides with the number of decompositions of w satisfying the requirements stated above. The result of our discussion is summarised in the following lemma:

Lemma 3.4. *Let G be a canonical grammar generating a block correlation language $L(R, \mathcal{F})$ over a relation $R \subseteq \Sigma^* \times \Sigma^*$ and a set $\mathcal{F} \subseteq \Sigma^*$, and let $w \in \big(\Sigma \cup \{\#\}\big)^*$. Then the number of derivation trees for w is:*

$$d_G(w) = |\{i \in \mathbb{N} \mid w \in \big(\mathcal{L}(R)\#\big)^i \mathcal{F} \big(\#\mathcal{R}(R)\big)^*\}|.$$

Example 3.5. Let $\Sigma = \{a\}$ and $R := \{(a^i, a^{2i}) \mid i \in \mathbb{N}\}$. Here $\mathcal{L}(R) = \mathcal{R}(R)$ since Σ is unary. To compute the ambiguity of a word w in a canonical grammar for the block correlation language $L(R, \Sigma^*)$ we consider each pair of consecutive blocks in w separated by a "#" symbol. From left to right we draw alternating arcs below and above consecutive pairs of blocks starting with an arc below the leftmost pair. An arc is drawn with a solid line if the pair is in relation, i.e., the number of a's in the right block is twice the number of a's in the left one. Otherwise the arc is dotted. Let us consider the following word:



By definition the free block is preceded and followed by an even number of blocks. Such a block is a candidate for the free block if all the arcs below the word to its left and all the arcs above the word to its right are solid. These criteria are satisfied for exactly those blocks of w written in boxes. Therefore the word w has exactly two derivation trees for any canonical grammar generating $L(R, \Sigma^*)$.

Note that a canonical grammar generating a block correlation language is not always the least ambiguous grammar generating it. For instance consider the unary alphabet $\Sigma = \{a\}$ and the unambiguous context-free relation $R := \{(a^i, a) \mid i \in \mathbb{N}\}$. Then it is easily seen that $L(R, \Sigma^*) = (a^*\#a\#)^*a^*(\#a^*\#a)^*$, which is regular, and therefore unambiguous context-free. Despite that, for each canonical grammar generating $L(R, \Sigma^*)$ and for each $i \in \mathbb{N}$ the word $(a\#)^{2i}a$ has $i + 1$ derivation trees.

Definition 3.6. *Let G be a context-free grammar over Σ. Then the* support *of $\hat{d}_G$ is the set:*

$$\mathit{support}_G := \{w \in L(G) \mid \forall u \in \Sigma^{<|w|} : d_G(u) < d_G(w)\}$$

Thus a word w is in the support of the ambiguity function of a context-free grammar G if it is a shortest word with ambiguity at least $d_G(w)$. To determine the ambiguity function $\hat{d}_G$ of G it is sufficient to consider the words in $\mathit{support}_G$ and their corresponding ambiguities. More precisely the ambiguity function $\hat{d}_G$ is uniquely determined by the set:

$$\{(|w|, d_G(w)) \in \mathbb{N} \times \mathbb{N} \mid w \in \mathit{support}(G)\}.$$

But how do the words in the support of a canonical grammar for a block correlation language look like? It turns out to be necessary for them that each pair of consecutive blocks is correlated. In the notation of Example 3.5 this means that a word in the support of a canonical grammar never has a "dotted" arc connecting consecutive blocks. Before showing this formally, we define:

Definition 3.7. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation and $\mathcal{F} \subseteq \Sigma^*$ a formal language. Then*

$$\mathit{val}(R, \mathcal{F}) := \left\{ w_0\#w_1^R\#\cdots\#w_{2n} \;\middle|\; \begin{array}{l} w_0, w_{2n} \in \mathcal{F} \wedge \\ \forall i \in \{0, \ldots, 2n-1\} : (w_i, w_{i+1}) \in R \end{array} \right\}.$$

It is easily seen that $\mathit{val}(R, \mathcal{F}) = \big(\mathcal{L}(R)\#\big)^*\mathcal{F} \cap \mathcal{F}\big(\#\mathcal{R}(R)\big)^*$ for each relation R and each language $\mathcal{F}$ over Σ.

Theorem 3.8. *Let G be a canonical grammar of some block correlation language $L(R, \mathcal{F})$. Then*

$$\mathit{support}_G \subseteq \mathit{val}(R, \mathcal{F}).$$

Proof. Let $w \in L(R, \mathcal{F}) \setminus \mathit{val}(R, \mathcal{F})$. By definition $w \in \big(\mathcal{L}(R)\#\big)^*\mathcal{F}\big(\#\mathcal{R}(R)\big)^*$. Since $\mathit{val}(R, \mathcal{F}) = \big(\mathcal{L}(R)\#\big)^*\mathcal{F} \cap \mathcal{F}\big(\#\mathcal{R}(R)\big)^*$ we know that $w \notin \big(\mathcal{L}(R)\#\big)^*\mathcal{F}$ or $w \notin \mathcal{F}\big(\#\mathcal{R}(R)\big)^*$. If $w \notin \mathcal{F}\big(\#\mathcal{R}(R)\big)^*$ then $w \in \big(\mathcal{L}(R)\#\big)^+\mathcal{F}\big(\#\mathcal{R}(R)\big)^*$. But then cancellation of the first two blocks in w yields a shorter word $w' \in L(R, \mathcal{F})$. Moreover the number of blocks of w' which belong to $\mathcal{F}$ and which are preceded by an even number of blocks equals the number of blocks in w with these properties. According to Lemma 3.4 this implies $d_G(w') = d_G(w)$. Thus w is not in the support of d_G. Analogously if $w \notin \big(\mathcal{L}(R)\#\big)^*\mathcal{F}$ we can

cancel the last two blocks to obtain a shorter word w' with the same ambiguity as w which implies that w is not in the support of d_G in this case either. Thus $(L(R,\mathcal{F})\setminus \mathit{val}(R,\mathcal{F}))\cap \mathit{support}_G = \emptyset$. But $\mathit{support}_G \subseteq L(R,\mathcal{F})$. Hence $\mathit{support}_G \subseteq \mathit{val}(R,\mathcal{F})$. □

Note that in the proof above we do neither state that the word w' obtained from the cancellation of a block pair is in $\mathit{support}_G$ nor that it is in $\mathit{val}(R,\mathcal{F})$. But since $w' \in L(R,\mathcal{F})$ we can iterate the cancellation of block pairs either from left or right until eventually a word in $\mathit{val}(R,\mathcal{F})$ with the same ambiguity as the original word is reached. (For the word w in Example 3.5 this procedure would yield $a^3\#a^6\#a^{12}$.) Since w has a finite number of blocks such an iteration terminates.

If we apply Theorem 3.8 to the language $L(R,\Sigma^*)$ of Example 3.5 we see that the support of each canonical grammar G for this language only contains words where the number of a's is doubled from block to block. That is the shortest word with ambiguity $i+1$ is $a^{2^0}\#a^{2^1}\#\cdots\#a^{2^{2^i}}$ for an arbitrary $i \in \mathbb{N}$. Since the length grows exponentially with the ambiguity we see that $\hat{d}_G$ is logarithmic for each canonical context-free grammar generating $L(R,\Sigma^*)$. It can be shown that $L(R,\Sigma^*)$ is even inherently ambiguous of logarithmic degree. A proof for the existence of an inherent logarithmic ambiguity function can be found in [16], actually for a very similar language. The result there is slightly stronger, since by a reordering of the blocks a *linear* context-free language could be achieved. The reordering used there resembles the one in [2] to show that recursively enumerable sets can be characterised by the homomorphic image of the intersection of two linear context-free languages. Moreover by the example in [16] it can be shown that there is a regular trace language with a logarithmic ambiguity degree, which has been observed in [3].

How can we get a divergent ambiguity function with a sublogarithmic growth rate? One trial may be to force an even stronger growth of the length of related blocks. But this approach doesn't work as the following lemma shows:

Lemma 3.9. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation and $\mathcal{L}(R)$ a context-free language with the pumping-constant n. Then*

$$\forall x,y \in \Sigma^* : \Big(\big((x,y)\in R \wedge \forall z \in \Sigma^{<|y|} : (x,z)\notin R\big) \Rightarrow n(|x|+1) \geq |y|\Big).$$

Proof. We prove this statement by induction on $|y|$. For $|y| \leq n$ it is trivial. Now assume for some $m \leq n$ the statement holds for all $y \in \Sigma^{\leq m}$. Let $y \in \Sigma^{m+1}$. For each $x \in \Sigma^*$ we have to check the implication stated above. The nontrivial case is the one where the left-hand side of the statement is satisfied. In this case $x\#y^R \in \mathcal{L}(R)$ and we can mark the rightmost $|y|$ symbols of this word. According to Ogden's Lemma [11, 8] we can pump down $x\#y^R$ into a word of the form $x'\#y' \in \mathcal{L}(R)$. Now $|y'| < |y| \leq |y'|+n$. Due to the minimality of y we obtain that $|x'|+1 \leq |x|$. Hence we finally get:

$$|y| = |y| - |y'| + |y'| \leq n + |y'| \leq n + n(|x'|+1) \leq n + n|x| = n(|x|+1).$$

□

Since $2|x| \geq |x| + 1$ for $x \neq \varepsilon$ Lemma 3.9 immediately implies:

$$\forall x, y \in \Sigma^+ : \Big(\big((x, y) \in R \wedge \forall z \in \Sigma^{<|y|} : (x, z) \notin R \big) \Rightarrow 2n|x| \geq |y| \Big).$$

Hence we cannot force consecutive blocks to grow faster than by a constant factor, except for the very first step. Therefore sublogarithmic ambiguity cannot be obtained by this method. Obviously we can prove a version of Lemma 3.9 with $\mathcal{L}(R)$ replaced by $\mathcal{R}(R)$ in an analogous way.

4 Valid Computations

In example 3.5 the language for the free blocks is Σ^*. Therefore no candidate for the free block can be excluded in this case. As we have seen there is no hope to achieve sublogarithmic ambiguity just by increasing the growth rate of the blocks any further.

The new idea is to find an unambiguous context-free relation R and an unambiguous context-free language $\mathcal{F}$ such that in an infinite chain of words $w_0, w_1, \ldots$ such that $(w_i, w_{i+1}) \in R$ for each $i \in \mathbb{N}$ there are infinitely many words with even index belonging to the free block language $\mathcal{F}$. But with rising index the blocks in $\mathcal{F}$ occur less frequent.

Let M be a Turing machine. For each $\mathcal{F} \subseteq ID_M$ the words in $val(\vdash_M, \mathcal{F})$ represent computations which start and end in configurations belonging to $\mathcal{F}$. It is easily seen that $\vdash_M$ is an unambiguous context-free relation for each Turing machine M (even if M is nondeterministic). In fact $\mathcal{L}(\vdash_M)$ and $\mathcal{R}(\vdash_M)$ are even deterministic and linear context-free languages. Therefore by application of Theorem 3.8 we obtain:

Corollary 4.1. *Let M be a Turing machine and let $\mathcal{F} \subseteq ID_M$ be an unambiguous context-free language. Then $L(\vdash_M, \mathcal{F})$ is a block correlation language. Moreover if G is a canonical grammar generating $L(\vdash_M, \mathcal{F})$ then*

$$support_G \subseteq val(\vdash_M, \mathcal{F}).$$

Even though $L(\vdash_M, \mathcal{F})$ is a large superset of $val(\vdash_M, \mathcal{F})$ we don't need to care for the words in $L(\vdash_M, \mathcal{F}) \setminus val(\vdash_M, \mathcal{F})$ since they don't contribute to the ambiguity function of G. Therefore Corollary 4.1 provides a strong tool to design ambiguity functions. For instance let M be a Turing machine and $\mathcal{F} \subseteq ID_M$ the set of configurations of M where M is in the initial state. Let G be a canonical grammar generating $L(\vdash_M, \mathcal{F})$. Then only the words in $val(\vdash_M, \mathcal{F})$ are relevant for the computation of the ambiguity function, i.e., the words representing computations which start and end in configurations containing the initial state of M. The ambiguity of such a word is just the number of occurrences of the initial state in w at positions preceded by an even number of configurations. By the use of $\mathcal{F}$ we can induce an additional unambiguous context-free constraint on the initial configuration.

5 The Design of Slow Divergent Ambiguity Functions

Now we construct suitable Turing machines by the use of Corollary 4.1.

Lemma 5.1. *Let $f : \mathbb{N} \to \mathbb{N}$ be a computable divergent total non-decreasing function Then there is a Turing machine M and an unambiguous context-free set $\mathcal{F} \subseteq ID_M$ with the following properties: For each $n \in \mathbb{N}\backslash\{0,1\}$ their is a word $w \in val(\vdash_M, \mathcal{F})$ which contains n occurrences of configurations in $\mathcal{F}$ and the shortest word with this property has a length of at least $f(n)$. Moreover each occurrence of a configuration in $\mathcal{F}$ is preceded by an even number of configurations.*

Proof. Let $g : \mathbb{N} \to \mathbb{N}$ be defined by $g(n) := f(n+2)$ for each $n \in \mathbb{N}$. Then g is obviously a computable divergent total non-decreasing function. Let M' be a Turing machine which computes g. Without loss of generality we assume that $\{0,1\}$ is the input alphabet of M' and that non negative integers are encoded binary. Let q_0 be the initial state of M. We further assume that the state sets of M and M' are disjoint. The set $\mathcal{F}$ contains all the configurations of the form $q_0 q_0' n \$ n^R$. Here \$ is a tape symbol of M which is not a tape symbol of M' and q_0' is the initial state of M' and at the same time a tape symbol of M. Finally n is a binary encoded non negative integer. For convenience we identify binary encodings and the corresponding non-negative integers. Note that $\mathcal{F}$ is an unambiguous context-free language. The machine M never corrupts the initial format, i.e., if $\tau \in ID_M$ is reached from a configuration in $\mathcal{F}$ then by erasing the state of M from τ we obtain a string of the form $u_1 \$ u_2 \ \in ID_{M'} \$ \mathbb{N}$, here $\mathbb{N}$ means the set of binary encodings of non negative integers. We refer to u_1 and u_2 by calling them the *first* or *second segment* of τ. Note that u_1 contains a state of M'. The string obtained from u_1 by erasing this state is called the *tape* of the first segment. Let M be in the configuration $q_0 q_0' n_1 \$ n_1^R \in \mathcal{F}$. Then M goes through the following infinite loop:

1. Switch to a state $q_1 \neq q_o$ which starts the simulation of M'.
2. Simulate M' on the first segment until it halts. This eventually happens since the function g is computed by M' is total.
3. Decrement the tape of the first segment stepwise until it is 0. This is an idle loop (which loops $g(n_1)$ times when step 3 is called for the first time.)
4. Increment the second segment in its reverse coding (which yield $(n_1+1)^R$ when step 4 is called for the first time.)
5. Overwrite the first segment by $q_0' n$, where n is the reversal of the second segment, ($q_0' n = q_0'(n_1+1)$ when step 5 is called for the first time), place the head at the position of q_0'. If the last time the machine M was in q_0 is an odd number of steps ago enter q_0 immediately, otherwise wait one step before entering q_0. The parity of a step can easily be stored within the finite control of the Turing machine M. This action returns M into the initial situation. Thus it performs a kind of "**goto** 1" command.

Moreover we require that the Turing machine M is programmed in such a way that it does not enter the state q_0 except for the cases where this is explicitly mentioned above.

If a word $w \in val(\vdash_M, \mathcal{F})$ contains n occurrences of configurations in $\mathcal{F}$ for some $n \in \mathbb{N} \setminus \{0,1\}$ then steps 1 to 5 have been called at least $n-1$ times each. The value computed in the last call of step 2 was $g(n_1+n-2)$, where n_1 is the argument for which g is computed the first time. Since g is non-decreasing we have $g(n_1+n-2) \geq g(n-2) = f(n)$, and the machine needs at least $g(n_1+n-2) \geq f(n)$ steps in the idle loop executed in the last call of point 3 which is denoted in w. Hence w contain at least $f(n)$ many configuration each of which requires at least one symbol to be denoted.

Finally since the first configuration of a word in $val(\vdash_M, \mathcal{F})$ is in $\mathcal{F}$ and M always makes an even number of steps before reentering a configuration in $\mathcal{F}$ each of these configurations are preceded by an even number of configurations. This completes the argument. □

Note that M started on a configuration in $\mathcal{F}$ runs forever and passes an infinite number of times through a configuration in $\mathcal{F}$. The set $val(\vdash_M, \mathcal{F})$ contains finite infixes of infinite runs of M.

The estimation in the previous proof is rather wasteful, but simple to understand. Since we are not looking for a result on the density of ambiguity functions here, we can afford to use such a rough estimation.

Theorem 5.2. *Let $f : \mathbb{N} \to \mathbb{N}$ be a computable divergent total non-decreasing function then there is a context-free grammar such that for each $n \in \mathbb{N} \setminus \{0,1\}$ a shortest word with at least n derivation tree has length at least $f(n)$.*

Proof. By Lemma 5.1 there is a Turing machine M and an unambiguous context-free language $\mathcal{F}$ such that a shortest word in $val(\vdash_M, \mathcal{F})$ with n occurrences of a configuration in $\mathcal{F}$ has length at least $f(n)$ for $n \in \mathbb{N} \setminus \{0,1\}$. Moreover each of these occurrences is preceded by an even number of configurations. Therefore according to Lemma 3.4 these words have n derivation trees in a canonical grammar generating $L(\vdash_M, \mathcal{F})$ and by Corollary 4.1 we do not need to consider other words in $L(\vdash_M, \mathcal{F})$. □

Now we can guarantee that the length of a shortest word with ambiguity $n \in \mathbb{N} \setminus \{0,1\}$ is larger than $f(n)$ where f belongs to a set of function which allow huge growth rates. While we have considered the word length as a function of the ambiguity, the ambiguity function considers ambiguity as a function of the word length. Thus if the shortest word with ambiguity at least n exceeds a length of $f(n)$ for a given grammar G then roughly speaking $\hat{d}_G$ falls below f^{-1}. But we have to take care of non injective functions which technically does not have an inverse. Therefore we define:

Definition 5.3. *Let $f : \mathbb{N} \to \mathbb{N}$ be a divergent total non-decreasing function. Then:*

$$f^{-1}(n) := \max\left\{x \in \mathbb{N} \mid \ f(x) = \min\{y \in f(\mathbb{N}) \mid y \geq n\}\right\}.$$

It can be easily shown that $(f^{-1})^{-1} = f$ holds for each divergent total non-decreasing function. Now let f be a computable divergent total non-decreasing

function. Then f^{-1} is as well a computable divergent total non-decreasing function. Thus according to Theorem 5.2 there is a context-free grammar G such that for each $n \in \mathbb{N} \setminus \{0,1\}$ a shortest word with ambiguity of at least n has length at least $f^{-1}(n)$. But one can easily verify that this translates into $\hat{d}_G(n) \leq (f^{-1})^{-1}(n) = f(n)$ for all $n \geq f^{-1}(1)$. Now it is possible to construct a context-free grammar G' for $L(G) \setminus \Sigma^{\leq f^{-1}(1)}$ from G such that $\hat{d}_{G'}(n) = \hat{d}_G(n)$ for all $n > f^{-1}(1)$. Obviously $\hat{d}_{G'}(n) = 0$ for each $n \leq f^{-1}(1)$. Therefore $\hat{d}_{G'}(n) \leq f(n)$ for each $n \in \mathbb{N}$. Obviously G' has a well defined ambiguity power series. Moreover we can assume that G' is reduced. Hence G' is cycle-free. This leads us to:

Theorem 5.4. *If f is a computable divergent total non-decreasing function then there is a cycle-free context-free grammar G such that $\hat{d}_G$ is a divergent function satisfying $\hat{d}_G(n) \leq f(n)$ for all $n \in \mathbb{N}$.*

By the use of Theorem 1.1 and Theorem 5.4 we immediately obtain:

Theorem 5.5. *If f is a computable divergent total non-decreasing function then there is a context-free language L such that L has a divergent inherent ambiguity function $\hat{d}_L$ such that $\hat{d}_L(n) \leq f(n)$ for all $n \in \mathbb{N}$.*

6 Conclusion

We have seen that for each computable divergent total non-decreasing function there is a divergent inherent ambiguity functions which fall below f. We have not examined which functions are indeed ambiguity functions. Seemingly there are no substantial gaps below linear ambiguity, in contrast to the gap between exponential and polynomially bounded ambiguity. But how can we characterise the "density" of ambiguity functions formally? To examine this question one should improve the estimation in this paper. There is no need to use single steps of Turing machines as a means of computation. Instead we can allow unambiguous context-free relations to perform single steps. Clearly for the computational power this is unimportant but it provides more control over the length of the computations, which is crucial to control the ambiguity in our construction.

A characterisation of the set of inherent ambiguity functions is still a challenging problem, not only for sublogarithmic, but also for the whole class of context-free languages with polynomially bounded ambiguity.

By the result of this paper it is obvious that for each context-free grammar G_1 with a divergent ambiguity function we can find a not necessarily equivalent context-free grammar G_2 with a substantially lower ambiguity function. Substantially lower here means that for any $c \in \mathbb{N}$ we have $d_{G_1}(n) \geq d_{G_2}(cn)$ for all but finitely many $n \in \mathbb{N}$. Is there a context-free language L such that a similar property holds for all the context-free grammars generating L? In this case L would not have an inherent ambiguity function. Are there in fact context-free languages which does not have an inherent ambiguity function?

References

1. A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation, and Compiling*, volume I. Prentice-Hall, Englewood Cliffs, N.J., 1972.
2. B. S. Baker and R. V. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences*, 8:315–332, 1974.
3. A. Bertoni, C. Choffrut, M. Goldwurm, and V. Lonati. On the number of occurrences of a symbol in words of regular languages. *Theoretical Computer Science*, 302:431–456, 2002.
4. A. Bertoni, M. Goldwurm, and M. Santini. Random generation and approximate counting of ambiguously described combinatorial structures. In H. Reichel and S. Tison, editors, *Proceedings of the STACS 2000*, number 1770 in Lecture Notes in Computer Science, pages 567–580, Berlin-Heidelberg-New York, 2000. Springer.
5. J. Crestin. Un langage non ambigu dont le carré est d'ambiguité non bornée. In M. Nivat, editor, *Automata, Languages and Programming*, pages 377–390. Amsterdam, North-Holland, 1973.
6. J. C. Earley. *An efficient context-free parsing algorithm.* PhD thesis, Carnegie-Mellon University, 1968.
7. S. Ginsburg, S. A. Greibach, and M. A. Harrison. One-way stack automata. *Journal of the Association for Computing Machinery*, 14:389–418, 1967.
8. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.
9. H. Maurer. The existence of context-free languages which are inherently ambiguous of any degree. Research series, Department of Mathematics, University of Calgary, 1968.
10. M. Naji. Grad der Mehrdeutigkeit kontextfreier Grammatiken und Sprachen, 1998. Diplomarbeit, FB Informatik, Johann–Wolfgang–Goethe–Universität.
11. W. Ogden. A helpful result for proving inherent ambiguity. *Mathematical Systems Theory*, 2(3):191–194, 1968.
12. R. J. Parikh. Language–generating devices. In *Quarterly Progress Report*, volume 60, pages 199–212. Research Laboratory of Electronics, M.I.T, 1961.
13. P. Rossmanith and W. Rytter. Observations on $\log(n)$ time parallel recognition of unambiguous cfl's. *Information Processing Letters*, 44:267–272, 1992.
14. E. Shamir. Some inherently ambiguous context-free languages. *Information and Control*, 18:355–363, 1971.
15. K. Wich. Exponential ambiguity of context-free grammars. In G. Rozenberg and W. Thomas, editors, *Proceedings of the DLT, 1999*, pages 125–138. World Scientific, Singapore, 2000.
16. K. Wich. Sublinear ambiguity. In M. Nielsen and B. Rovan, editors, *Proceedings of the MFCS 2000*, number 1893 in Lecture Notes in Computer Science, pages 690–698, Berlin-Heidelberg-New York, 2000. Springer.
17. K. Wich. Characterization of context-free languages with polynomially bounded ambiguity. In J. Sgall, A. Pultr, and P. Kolman, editors, *Proceedings of the MFCS 2001, Mariánské Lázně, (Czech Republic), 2001*, number 2136 in Lecture Notes in Computer Science, pages 703–714, Berlin-Heidelberg-New York, 2001. Springer.
18. K. Wich. Universal inherence of cycle-free context-free ambiguity functions. In P. Widmayer et al., editors, *Proceedings of the ICALP 2002*, number 2380 in Lecture Notes in Computer Science, pages 669–680, Berlin-Heidelberg-New York, 2002. Springer.