Characterization of context-free languages with polynomially bounded ambiguity

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

Institut für Informatik, Universität Stuttgart,
Breitwiesenstr. 20-22, 70565 Stuttgart.

Abstract. We prove that the class of context-free languages with polynomially bounded ambiguity (*PCFL*) is the closure of the class of unambiguous languages (*UCFL*) under projections which deletes Parikh bounded symbols only. A symbol a is Parikh bounded in a language L if there is a constant c such that no word of L contains more than c occurrences of a. Furthermore *PCFL* is closed under the formally stronger operation of Parikh bounded substitution, i.e., a substitution which is the identity for non Parikh bounded symbols. Finally we prove that the closure of *UCFL* under union and concatenation is a proper subset of *PCFL*.

1 Introduction

A context-free (for short cf) grammar is unambiguous if each word has at most one derivation tree. Otherwise it is ambiguous. There are cf languages which cannot be generated by unambiguous cf grammars [8]. These languages are called (inherently) ambiguous. A cf grammar has a k-bounded ambiguity if no word has more than k derivation trees. It is k-ambiguous if it is k-bounded but not $(k-1)$-bounded. A language L is ambiguous of degree k if it is generated by a k-ambiguous grammar, but it cannot be generated by a $(k-1)$-bounded grammar. There are examples for k-ambiguous languages for each $k \in \mathbb{N}$ [6]. But even languages with infinite degree of ambiguity exist [3]. We can distinguish these languages by the asymptotic behavior of their ambiguity with respect to the length of the words. The ambiguity function of a cf grammar is a monotonous function which yields the maximal number of derivation trees for words up to a given length. By an undecidable criterion which is related to the Pumping Lemma it can be shown that each cycle-free cf grammar is either exponentially ambiguous or its ambiguity is bounded by a polynomial [10, 11]. Consequently there is a gap between polynomial and exponential ambiguity. We introduce (inherent) asymptotic ambiguity for languages similarly as above. There are cf languages with exponential ambiguity and with $\Theta(n^k)$ ambiguity for each $k \in \mathbb{N}$ [7]. Even a linear cf language with logarithmic ambiguity is known [12].

A symbol a is Parikh bounded in a language L if there is a constant c such that no word of L contains more than c occurrences of a. Thus c is an upper

bound for the corresponding component in the Parikh mapping. A substitution is called Parikh bounded if it is the identity for non Parikh bounded symbols. It is easily seen that *PCFL* is closed under Parikh bounded substitution. We prove that *PCFL* is the closure of unambiguous context-free languages (*UCFL*) under Parikh bounded projection, i.e., a substitution which deletes Parikh bounded symbols only. The construction is effective and can be computed in polynomial time. Note that Parikh bounded projection is a special case of Parikh bounded substitution for *UCFL* languages since singletons are unambiguous. Finally we prove that the closure of *UCFL* under union and concatenation ($UCFL[\cdot, \cup]$) is a proper subset of *PCFL*. This class is interesting since each language in $UCFL[\cdot, \cup]$ is generated by a cf grammar with quadratic Earley parsing time.

2 Preliminaries

Let Σ be a finite alphabet. Let $u = a_1 \cdots a_n \in \Sigma^*$ be a word over Σ, where $a_i \in \Sigma$ for $1 \leq i \leq n$. The *length* of u is $|u| := n$. For $1 \leq i \leq n$ we define $u[i] = a_i$. The *empty word* is denoted by ε. Formal languages and their concatenation are defined as usual [1]. Let Σ and Γ be two alphabets. A *substitution* σ is the homomorphic extension of a mapping $\sigma : \Sigma \to 2^{\Gamma^*}$. For $a \in \Sigma$, $L \subseteq \Sigma^*$, and $\tilde{L} \subseteq \Gamma^*$, the *single symbol substitution* defined by $\sigma(a) = \tilde{L}$ and $\sigma(b) = \{b\}$ for each $b \in \Sigma \setminus \{a\}$ is denoted $[a/\tilde{L}]$. We write $L[a/\tilde{L}]$ for $[a/\tilde{L}](L)$. For $L \subseteq \Sigma^*$ and a substitution σ we define $\sigma(L) := \{u \mid u \in \sigma(w) \text{ for some } w \in L\}$. For $\Gamma \subseteq \Sigma$ the *projection* π_Γ is the homomorphism given by $\pi_\Gamma(b) = b$ for all $b \in \Gamma$ and $\pi_\Gamma(b) = \varepsilon$ for $b \in \Sigma \setminus \Gamma$. For each $w \in \Sigma^*, a \in \Sigma$, and $\Gamma \subseteq \Sigma$ we define $|w|_\Gamma := |\pi_\Gamma(w)|$ and $|w|_a := |w|_{\{a\}}$.

A *context-free grammar* is a triple $G = (V, P, S)$, where V is a finite alphabet, $P \subseteq V \times V^*$ is a finite set of *productions*, and $S \in V$ is the *start symbol*. $N_G := \{A \in V \mid A \times V^* \cap P \neq \emptyset\}$ is called the set of *nonterminals*. $\Sigma_G := V \setminus N_G$ is the set of *terminals*. A production $p = (A, \alpha)$ is denoted by $A \to \alpha$ or $(A \to \alpha)$. The *left-*, and *right-hand sides* of p are $\ell(p) := A$, and $r(p) := \alpha$, respectively. We call p an *ε-production* if $r(p) = \varepsilon$.

The usual way to introduce the ambiguity of a word w over Σ_G is by the number of its leftmost derivations, which are in one to one correspondence with the derivation trees for w. This correspondence does not hold for trees which contain nonterminals in their frontier. But in our consideration trees with this property, especially so called pumping trees, play a crucial rule. On the other hand we do not need the full term rewriting formalism to handle derivation trees. Therefore we use an intermediate formalism.

Let $X \in V \cup P$ and $\tau \in (V \cup P)^*$. The *root* of $X\tau$ is X if $X \in V$ and $\ell(X)$ if $X \in P$. The root of $X\tau$ is denoted by $\uparrow X\tau$. The projection $\pi_V(\tau)$ is the *yield* of τ denoted by $\downarrow\tau$. A *derivation tree* ρ is either an element of $V \cup \{p\, r(p) \mid p \in P\}$ or it can be decomposed as $\rho = \tau_1 \rho' \tau_2$ such that ρ' and $\tau_1\, (\uparrow\rho')\, \tau_2$ are derivation trees. The set of derivation trees of G is denoted by $\triangle(G)$. If ρ is a derivation tree then the *interface* of ρ is the pair $\updownarrow\rho := (\uparrow\rho, \downarrow\rho)$. An element from $\{1, \ldots, |\rho|\}$ is said to be a *node* of ρ. A node ν is an *internal node* if $\rho[\nu] \in P$ and we say that

the *P-label* of ν is $\rho[\nu]$. The node ν is a *leaf* if $\rho[\nu] \in V$. The *label of* ν is $\rho[\nu]$ if ν is a leaf and it is $\ell(\rho[\nu])$ if ν is an internal node. Note that $|\downarrow\rho|_\Gamma = |\rho|_\Gamma$ holds for each $\Gamma \subseteq V$. Thus $|\rho|_\Gamma$ is the number of leaves in ρ labeled with an element of Γ. A node μ is an *ancestor* of ν if $\rho = \tau_1 \cdots \tau_5$, $\mu = |\tau_1| + 1$, $\nu = |\tau_1\tau_2| + 1$, and $\tau_2\tau_3\tau_4$, $\tau_3 \in \triangle(G)$ for some $\tau_1, \ldots, \tau_5 \in (V \cup P)^*$. The node ν is a *descendant* of μ if μ is an ancestor of ν .The node μ is a *proper ancestor (descendant)* of ν if μ is an ancestor (a descendant) such that $\mu \neq \nu$. Let $\mathcal{P}$ be a property of nodes. The node μ is the *first ancestor* of ν with property $\mathcal{P}$ if no proper descendant of μ which satisfies $\mathcal{P}$ is an ancestor of ν. The *first common ancestor* of two nodes ν_1 and ν_2 is the first ancestor of ν_1 which is an ancestor of ν_2. A node ν is a son of a node μ if μ is the first proper ancestor of ν. A *tree format* Δ is a mapping that assigns to each cf grammar G a subset of $\triangle(G)$. The tree format defined by $\triangle(G)$ for each cf grammar G is $\triangle$. The set of *compressed derivation trees* is defined by $comp(\triangle(G)) := \{\pi_{V\cup P}(\rho) \mid \rho \in \triangle(G)\}$. Note that $\pi_{V\cup P}$ restricted to $\triangle(G)$ is a bijection from $\triangle(G)$ to $comp(\triangle(G))$. If $\downarrow\rho \in \Sigma^*$ then $\pi_{V\cup P}(\rho)$ coincides with the well known left-parse. The *standard derivation tree format* $\triangle^S_\Sigma$ is defined by $\triangle^S_\Sigma(G) := \{\rho \in \triangle(G) \mid \updownarrow\rho \in \{S\} \times \Sigma^*_G\}$. The *language generated by* G is $L(G) := \{\downarrow\rho \mid \rho \in \triangle^S_\Sigma(G)\}$. The set of *sentential forms* is $\zeta(G) := \{\downarrow\rho \mid \rho \in \triangle(G)\}$. *In the sequel the notions language and grammar represent context-free languages and context-free grammars, respectively.* A grammar $G = (V, P, S)$ is *cycle-free* if $\uparrow\rho = \downarrow\rho$ implies $\rho \in V$. It is *reduced* if either for each $p \in P$ there is a $\rho \in \triangle^S_\Sigma(G)$, such that p appears in ρ, or $P = \{S \to S\}$ in case $L(G) = \emptyset$. A reduced grammar $G = (V, P, S)$ is *strongly reduced* if either $L((V, P, A)) \neq \{\varepsilon\}$ for each $A \in N_G$ or $P = \{S \to \varepsilon\}$ in case $L(G) = \{\varepsilon\}$. For each grammar G there is an equivalent strongly reduced grammar which is not larger than G. If G is a grammar, then its canonical strongly reduced equivalent grammar is denoted by *red*(G). *If not stated otherwise* $G = (V, P, S)$ *is an implicitly "for all" quantified cycle-free strongly reduced grammar and* $N = N_G$, $\Sigma = \Sigma_G$, $A, B \in N$; $a, b, c \in \Sigma$; $X, Y \in V$. The *ambiguity function* $\mathrm{am}_G : \mathbb{N}_0 \to \mathbb{N}_0$ is defined by $\mathrm{am}_G(n) := \max\left\{|\{\rho \in \triangle^S_\Sigma(G) | \downarrow\rho = w\}| \,\middle|\, w \in \Sigma^* \wedge |w| \leq n\right\}$. The grammar G is *unambiguous* if $am_G(n) \leq 1$ for each $n \in \mathbb{N}$, it is of *polynomially bounded ambiguity* if $am_G(n) = \mathcal{O}(n^k)$ for some $k \in \mathbb{N}$, and it is *exponentially ambiguous* if $am_G(n) = 2^{\Omega(n)}$. The set of grammars with the corresponding asymptotic ambiguities are denoted by *UCFG*, *PCFG*, and *ECFG*, respectively. The class of languages generated by unambiguous grammars, and grammars with polynomially bounded ambiguity are called *UCFL*, and *PCFL*, respectively. The class of languages which require an exponentially ambiguous grammar to be generated is called *ECFL*. If X is a set of languages or grammars then $\overline{X}$ denotes the set of cycle-free strongly reduced cf grammars or cf languages which do not belong to X. Let Δ be a tree format. The set $\Delta(G)$ is unambiguous if for all $\rho_1, \rho_2 \in \Delta(G)$ we have $\updownarrow\rho_1 = \updownarrow\rho_2 \Rightarrow \rho_1 = \rho_2$. The set of grammars which are unambiguous w.r.t. Δ is defined by $U(\Delta) = \{G$ cycle-free and strongly reduced $\mid$ $\Delta(G)$ *is unambiguous*$\}$. Note that $U(\triangle) = UCFG = U(\triangle^S_\Sigma)$.

3 Closure properties of *PCFL*

Lemma 3.1. *PCFL is not closed under length preserving homomorphisms.*

Proof. Let $L := (\{a^i\$b^{i-1}c^j \mid i,j \geq 1\} \cup \{a^ib^{j-1}\$c^j \mid i,j \geq 1\})^*$ and $h : \{a,b,c,\$\}^* \to \{a,b,c\}^*$ be defined by $h(x) = x$ for $x \neq \$$ and $h(\$) = b$. It is easily seen that $L \in$ *UCFL*. But $h(L) = \{a^ib^jc^k \mid i = j \vee j = k\}^*$ is an inherently exponentially ambiguous language [7].

Definition 3.2. *We extend* $\mathbb{N}$ *to a complete lattice by adding a maximal element* ω*. Thus each subset of* $\mathbb{N} \cup \{\omega\}$ *has a supremum. Let* L *be a language over* Σ*. Let* $\Gamma \subseteq \Sigma$*. The Parikh supremum is defined by* $\Psi_{\sup} : 2^{\Sigma^*} \to (2^\Sigma \to \mathbb{N} \cup \{\omega\})$ *where* $\Psi_{\sup}(L)(\Gamma) := \sup\{|w|_\Gamma \mid w \in L\}$*. For each grammar* $G = (V,P,S)$ *and each* $\Gamma \subseteq V$*, the set* Γ *is called Parikh bounded (Pb) if* $\Psi_{\sup}(G)(\Gamma) := \Psi_{\sup}(\zeta(G))(\Gamma) < \omega$*. A symbol* $X \in V$ *is Pb if* $\{X\}$ *is Pb.*

The Parikh supremum is the maximum number of Γ symbols which can occur in a word of L. We write $\Psi_{\sup}(L)(a)$ for $\Psi_{\sup}(L)(\{a\})$. $\Psi_{\sup}(L)(a)$ can be considered as the supremum of the corresponding component over all Parikh vectors for L. See [9] for the definition of Parikh vectors.

Definition 3.3. *Let* Σ *and* Γ *be finite alphabets. A substitution* $\sigma : \Sigma^* \to 2^{\Gamma^*}$ *is* Parikh bounded *for a language* $L \subseteq \Sigma^*$ *if for each* $a \in \Sigma$ *which is not Parikh bounded* $\sigma(a) = \{a\}$ *holds. The projection* $\pi_\Gamma(L)$ *is a* Parikh bounded projection *if* $\Sigma \setminus \Gamma$ *contains Parikh bounded symbols only.*

Lemma 3.4. *The language class PCFL is closed under Parikh bounded substitution, Parikh bounded projection, concatenation and union.*

Proof. Let $L_1, L_2 \in$ *PCFL* generated by $G_1, G_2 \in$ *PCFG* with disjoint sets of nonterminals. Let a be a Parikh bounded symbol in L_1 with the Parikh supremum k. We can construct a grammar for $L_1[a/L_2]$ by replacing each occurrence of an a by the start symbol of G_2. Then each word of $L_1[a/L_2]$ consists of a word from L_1, where all occurrences of a in w (at most k many) have been replaced by words of L_2. Each inserted word is specified by the positions where it starts and ends. Therefore the number of possible factorizations is bounded by $\mathcal{O}(n^{2k})$. Since ambiguity functions are monotonous the ambiguity of a word of length n in $L_1[a/L_2]$ is bounded by $\mathcal{O}(am_{G_1}(n) \cdot (am_{G_2}(n))^k \cdot n^{2k})$, which is polynomially bounded. Each Parikh bounded substitution can be written as a sequence of single symbol substitutions. For the Parikh bounded projection $\pi_{\Sigma_{G_1} \setminus \{a\}}(L_1)$, concatenation, and union we obtain the better upper bounds $\mathcal{O}(am_{G_1}(n) \cdot n^k)$, $\mathcal{O}(am_{G_1}(n) \cdot am_{G_2}(n) \cdot n)$, and $\mathcal{O}(am_{G_1}(n) + am_{G_2}(n))$, respectively. □

4 Types of symbols, productions, trees and tree formats

Definition 4.1. *We say* A *derives* B*, denoted by* $A \vdash B$*, if there exists* $\rho \in \triangle(G)$ *such that* $\uparrow\rho = A$ *and* $|\rho|_B > 0$*. We define an equivalence relation* $\equiv$ *by:* $A \equiv B$ *if* $A \vdash B \ \wedge \ B \vdash A$*. The equivalence class of* A *is denoted by* $[A]$*.*

Lemma 4.2. *Let $X \in V$, $\Gamma_X = \{A \in N \mid A \vdash X\}$ and $\Psi := \Psi_{\sup}((V, P, S))$. Then $\Psi(\Gamma_X) = \Psi(X)$.*

Proof. Since each Element of Γ_X can generate at least one occurrence of X, we obtain $\Psi(\Gamma_X) \leq \Psi(X)$. On the other hand $X \in \Gamma_X$. Therefore $\Psi(X) \leq \Psi(\Gamma_X)$. Thus the claim follows. □

Definition 4.3. *A production $(A \to \alpha) \in P$ is called*

- pumping production *if* $|\alpha|_{[A]} > 0$.
- descending production *if it is not a pumping production.*
- bounded production *if it is descending and A is Parikh bounded.*
- unbounded production *if it is not bounded.*

The sets of pumping, descending, bounded, and unbounded productions are denoted by, $P_=$, $P_<$, $P_{<\omega}$, and P_ω, respectively.

On a path from a leaf to the root a pumping production can occur arbitrarily often, while a descending production can occur only once. Hence the maximal number of occurrences of a descending production in an arbitrary derivation tree is bounded by the Parikh supremum of its left-hand side. Bounded productions will play a particularly crucial role in the sequel.

Definition 4.4. *A derivation tree $\rho \in \triangle(G)$ is a pumping tree if $|\rho|_{\uparrow\rho} > 0$, it is a partial pumping tree if $|\rho|_{[\uparrow\rho]} > 0$, and it is an unbounded production tree if it doesn't contain a bounded production, i.e., $|\rho|_{P_{<\omega}} = 0$. The set of pumping, partial pumping,and unbounded production trees, are denoted by $\mathbin{\vartriangle\!\!\!\!\!\!\vartriangle}(G)$, $\mathbin{\vartriangle\!\!\!\!\!\!\vartriangle\!\!\!\!\!\!\vartriangle}(G)$, and $\mathbin{\vartriangle\!\!\!\!\!\!\vartriangle\!\!\!\!\!\!\vartriangle\!\!\!\!\!\!\vartriangle}(G)$, respectively. The corresponding tree formats are $\mathbin{\vartriangle\!\!\!\!\!\!\vartriangle}$, $\mathbin{\vartriangle\!\!\!\!\!\!\vartriangle\!\!\!\!\!\!\vartriangle}$, and $\mathbin{\vartriangle\!\!\!\!\!\!\vartriangle\!\!\!\!\!\!\vartriangle\!\!\!\!\!\!\vartriangle}$. For an arbitrary tree format Δ the tree formats Δ^ω and $\Delta^{<\omega}$ are defined by the restriction of Δ to trees with non Parikh bounded and Parikh bounded roots, respectively.*

Note that each tree obtained from a pumping tree by cutting off some subtrees is a partial pumping tree.

Definition 4.5. *A symbol X is pumpable, if there is a $\rho \in \mathbin{\vartriangle\!\!\!\!\!\!\vartriangle}(G)$ such that $|\rho|_X > |\uparrow\rho|_X$.*

Lemma 4.6. *A symbol X is pumpable if and only if it is not Parikh bounded.*

Proof. If X is pumpable then by definition we have a pumping tree, which pumps occurrences of X. Hence X is not Parikh bounded. If X is not Parikh bounded, we choose a word with sufficiently many occurrences of X. We mark them according to Ogden's iteration Lemma [1] for context-free grammars, and obtain a pumping tree which is appropriate to show the pumpability of X. □

In the sequel we will use the notion pumpable symbol synonymous for non Parikh bounded symbols, without explicitly referencing the previous lemma.

5 Computation of Parikh Suprema

There is a polynomial time algorithm which takes a pair, consisting of a reduced grammar $G = (V, P, S)$ and an alphabet $\Gamma \subseteq V$ (in a binary encoding), as the input and computes $\Psi_{\sup}(G)(\Gamma)$. The algorithm works as follows: First we compute *directlyPumpable* $:= \{X \mid \exists p \in P_{=} : |r(p)|_X > |\ell(p)|_X\}$. Then we compute *Pumpable* $:= \{Y \mid X \in$ *directlyPumpable* $\wedge\ X \vdash Y\}$. It is easily seen that *Pumpable* is the set of pumpable symbols. The set $\Gamma \subseteq V$ is not Parikh bounded if and only if at least one $X \in \Gamma$ is not Parikh bounded. Hence we can determine by the algorithm above whether Γ is Parikh bounded or not. If Γ is not Parikh bounded then $\Psi_{\sup}(G)(\Gamma) = \infty$. Otherwise we proceed as follows: First we construct G' by erasing all productions p where $\ell(p)$ is a pumpable symbol and by erasing all occurrences of pumpable symbols in right-hand sides of productions. By Lemma 4.2 no Parikh bounded symbol can ever be generated by a pumpable symbol. Hence G' is reduced. Again by Lemma 4.2 the Parikh bound is an invariant w.r.t. the equivalence $\equiv$. Hence we can replace each occurrence of a symbol by its equivalence class and obtain grammar G''. The remaining pumping productions all have the form $[X] \to [X]$ and can be eliminated to obtain G'''. It is easily seen that $\Psi_{\sup}(G)(\Gamma) = \Psi_{\sup}(G''')(\Gamma')$ for $\Gamma' := \{[X] \mid X \in \Gamma\}$. Since the grammar G''' has descending productions only $\triangle(G''')$ is finite. Obviously we can explore all the trees in $\triangle(G''')$ to compute the Parikh supremum in exponential time. But we can use dynamic programming methods to compute the Parikh bound efficiently. Let $G''' = (V, P, S)$. We can allow productions as start symbols with the semantic that this production has to be the first one in each derivation tree. Obviously this definition does not increase the generative power of cf grammars. We compute for each $\mu \in P \cup V$ the Parikh supremum of Γ' in the grammar (V, P, μ). For $\mu \in \Sigma_{G'''}$ the task is trivial. Starting with the terminals we compute $\Psi_{\sup}(G''')(\Gamma')$ bottom up by the use of the equation:

$$\Psi_{\sup}((V,P,X))(\Gamma') = \begin{cases} \max\{\Psi_{\sup}((V,P,\nu))(\Gamma') \mid \nu \in P \cap X \times V^*\} \text{ if } X \in V \\ \sum_{i=1}^{k} \Psi_{\sup}((V,P,X_k))(\Gamma') \text{ if } X = A \to X_1 \cdots X_k \in P \end{cases}$$

6 Characterization of *PCFL*

Theorem 6.1 ([11]). $\overline{U(\mathbb{A})} \subseteq$ *ECFG.*

The idea of the proof is as follows. If $G \notin U(\mathbb{A})$ then there exist two different pumping trees ρ_1, ρ_2 with a common interface. Thus we can construct trees which contain chains of n such pumping trees. Their interfaces do not depend on the chosen pumping trees at an arbitrary position in the chain. Hence we have at least 2^n ways to construct derivation trees of that kind, all having the same interface. To be sure that this idea yields exponential ambiguity we need the fact that the grammar is *cycle-free* and *strongly reduced*. Even $U(\mathbb{A}) \subseteq$ *PCFG* and the undecidability of $U(\mathbb{A})$ have been proved in [11]. But we will use Theorem 6.1 only. Included in our final result we will get an alternative proof for $U(\mathbb{A}) \subseteq$ *PCFG*. As an immediate consequence of the definitions we obtain the following Lemma:

Lemma 6.2. *Let Δ_1 and Δ_2 be tree formats. Then* $(\forall G : \Delta_1(G) \subseteq \Delta_2(G)) \Rightarrow U(\Delta_2) \subseteq U(\Delta_1)$.

Lemma 6.3. *Let Δ, Δ_1, and Δ_2 be tree formats such that:* $\forall G : \Delta(G) \subseteq \Delta_1(G) \cup \Delta_2(G)$ *and* $\{\updownarrow\rho \mid \rho \in \Delta_1(G)\} \cap \{\updownarrow\rho \mid \rho \in \Delta_2(G)\} = \emptyset$. *Then* $U(\Delta_1) \cap U(\Delta_2) \subseteq U(\Delta)$.

Proof. Let $G \in U(\Delta_1) \cap U(\Delta_2)$ and $\rho_1, \rho_2 \in \Delta(G)$ such that $\updownarrow\rho_1 = \updownarrow\rho_2$. In case $\rho_1 \in \Delta_1(G)$, the assumption $\rho_2 \in \Delta_2(G)$ would lead to a nonempty intersection of the interfaces of $\Delta_1(G)$ and $\Delta_2(G)$, contradicting our requirements. Hence $\rho_2 \in \Delta_1(G)$ too. Thus $\rho_1 = \rho_2$ follows from $G \in U(\Delta_1)$. The assumption $\rho_1 \in \Delta_2(G)$ yields in an analogous manner $\rho_1 = \rho_2$. Hence $G \in U(\Delta)$. □

Lemma 6.4. $U(\mathbb{\Delta}) = U(\mathbb{\Delta}_{\amalg})$.

Proof. Since $\mathbb{\Delta}(G) \subseteq \mathbb{\Delta}_{\amalg}(G)$ for each grammar G the inclusion $U(\mathbb{\Delta}_{\amalg}) \subseteq U(\mathbb{\Delta})$ is a consequence of Lemma 6.2. Let $G \in U(\mathbb{\Delta})$. Choose arbitrary $\rho_1, \rho_2 \in \mathbb{\Delta}_{\amalg}(G)$ such that $\updownarrow\rho_1 = \updownarrow\rho_2$. We have to show that this implies $\rho_1 = \rho_2$. By definition of $\mathbb{\Delta}_{\amalg}$ there is a $B \in N$ such that B is contained in the yield and $\uparrow\rho_1 \equiv B$. By definition of $\equiv$ this implies that there is a derivation tree $\rho = \tau_1(\uparrow\rho_1)\tau_2 \in \triangle(G)$ such that $\uparrow\rho = B$. Then we obtain that $\tau_1\rho_1\tau_2$ and $\tau_1\rho_2\tau_2$ are pumping trees with a common interface. Since $G \in U(\mathbb{\Delta})$ this implies $\tau_1\rho_1\tau_2 = \tau_1\rho_2\tau_2$. By left- and right cancellation we obtain $\rho_1 = \rho_2$. Thus $G \in U(\mathbb{\Delta}_{\amalg})$. □

Lemma 6.5. $U(\mathbb{\Delta}) \subseteq U(\triangle^\omega)$.

Proof. Let $G \in U(\mathbb{\Delta})$. Choose arbitrary $\rho_1, \rho_2 \in \triangle^\omega(G)$ such that $\updownarrow\rho_1 = \updownarrow\rho_2$. Then $X := \uparrow\rho_1$ is pumpable. Hence by definition there is a derivation tree $\rho = \tau_1 X \tau_2 \in \mathbb{\Delta}(G)$, such that $|\rho|_X - |\uparrow\rho|_X > 0$. Now either $X = \uparrow\rho$ then $|\tau_1\tau_2|_{\uparrow\rho} = |\tau_1(\uparrow\rho)\tau_2|_{\uparrow\rho} - |\uparrow\rho|_{\uparrow\rho} = |\rho|_X - |\uparrow\rho|_X > 0$ or $\uparrow\rho \neq X$ then $|\tau_1\tau_2|_{\uparrow\rho} = |\tau_1 X \tau_2|_{\uparrow\rho} = |\rho|_{\uparrow\rho} > 0$ since $\rho \in \mathbb{\Delta}(G)$. Hence in both cases $|\tau_1\tau_2|_{\uparrow\rho} > 0$. This implies $\tau_1\rho_1\tau_2$, $\tau_1\rho_2\tau_2 \in \mathbb{\Delta}(G)$. Now $G \in U(\mathbb{\Delta})$ implies $\tau_1\rho_1\tau_2 = \tau_1\rho_2\tau_2$. By left- and right cancellation we obtain $\rho_1 = \rho_2$. Hence $G \in U(\triangle^\omega)$. □

Lemma 6.6. *Let $\rho \in \triangle(G)$. If ρ contains a node ν labeled with a Pb symbol A then each ancestor of ν is labeled with a Pb symbol.*

Proof. If ν' is an ancestor of ν labeled with B then B generates A (i.e. $B \vdash A$). Thus by Lemma 4.2 we have $\Psi_{\mathrm{sup}}(B) \leq \Psi_{\mathrm{sup}}(A)$. Hence B is a Pb symbol. □

Lemma 6.7. *Let $\rho \in \triangle(G)$. Let ν_1 and ν_2 be two nodes labeled with a Pb symbol where neither one is an ancestor of the other. Then there is a node $\nu < \min\{\nu_1, \nu_2\}$ such that the P-label of ν is a bounded production.*

Proof. Let ν be the first common ancestor of ν_1 and ν_2. Then $\nu \leq \min\{\nu_1, \nu_2\}$. Since neither one is an ancestor of the other $\nu < \min\{\nu_1, \nu_2\}$. By Lemma 6.6 the node ν and two distinct sons of ν are labeled with Pb symbols. But among the descendants of a pumping production there is at most one Pb symbol. Therefore the P-label of ν must be a descending production. Thus we obtain that the P-label of ν is a bounded production. □

Corollary 6.8. *Let $\rho \in \triangle(G)$. If ρ contains more than one leaf labeled with a Pb symbol then $\rho \notin \mathbb{\triangle}(G)$.*

Proof. Let ν_1, ν_2 be two distinct leaves of ρ labeled with a Pb symbol . Obviously neither is an ancestor of the other. Hence by Lemma 6.7 the tree ρ contains a bounded production. □

Lemma 6.9. *Let $\rho \in \triangle^{<\omega}(G)$. If $|\rho|_{[\uparrow\rho]} = 0$ then $\rho \notin \mathbb{\triangle}(G)$.*

Proof. By definition $\uparrow\rho$ is a Pb symbol. Hence ρ contains a node ν, labeled with a symbol in $[\uparrow\rho]$, which has no descendants in $[\uparrow\rho]$. If $|\rho|_{[\uparrow\rho]} = 0$ then this node cannot be a leaf, i.e., there is a production p which is the P-label of ν. This production is descending by the choice of ν. Moreover $\ell(p) \in [\uparrow\rho]$ is Parikh bounded. Thus p is a bounded production, which implies $\rho \notin \mathbb{\triangle}(G)$. □

Immediately by Corollary 6.8 and Lemma 6.9 we obtain:

Lemma 6.10. *If $\rho \in \mathbb{\triangle}^{<\omega}(G)$ then ρ contains exactly one leaf labeled with a Pb symbol and this symbol is in $[\uparrow\rho]$.*

Lemma 6.11. $\mathbb{\triangle}^{<\omega}(G) = \mathbb{\pitchfork}^{<\omega}(G)$.

Proof. The inclusion $\mathbb{\triangle}^{<\omega}(G) \subseteq \mathbb{\pitchfork}^{<\omega}(G)$ is an immediate consequence of Lemma 6.10. Let ρ be an arbitrary element of $\mathbb{\pitchfork}^{<\omega}(G)$. Then ρ contains a leaf ν labeled with a symbol $A \in [\uparrow\rho]$. Since $\uparrow\rho$ is Pb it suffices to show that an arbitrary internal node μ of ρ must not have a bounded production as its P-label. First assume μ is an ancestor of ν labeled B. Then $\uparrow\rho \vdash B \vdash A \vdash \uparrow\rho$ since $A \in [\uparrow\rho]$. Hence $B \in [\uparrow\rho]$. But then the first ancestor of ν which is a son of μ is in $[\uparrow\rho]$ as well. This implies that the P-label of μ is a pumping production p. Thus p is not bounded. Now assume μ is no ancestor of ν, then the first common ancestor of μ and ν is a pumping production capable to pump the label of μ and again the P-label of μ must not be a bounded production. □

Theorem 6.12. *$PCFG \subseteq U(\mathbb{\triangle}) \subseteq U(\mathbb{\triangle})$.*

Proof. $PCFG \subseteq_{\text{T6.1}} U(\mathbb{\triangle}) \subseteq_{\text{L6.4}} U(\mathbb{\pitchfork}) \subseteq_{\text{L6.2}} U(\mathbb{\pitchfork}^{<\omega})$. Moreover $U(\mathbb{\triangle}) \subseteq_{\text{L6.5}} U(\triangle^{\omega})$. Hence $U(\mathbb{\triangle}) \subseteq U(\triangle^{\omega}) \cap U(\mathbb{\pitchfork}^{<\omega})$. Obviously $\{\updownarrow\rho \mid \rho \in \triangle^{\omega}\} \cap \{\updownarrow\rho \mid \rho \in \mathbb{\pitchfork}^{<\omega}\} = \emptyset$ and $\mathbb{\triangle}(G) \subseteq \triangle^{\omega}(G) \cup \mathbb{\triangle}^{<\omega}(G) \subseteq_{\text{L6.11}} \triangle^{\omega}(G) \cup \mathbb{\pitchfork}^{<\omega}(G)$. Thus by Lemma 6.3 we obtain $U(\triangle^{\omega}) \cap U(\mathbb{\pitchfork}^{<\omega}) \subseteq U(\mathbb{\triangle})$. □

Obviously $U(\mathbb{\triangle})$ and $\overline{U(\mathbb{\triangle})}$ are closed under deletion and insertion of terminals in bounded productions. Now Theorem 6.12 states that for the generation of polynomially bounded ambiguity, bounded productions must be essential. If we insert sufficiently many markers in bounded productions to destroy their capacity to cause ambiguity then the resulting grammar should be unambiguous. This is exactly what we are about to do. Note that we will use productions of the original grammar as marker symbols in the constructed grammar.

Lemma 6.13. *Each $\rho \in (\triangle^{<\omega}(G) \cap (\triangle(G) \setminus \triangleq(G)))$ has a unique decomposition $\rho = \xi p \tau \chi$ where $p \in P_{<\omega}$ is a bounded production, such that $p\tau \in \triangle(G)$, and $\xi\, \ell(p)\, \chi \in \triangleq^{<\omega}$.*

Proof. Let ρ be an element of $(\triangle^{<\omega}(G) \cap (\triangle(G) \setminus \triangleq(G)))$. Then ρ contains at least on bounded production. For each internal node μ in ρ there is a uniquely defined decomposition $\rho = \xi p \tau \chi$ such that $|\xi| = \mu - 1$ and $p\tau,\ \xi\ell(p)\chi \in \triangle(G)$. Our task is to find the appropriate μ. Since ξ must not contain a bounded production but p is a bounded production, the only possible candidate for μ is the smallest integer i such that $\rho[i]$ is a bounded production. Now we choose the uniquely defined $\xi \in (V \cup P_\omega)^*,\ \tau, \chi \in (V \cup P)^*$, and $p \in P_{<\omega}$ with the property $\rho = \xi p \tau \chi$ and $\rho' := \xi\ell(p)\chi$, $p\tau \in \triangle(G)$. Now assume that there is a node μ' in the χ portion of ρ' which is P-labeled by a bounded production. Since $\mu < \mu'$ the node μ' cannot be an ancestor of μ. On the other hand μ is a leaf and is therefore no ancestor of μ'. Thus by Lemma 6.7 there is a node $\nu < \mu$ which is P-labeled by a bounded production, which is a contradiction to our choice of μ. That implies χ does not contain bounded productions. Therefore $\xi\ell(p)\chi \in \triangleq(G)$. Finally, since $\ell(p)$ is Pb, Lemma 6.6 implies that $\xi\ell(p)\chi \in \triangleq^{<\omega}(G)$. □

Definition 6.14. *Let $h_G : (N \cup P)^* \to (N \cup (N \times (V \cup P_{<\omega})^*))^*$ be the homomorphism defined by $h_G(X) := X$ for $X \in N \cup P_\omega$ and $h_G(p) := (A \to pu_0A_1 \cdots pu_{k-1}A_kpu_k)$ for $p = (A \to u_0A_1 \cdots u_{k-1}A_ku_k) \in P_{<\omega}$. The skeleton grammar for $G = (V, P, S)$ is defined by $s(G) := (V \cup P_{<\omega}, P', S)$ where $P' := \{h_G(p) \mid p \in P\}$.*

Note that restricted to compressed derivation trees the mapping h_G is a bijection from $comp(\triangle(G))$ to $comp(\triangle(s(G)))$.

Lemma 6.15. *$G \in U(\triangleq) \Rightarrow s(G) \in$ UCFG.*

Proof. Observe that $p \in P$ is an unbounded production of G if and only if $h_G(p)$ is an unbounded production of $s(G)$. Moreover $P_\omega = P'_\omega$ which implies $\triangleq(G) = \triangleq(s(G))$. For $G \in U(\triangleq)$ we must show that arbitrary $\rho_1, \rho_2 \in \triangle(s(G))$ have common interfaces only if $\rho_1 = \rho_2$. We prove this by induction on $|\rho_1|_{P_{<\omega}}$. The basis is that $\rho_1 \in \triangleq(s(G))$. Now $\downarrow\rho_1$ does not contain any symbols in $P_{<\omega}$. Since each production in $h(P_{<\omega})$ generates symbols in $P_{<\omega}$ and $\updownarrow\rho_1 = \updownarrow\rho_2$ we obtain that ρ_2 is in $\triangleq(s(G))$ too. By the observation above $\rho_1, \rho_2 \in \triangleq(G)$. Hence $G \in U(\triangleq)$ implies $\rho_1 = \rho_2$. Assume the claim has been proved for all $\rho \in \triangle(s(G))$ with at most n bounded productions. Let ρ_1 contain $n+1$ bounded productions. By Lemma 6.13 for $i \in \{1, 2\}$ we can uniquely decompose $\rho_i = \xi_i\ h(p_i)\ \tau_i\chi_i$ such that $\rho'_i := \xi_i l(h(p_i))\chi_i \in \triangleq(s(G))$, $h(p_i) \in P'_{<\omega}$, and $h(p_i)\tau_i \in \triangle(s(G))$. Since all bounded productions happen to occur in $h(p_i)\tau_i$ it follows that p_1 and p_2 generate the leftmost and rightmost occurrences of symbols from $P_{<\omega}$ in $\downarrow\rho_1$ and $\downarrow\rho_2$, respectively. By $\updownarrow\rho_1 = \updownarrow\rho_2$ this implies $p := p_1 = p_2$ and $\updownarrow\rho'_1 = \updownarrow\rho'_2$. Now $\rho'_1, \rho'_2 \in \triangleq(s(G))$ implies $\rho'_1, \rho'_2 \in \triangleq(G)$. Since $G \in U(\triangleq)$ this implies $\rho'_1 = \rho'_2$. Now p is both a terminal of $s(G)$ and a bounded production of G. Thus $h(p) = (A \to pu_0A_1pu_1 \cdots A_kpu_k)$ for some $k \in \mathbb{N}$, and for each $j \in \{1, \ldots, k\}$

we have $A, A_j \in N$ and $u_j \in \Sigma^*$. Then $\tau_i = \tau_{i,1} \cdots \tau_{i,k}$ has, for each $i \in \{1,2\}$, a unique decomposition in k derivation trees $\tau_{i,1}, \ldots, \tau_{i,k} \in \triangle(s(G))$ such that $A_j = \uparrow\tau_{i,j}$. Since $h(p)$ is a descending production it cannot occur in any $\tau_{i,j}$. Hence their yields cannot contain a p. Therefore we can uniquely retrieve the yield of each $\tau_{i,j}$ from $\downarrow\rho_i$, i.e., for each $j \in \{1,\ldots,k\}$ we have $\downarrow\tau_{1,j} = \downarrow\tau_{2,j}$. Hence $\updownarrow\tau_{1,j} = \updownarrow\tau_{2,j}$ for each $j \in \{1,\ldots,k\}$. But since they must not contain $h(p)$ they contain at most n bounded productions. Therefore by the inductive hypothesis $\tau_{1,j} = \tau_{2,j}$ for each $j \in \{1,\ldots,k\}$. This finally implies $\rho_1 = \rho_2$. □

By elementary combinatorial considerations we obtain:

Lemma 6.16. *$s(G) \in UCFG \Rightarrow am_G(n) = \mathcal{O}(n^k)$, where $k = \psi(s(G))(P_{<\omega})$.*

Lemma 6.17. *$PCFG = U(\mathbb{A}) = U(\mathbb{A}_{\uparrow\downarrow}) = U(\mathbb{A}_{\updownarrow})$.*

Proof. $PCFG \subseteq_{\text{T6.1}} U(\mathbb{A}) \subseteq_{\text{L6.12}} U(\mathbb{A}_{\updownarrow})$. By Lemma 6.4 we have $U(\mathbb{A}) = U(\mathbb{A}_{\uparrow\downarrow})$. Finally let $G \in U(\mathbb{A}_{\updownarrow})$ then $s(G) \in UCFG$ by Lemma 6.15. Thus $G \in PCFG$ by Lemma 6.16. □

Theorem 6.18. *$am_G = 2^{\Omega(n)}$ or $am_G = \mathcal{O}(n^k)$, where $k = \psi(s(G))(P_{<\omega})$.*

Proof. If $G \in \overline{U(\mathbb{A})}$ then $G \in ECFG$ by Theorem 6.1, i.e., $\text{am}_G = 2^{\Omega(n)}$. If $G \in U(\mathbb{A})$ then $G \in U(\mathbb{A}_{\updownarrow})$ follows by Lemma 6.17. Thus $s(G) \in UCFG$ by Lemma 6.15 and by Lemma 6.16 we obtain $\text{am}_G(n) = \mathcal{O}(n^k)$. □

Note that the value of k in the theorem above can be computed in polynomial time w.r.t. the size of the grammar.

Theorem 6.19. *The closure of UCFL under Parikh bounded projection coincides with PCFL*

Proof. By Lemma 3.4 the closure of *UCFL* under Parikh bounded projection is a subset of *PCFL*. Let $L \in PCFL$ and let $G = (V,P,S) \in PCFG$ such that $L = L(G)$. Then $G \in U(\mathbb{A}_{\updownarrow})$ by Lemma 6.17. By Lemma 6.15 this implies $L(s(G)) \in UCFL$. Obviously $L = \pi_V(L(s(G)))$. Finally we observe that π_V is Parikh bounded for $L(s(G))$. □

Corollary 6.20. *A grammar $G = (V,P,S)$ is in PCFG if and only if (V,P_ω,S) is in $U(\triangle)$.*

Proof. Obviously $\triangle((V,P_\omega,S)) = \mathbb{A}_{\updownarrow}(G)$. Hence by Theorem 6.17 the claim follows. □

In general *PCFG* is undecidable, which has been shown in [11]. But in special cases Corollary 6.20 can help us to decide whether G belongs to *PCFG* since there are fewer derivation trees to consider. Furthermore the constructed grammar is not necessarily reduced. This can be used to break it into a bunch of grammars which are likely to be much smaller and easier to handle than the original one:

Corollary 6.21. *Let $G = (V,P,S)$ be a grammar and $N_r := \{B \in N \mid \exists p \in P_{<\omega} : |r(p)|_B > 0\} \cup \{S\}$. Then $G \in PCFG$ if and only if for all $A \in N_r$ we have $red((V \cup \tilde{N}_G, P_\omega \cup P', A)) \in UCFG$, where $\tilde{N}_G := \{\tilde{X} \mid X \in N\}$ is a copy of the nonterminals disjoint from V, and $P' := \{A \to \tilde{A} \mid A \in \tilde{N}_G\}$.*

7 The semiring closure of *UCFL*

Definition 7.1. *If X is a language class then $X[\cup]$ and $X[\cdot]$ denote the closure of X under union and concatenation, respectively. $X[\cdot,\cup]$ denotes the closure under union and concatenation. $X[\cdot,\cup]$ is called the semiring closure of X.*

Lemma 7.2. *Each language in $UCFL[\cdot,\cup]$ can be parsed by the Earley algorithm in $\mathcal{O}(n^2)$.*

The previous lemma can be proved analogously to the proof for the quadratic parsing time of metalinear languages [4].

Definition 7.3. *We define $L_p := \{u \in \{b,c\}^* \mid u = u^R\}$, where u^R is the reversal of u. Thus L_p is the set of palindromes. Now we define $L_\diamond := \{w \in \{a,b,c,\#\}^* \mid \exists i \in \mathbb{N} : \exists u,v \in L_p : w = a^i\#uv\#a^i\}$.*

Lemma 7.4. *$L_\diamond \in PCFL$.*

Proof. Let $L_1 := \{a^i\#\$\$\#a^i \mid i \in \mathbb{N}\}$. Obviously $L_1, L_p \in UCFL$ and the substitution $[\$/L_p]$ is Pb. By Lemma 3.4 this implies $L_\diamond = L_1[\$/L_p] \in PCFL$. □

Lemma 7.5. *$L_\diamond \notin UCFL[\cup,\cdot]$.*

Proof. Assume $L_\diamond \in UCFL[\cup,\cdot]$. By the distributive laws this is equivalent to $L_\diamond \in UCFL[\cdot][\cup]$. Thus for some $k,\ell \in \mathbb{N}$, $U_1,\ldots,U_\ell \in UCFL$, and $L_1,\ldots,L_k \in UCFL[\cdot]\setminus UCFL$ we have $L_\diamond = (\cup_{i=1}^{\ell} U_i) \cup (\cup_{i=1}^{k} L_i)$. Let us consider L_i for an arbitrary $i \in \{1,\ldots,k\}$. Now for some minimal $m \in \mathbb{N}$ we can write $L_i = \tilde{U}_1 \cdots \tilde{U}_m$ where $\tilde{U}_1,\ldots,\tilde{U}_m \in UCFL$. Since L_i is ambiguous we have $m > 1$. Each word in $L_\diamond$ contains exactly two #'s. Therefore $\forall j \in \{1,\ldots,m\} : \forall u,v \in \tilde{U}_j : |u|_\# = |v|_\#$. Assume the words in $\tilde{U}_1$ do not contain a # then $\tilde{U}_1$ only contains words of the form a^*. Recall that for each $w \in L_i = \tilde{U}_1 \cdots \tilde{U}_m$ the number of a's to the left of the first # must match the number of a's to the right of the second #. Therefore $\tilde{U}_1$ must be a singleton. But then $\tilde{U}_1 \cdot \tilde{U}_2$ is unambiguous contradicting the minimal choice of m. Thus each word in $\tilde{U}_1$ must contain the first #. Similarly we obtain that each word in $\tilde{U}_m$ must contain the second #. This implies that the words in $\tilde{U}_1$ and $\tilde{U}_m$ consist of words of the forms $a^*\#\{b,c\}^*$ and $\{b,c\}^*\#a^*$, respectively. Again, if the number of a's would not be fixed we could compose words with non matching "a" blocks. Hence $L_i \subseteq a^{n_i}\#\{b,c\}^*\#a^{n_i}$ for some $n_i \in \mathbb{N}$. We define $n = \max\{n_i \mid i \in \{1,\ldots,k\}\} + 1$. Let $R := a^n\#\{b,c\}^*\#a^n$. Then $L_\diamond \cap R = ((\cup_{i=1}^{\ell} U_i) \cup (\cup_{i=1}^{k} L_i)) \cap R = ((\cup_{i=1}^{\ell} U_i) \cap R) \cup ((\cup_{i=1}^{k} L_i) \cap R) = (\cup_{i=1}^{\ell} U_i) \cap R = \cup_{i=1}^{\ell} (U_i \cap R)$. Since unambiguous languages are closed under intersection with regular sets [5], this implies $L_\diamond \cap R \in UCFL[\cup]$. Moreover, unambiguous languages are closed under cancellation of singletons [5]. By cancellation of $a^n\#$ from the left-hand side and $\#a^n$ from the right-hand side, we obtain $L_pL_p \in UCFL[\cup]$. But this is false since in [3] it is proved that L_pL_p has infinite ambiguity. Therefore $L_\diamond \notin UCFL[\cup,\cdot]$. □

As an immediate consequence of Lemmas 3.4, 7.4 and 7.5 we obtain:

Theorem 7.6. *$UCFL[\cup,\cdot] \subsetneq PCFL$.*

8 Conclusion

We have shown that *PCFL* is the closure of unambiguous languages under Parikh bounded projection. Even if we use the formally stronger operation of Parikh bounded substitution we cannot leave *PCFL*. There is another nontrivial characterization of *PCFL* which proves a gap between polynomial and exponential ambiguity [11]. By Corollary 6.20 we know that exponential ambiguity is independent from bounded productions. On the other hand ambiguity which is polynomially bounded crucially depends on bounded productions. Recall that in the construction of the skeleton grammar we have inserted terminals in bounded productions only. By Lemma 6.17 and Lemma 6.15 this is sufficient to destroy subexponential ambiguity. The class *PCFL* is not only interesting for structural research, but also for applications. For example, the concept of polynomially bounded ambiguity has been recently applied in [2].

Acknowledgments: Thanks to Markus Lohrey and Gundula Niemann for proofreading and valueable discussions, and to Horst Prote, for some LaTeX tips.

References

1. J. Berstel. *Transductions and context-free languages.* Teubner, Stuttgart, 1979.
2. A. Bertoni, M. Goldwurm, and M. Santini. Random generation and approximate counting of ambiguously described combinatorical structures. In H. Reichel and S. Tison, editors, *Proc. STACS 2000*, LNCS 1770, pp. 567–580, Berlin-Heidelberg-New York, 2000. Springer.
3. J. Crestin. Un langage non ambigu dont le carré est d'ambiguité non bornée. In M. Nivat, editor, *Automata, Languages and Programming*, pp. 377–390. Amsterdam, North-Holland, 1973.
4. J. C. Earley. *An efficient context-free parsing algorithm.* PhD thesis, Carnegie-Mellon Uni., 1968.
5. M. A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, 1978.
6. H. Maurer. The existence of context-free languages which are inherently ambiguous of any degree. Research series, Dept. of Mathematics, Uni. of Calgary, 1968.
7. M. Naji. Grad der Mehrdeutigkeit kontextfreier Grammatiken und Sprachen, 1998. Diplomarbeit, FB Informatik, JWG-Universität Frankfurt/M.
8. R. J. Parikh. Language–generating devices. In *Quarterly Progress Report*, volume 60, pp. 199–212. Research Laboratory of Electronics, M.I.T, 1961.
9. A. Salomaa and M. Soittola. *Automata theoretic aspects of formal power series.* Springer, 1978.
10. K. Wich. Kriterien für die Mehrdeutigkeit kontextfreier Grammatiken, 1997. Diplomarbeit, FB Informatik, JWG-Universität Frankfurt/M.
11. K. Wich. Exponential ambiguity of context-free grammars. In G. Rozenberg and W. Thomas, editors, *Proc. DLT, 1999*, pp. 125–138. World Scientific, Singapore, 2000.
12. K. Wich. Sublinear ambiguity. In M. Nielsen and B. Rovan, editors, *Proc. MFCS 2000*, LNCS 1893, pp. 690–698, Berlin-Heidelberg-New York, 2000. Springer.