数理解析研究所講究録
1489巻 2006年 22-28

CFG/PDAのalternation付与方法ほかについて

(Some Remarks on Alternating Context-Free Grammars)

– An Extended Abstract –

早稲田大学 教育・総合科学学術院　守屋 悦朗 (Etsuro Moriya)

Abstract

交代性CFG (ACFG) は，文脈自由文法 (CFG) の非終端記号に alternation を入れることにより [Mor89] で導入された．[MHHO05] では，CFG に (alternation を含む) 状態を入れることにより状態交代性 CFG (sACFG) を導入し，交代性プッシュダウンオートマトン (APDA) の特徴付けを初めて与えた．本ノートでは，スタック記号に alternation を入れた PDA と，ACFG に (alternation の無い) 状態を導入した EACFG について考察し，ACFG と sACFG の特徴付けを与える．また，ACFG や sACFG の拡張の生成能力についても吟味する．

The first attempt to introduce 'alternation' into the CFG to define the ACFG was proposed in [Mor89]. Another alternating CFG called the state-alternating CFG (sACFG) was introduced recently in [MHHO05], where characterizations of the alternating pushdown automaton (APDA) by means of ACFGs were established. In this note we introduce further new variants of alternating CFG and PDA to obtain characterizations of ACFG and sACFG. We also investigate relationships among some extensions of the grammars.

0 Introduction

交代性文脈自由文法 (ACFG) の概念は交代性プッシュダウンオートマトン (APDA) を特徴付ける目的で [Mor89] において導入された．ACFG(*alternating context-free grammar*) は5項組 $G = (V, U, \Sigma, P, S)$ で表わされるが，ここで，V は非終端記号アルファベット，$U \subseteq V$ は全称型 (*universal*) 非終端記号の集合，$V \setminus U$ は存在型 (*existential*) 非終端記号の集合，Σ は終端記号アルファベット，$S \in V$ は出発記号，P は文脈自由プロダクションの有限集合である．

G における導出の過程で存在型非終端記号は通常の CFG における非終端記号と同様に書き換えが行なわれるが，全称型の非終端記号には，その記号を左辺に持つプロダクションすべてを同時に適用する (その結果，複数個の文形式が同時に生成される)．こうして，導出は文形式の1次元的連なりではなく，文形式をラベルとして付けられた木となる．終端語 w が G において生成されたとみなされるのは，そのような有限の木 T で次の条件を満たすものが存在する場合である：

(i) T の根には出発記号 S がラベル付けされている．
(ii) T のどの葉にも w がラベル付けされている．

以下では，ACFG のクラスを ACFG で表わし，ACFG によって生成される言語のクラスを $\mathcal{L}$(ACFG) で表わす．また，ε-プロダクションを持たない ACFG によって生成される言語のクラスを $\mathcal{L}$(ε-free-ACFG) で表わし，線形 ACFG によって生成される言語のクラスを $\mathcal{L}$(linear-ACFG) で表わす．さらに，最左導出に限定した ACFG によって生成される言語のクラスを $\mathcal{L}_{\rm lm}$(ACFG) および $\mathcal{L}_{\rm lm}$(ε-free-ACFG) で表わす．

[Mor89] では APDA の特徴付けは得られなかったが，その後，ACFG に関する興味深い結果がいくつか証明されている．例えば，[ChTo90] では，$\mathcal{L}_{\mathsf{lm}}(\mathsf{linear\text{-}ACFG})$ とか $\mathcal{L}_{\mathsf{lm}}(\varepsilon\text{-}\mathsf{free\text{-}ACFG})$ といった ACFG 言語のクラスと計算量クラスの間の密接な関係

$$\mathsf{P} = \mathsf{LOG}(\mathcal{L}_{\mathsf{lm}}(\mathsf{linear\text{-}ACFG})), \quad \mathsf{PSPACE} = \mathsf{LOG}(\mathcal{L}_{\mathsf{lm}}(\varepsilon\text{-}\mathsf{free\text{-}ACFG}))$$

が示されている．ここで，$\mathsf{LOG}(\mathcal{L})$ は言語のクラス $\mathcal{L}$ の対数領域還元の下での閉包を表わす．線形 ACFG は必然的に最左導出しかないので，最初の式から $\mathsf{P} = \mathsf{LOG}(\mathcal{L}(\mathsf{linear\text{-}ACFG}))$ が得られる．

次に，[IJW92] では，$\mathcal{L}(\mathsf{APDA}) = \mathcal{L}(\mathsf{linear\text{-}erasing\text{-}ACFG})$ ということを証明することにより，不完全ながら $\mathcal{L}(\mathsf{APDA})$ の文法的特徴付けを与えている．ここで，ACFG G が線形消去型 (*linear erasing*) であるとは，ある定数 c が存在して，G において生成される長さ n のどの語も長さが高々 $c \cdot n$ である文形式しかラベルとして含んでいないような導出木を持つことである．しかし，[IJW92] で用いられている ACFG では，終端語 w に終止符 (*endmarker*) \$ を付けた \$$w$\$ という形の語だけを考えることにより導出のコントロールを行なっているので真正の ACFG ではない．実際，$\mathcal{L}(\mathsf{linear\text{-}erasing\text{-}ACFG}) \subseteq \mathcal{L}(\mathsf{APDA})$ が成り立つことは容易に証明できるが，逆の包含関係が成り立つかどうかはわかっていない．

著者らは [MHHO05] において，[Kas70] により導入された状態付き CFG (以下，ECFG と略記する) の概念と alternation の概念とを結びつけることにより，次のような状態付き ACFG (sACFG) を導入し，APDA の文法的特徴付けに成功した．ここで，sACFG(*state-alternating context-free grammar*) とは 8 項組

$$G = (Q, U, V, \Sigma, P, S, q_0, F)$$

で定まるシステムである．$U \subseteq Q$ は全称状態 (*universal state*) の集合，$F \subseteq Q$ は受理状態の集合である．G における文形式は (p, α) なる形の対 ($p \in Q$, $\alpha \in (V \cup \Sigma)^*$) である．$p$ が存在状態，すなわち $p \in Q \setminus U$ である場合には，α の中に現れる非終端記号 A ($\alpha = \alpha_1 A \alpha_2$ であるとする) はプロダクション $(p, A) \to (q, \beta)$ によって (p, α) から $(q, \alpha_1 \beta \alpha_2)$ を直接導出すると定義する．しかし，もし p が全称状態であり，左辺が (p, A) であるプロダクションが全部で

$$(p, A) \to (q_1, \beta_1),\ \ldots,\ (p, A) \to (q_s, \beta_s)$$

だけあった場合には，$(p, \alpha_1 A \alpha_2)$ からは

$$(q_1, \alpha_1 \beta_1 \alpha_2),\ \ldots,\ (q_s, \alpha_1 \beta_s \alpha_2)$$

という文形式すべてが同時に直接導出されるものと定義する．その後，これらの文形式はそれぞれ独立に同様に直接書き換えが行なわれていく．このように，G における導出は，交代性オートマトンにおける計算木のように展開される．根のラベルが (p, α) で，どの葉のラベルも (p, w)，$p \in F$，であるような導出木が存在するとき $w \in \Sigma^*$ は G によって生成されたといい，G が生成する終端語の集合を $L(G)$ で表わす．

最後に，最左導出については述べるまでもないが，制限最左導出 (*leftish derivation*) を次のように定義する．これは [Kas70] において初めて導入された概念である．$(p, A) \to (q, \alpha)$ がプロダクションであり，β のいかなるプレフィックスにも状態 p の下で適用できるプロダクションが無いとき，直接導出 $(p, \beta A \gamma) \Rightarrow (q, \beta \alpha \gamma)$ は制限最左であるという．すなわち，β は非終端記号を含んでいてもよいが，状態 p の下ではそれを書き換えるプロダクションが存在せず，p の下でプロダクションが適用できる最左の非終端記号が A であるというのが制限最左の条件である．ACFG や ε-プロダクションを持たない ACFG によって制限最左導出の下で生成される言語のクラスを $\mathcal{L}_{\mathsf{lt}}(\mathsf{ACFG})$ および $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-}\mathsf{free\text{-}ACFG})$ で表わす．

1 Yet Another Variant of ACFG

We will consider yet another variant of ACFG. An *extended alternating context-free grammar*, EACFG for short, is a context-free grammar $G = (Q, V, U, \Sigma, P, S, q_0, F)$ with states, in which a subset U of the set of variables V is designated as *universal variables*, and a subset F of the set of states Q is designated as final states. Let $(p, \beta A\gamma)$ be a sentential form of G, where $p \in Q$, $\beta, \gamma \in (V \cup \Sigma)^*$, and $A \in V$. If A is an existential variable, that is, $A \in V \setminus U$, then we choose a production $(p, A) \to (q, \alpha)$ and derive $(q, \beta\alpha\gamma)$ from $(p, \beta A\gamma)$. If, however, A is a universal variable, and if $(p, A) \to (q_i, \alpha_i)$ $(1 \leqq i \leqq k)$ are all the productions with left-hand side (p, A), then from $(p, \beta A\gamma)$ we obtain all the sentential forms $(q_i, \beta\alpha_i\gamma)$ $(1 \leqq i \leqq k)$ in parallel, and following this step all these sentential forms are rewritten further, independently of each other. In this way a *derivation tree* is obtained from G in analogy to the computation tree that is associated with an alternating automaton and its input. The language $L(G)$ that is generated by G consists of all those words $w \in \Sigma^*$ for which there exists a derivation tree such that the root is labelled with the pair (q_0, S) and each leaf is labelled with a pair of the form (p, w) with $p \in F$. Here we remark that the labels of different leaves may differ in the first component, that is, in the final state, but that they must agree in the second component, that is, in the terminal string generated. Thus, the EACFGs are obtained from the ACFGs by introducing states, that is, in just the same way as the ECFGs are obtained from the context-free grammars. Also the EACFGs can be seen as being obtained from the ECFGs by distinguishing between universal and existential variables, that is, in just the same way as the ACFGs are obtained from the context-free grammars. Hence, the EACFGs unify these two generalizations of the context-free grammars.

As we see below, the EACFGs are equivalent in expressive power to the sACFGs.

Theorem 1.1. *The* EACFG *and the* sACFG *have exactly the same expressive power.*

Thus, in what follows we will mainly consider the sACFGs. Concerning the relationship between ACFG and sACFG, the next theorem shows that sACFG is at least more powerful than ACFG.

Theorem 1.2. *For each* ACFG G, *we can construct an* sACFG G' *such that* $L_{\mathsf{m}}(G) = L_{\mathsf{m}}(G')$ *holds for each derivation mode* m, $\mathsf{m} \in \{$*leftmost, rightmost, leftish, unrestricted*$\}$. *Moreover, if* G *is* ε*-free and/or (right-) linear, then so is* G'.

2 A Machine Characterization

The original purpose for introducing the ACFGs in [Mor89] was to give a characterization for the language class $\mathcal{L}$(APDA), the class of languages accepted by alternating pushdown automata. The following characterization of APDA in terms of sACFG was shown in [MHHO05].

Theorem 2.1. $\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG}) = \mathcal{L}(\mathsf{APDA})$.

We will next derive a characterization of the language class $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACFG})$ in terms of a variant of the alternating pushdown automata, the so-called *stack-alternating pushdown automaton*, abbreviated as stackAPDA. A stackAPDA is a pushdown automaton which has a single state only and whose pushdown symbols are divided into two types, *universal* and *existential* ones. Further, a stackAPDA accepts by empty pushdown. Thus, a stackAPDA is denoted by a 5-tuple $M = (\Sigma, \Gamma, U, \delta, Z_0)$, where Σ and Γ are the finite sets of input and pushdown symbols, respectively, $U \subseteq \Gamma$ is the set of universal pushdown symbols, and $Z_0 \in \Gamma$ is the initial pushdown symbol. The transition relation δ is a partial function from $(\Sigma \cup \{\varepsilon\}) \times \Gamma$ into the finite subsets of Γ^*. A configuration $(x, Z\alpha) \in \Sigma^* \times \Gamma\Gamma^*$ of M represents the current content x on the input tape and the current content $Z\alpha$ on the pushdown store, where the input head is on the leftmost symbol of x and Z is the topmost symbol on the pushdown store. The initial configuration on input x is (x, Z_0). For a given input x, a *computation tree* for M is a finite rooted tree the nodes of which are labelled with configurations. It is defined in the same way as that for an ordinary alternating pushdown automaton, except that the pushdown symbol on the top of the pushdown store of M plays the counterpart to the universal or existential states of an ordinary alternating pushdown automaton. Thus, an input x is accepted by the stackAPDA M, if there is a computation tree for M such that the root is labelled with (x, Z_0), and each leaf is labelled with the pair $(\varepsilon, \varepsilon)$. Such a computation tree is called an *accepting computation tree* for the input x.

It is important to note that if a is an input symbol and Z is a universal stack symbol of a stackAPDA M, and if M contains the transitions $\delta(a, Z) = \{\beta_1, \ldots, \beta_p\}$ as well as $\delta(\varepsilon, Z) = \{\gamma_1, \ldots, \gamma_q\}$, then a node π of a computation tree of M that is labelled with a configuration of the form $(ax, Z\alpha)$ has either p sons labelled with $(x, \beta_1\alpha), \ldots, (x, \beta_p\alpha)$, respectively, or q sons labelled with $(ax, \gamma_1\alpha), \ldots, (ax, \gamma_q\alpha)$, respectively. That is, we have to choose nondeterministically whether to apply the a-transitions or the ε-transitions, and we cannot apply both simultaneously. If we relax that condition, requiring that in the above situation both the a- and the ε-transitions must be applied, then we say that the stackAPDA M works in *relaxed mode.* By $L_{\mathsf{relaxed}}(M)$ we denote the language that is accepted by M in relaxed mode, and $\mathcal{L}_{\mathsf{relaxed}}(\mathsf{stackAPDA})$ denotes the class of all languages that are accepted in this way.

Lemma 2.2. $\mathcal{L}(\mathsf{stackAPDA}) = \mathcal{L}_{\mathsf{relaxed}}(\mathsf{stackAPDA})$.

Hence, we see that for stackAPDAs, it does not matter whether we consider the standard mode or the relaxed mode of operation. Based on the normal form result for ACFGs and the above result on stackAPDAs we can derive the following characterization.

Theorem 2.3. $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACFG}) = \mathcal{L}(\mathsf{stackAPDA})$.

3 Extensions of Alternating CFGs

It should be natural to apply the notion of alternation to each type of grammars in the Chomsky hierarchy. For the purpose, we will consider in this note alternating variants of the type i grammar in the Chomsky hierarchy.

A string of variables and terminal symbols is said to be *universal* if the leftmost occurrence among occurrences of variables is a universal variable, otherwise it is *existential.* Accordingly a production is said to be universal if its lefthand side is universal, otherwise it is existential. Without loss of generality, we may assume that each production of a type 0 grammar is of the following form:

$$\alpha \rightarrow \beta,$$

where α is a string of variables and β is a string of terminal symbols and variables.

Now, derivation trees for a given type 0 grammar can be defined similarly as those for ACFGs. To be precise, let us restate how productions are to be applied universally (Existential application of a production is defined similarly). Let n be a node of a derivation tree labelled with a sentential form $\alpha\beta\gamma$. If $\beta \rightarrow \eta_1, \ldots, \beta \rightarrow \eta_k$ are the productions in P each of whose lefthand sides is β, then these productions must be applied simultaneously to the same occurrence of β in the sentential form $\alpha\beta\gamma$ to yield k sons of n, with respective labels $\alpha\eta_1\gamma, \ldots, \alpha\eta_k\gamma$. If, in addition, the restriction that α is a terminal string is imposed, then this derivation step is said to be *leftmost.* By $\mathcal{L}_{\mathsf{lm}}(\mathsf{X})$ we denote the class of languages generated by grammars of type X with respect to the leftmost derivation mode.

In [MN97] the fact that $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG}) = \mathcal{L}(\mathsf{APDA})$ holds was shown by a rather complicated proof. Based on the equivalece $\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$ and $\mathcal{L}(\mathsf{APDA})$ established in [MHHO05] we can give a generalization of their result by a simpler proof.

Theorem 3.1. $\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{Atype0})$.

It should be natural and easy to introduce a 'state-alternating' vartiant of any type of grammar in the Chomsky hierarchy. For each type of grammar X, let sAX denote the state-alternating variant of X that is defined similarly as sACFG is defined from CFG.

We have the following equivalence.

Theorem 3.2. $\mathcal{L}_{\mathsf{lm}}(\mathsf{Atype0}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sAtype0}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$.

Obviously we have $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{Atype0})$ and thus we also obtain the following corollary.

Corollary 3.3. $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG}) = \mathcal{L}(\mathsf{APDA})$.

We don't know at present whether or not the converse inclusion of Corollary 3.3 holds.

Now, let ALBA stand for 'alternating linear bounded automaton.' We would like to know any relationship of $\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$ to $\mathcal{L}(\mathsf{ACSG})$, since it is known that $\mathcal{L}(\mathsf{APDA}) = \mathcal{L}(\mathsf{ALBA})$ [CKS81] and $\mathcal{L}(\mathsf{APDA}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$ [MHHO05]. We have the following result.

Theorem 3.4. $\mathcal{L}(\mathsf{ACSG}) = \mathcal{L}(\mathsf{ALBA})$.

Corollary 3.5. $\mathcal{L}(\mathsf{ACSG}) = \mathcal{L}(\mathsf{APDA}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$.

Corollary 3.6. $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG}) \subseteq \mathcal{L}(\mathsf{ACSG})$.

The remaining problem we have to consider is the converse of Corollary 3.6. By Corollary 3.5, this is equivalent to the inclusion $\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG})$. As far as we restrict ourselves to ε-free sACFGs, this inclusion is in fact true.

Lemma 3.7. $\mathcal{L}_{\mathsf{lm}}(\varepsilon\text{-free sACFG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG})$.

On the other hand, by a similar method used in the proof of Theorem 3.1, we can prove the following equivalence.

Theorem 3.8. $\mathcal{L}_{\mathsf{lm}}(\mathsf{sACSG}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG})$.

4 The leftish mode

Compared to the class $\mathcal{L}_{\mathsf{lm}}((\mathsf{s})\mathsf{ACFG})$ of languages generated by (s)ACFGs with respect to the leftmost derivation mode, almost nothing is known about the corresponding class $\mathcal{L}_{\mathsf{lt}}((\mathsf{s})\mathsf{ACFG})$ of languages generated by (s)ACFGs with respect to the leftish derivation mode.

Obviously we have $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free ACFG}) \subseteq \mathcal{L}_{\mathsf{lt}}(\mathsf{ACSG})$. In fact, we can prove a little bit more. Note that $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free ACFG}) \subseteq \mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free sACFG})$ [MHHO05]. Also note that it is not known whether or not $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free sACFG}) \subseteq \mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free ACFG})$ holds.

We can show the following interesting result:

Theorem 4.1. $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free sACFG}) = \mathcal{L}_{\mathsf{lm}}(\varepsilon\text{-free sACFG})$.

Theorem 4.2. $\mathcal{L}_{\mathsf{lt}}(\mathsf{ACSG}) = \mathcal{L}_{\mathsf{lt}}(\mathsf{sACSG})$.

It is known that $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free sCFG}) = \mathcal{L}(\mathsf{CSG})$ [Mau73, Sal72]. Corresponding to this fact, it should be interesting to know whether or not the equivalence $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free sACFG}) = \mathcal{L}(\mathsf{ACSG})$ holds. It is already known that $\mathcal{L}(\varepsilon\text{-free ACFG}) \subseteq \mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free sACFG})$ [MHHO05].

5 Problems to be considered further

In this note we have considered mostly several classes of languages generated by (state-) alternating phrase-structure grammars with respect to leftmost derivation mode, as extensions of (s)ACFG. The most important problem we have left open is the following.

Open Problem 1. *Do the following equivalences hold?*
(1) $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACFG}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$*?*
(2) Does $\mathcal{L}(\mathsf{ACFG}) = \mathcal{L}(\mathsf{sACFG})$*?*

As to the difference of leftmost and unrestricted modes of derivations, the following problem is to be considered.

Open Problem 2. *Does* $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG}) = \mathcal{L}(\mathsf{ACSG})$ *hold?*

Among many known inclusions about alternating CFGs, the following one is interesting, because it can be regarded as the corresponding variant of the fact that the language generated by an arbitrary type 0 phrase structure grammar with respect to leftmost derivations is a contex-free language [Mat64].

Open Problem 3. *Does* $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACFG}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{Atype0})$ *hold?*

Other important problems to be considered are to find some relationships between the langauges classes generated by alternating grammars with respect to leftmost, leftish and unrestricted derivation mode. For example,

Open Problem 4. *Can we prove* $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free sACFG}) = \mathcal{L}(\mathsf{ACSG})$*?*

Note that for non-alternating grammars we have $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-free sCFG}) = \mathcal{L}(\mathsf{CSG})$.

Taking the above result concerning the relationship between ACSGs and ACFGs (with respect to either leftmost mode or unrestricted mode) into consideration, it seems meeningful to consider alternating version of growing CSGs [BO98].

Acknowledgement. The author expresses his sincere thanks to Prof. F.Otto of Kassel University, Germany, for many fruitful discussions on the results presented here.

References

[BO98] G. Buntrock and F. Otto. Growing context-sensitive languages and Church-Rosser languages. *Information and Computation*, 141:1–36, 1998.

[CKS81] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Assoc. for Comput. Mach.*, 28: 114–133, 1981.

[ChTo90] Z.Z. Chen and S. Toda. Grammatical characterizations of P and PSPACE. *The Transactions of the IEICE*, E 73: 1540–1548, 1990.

[IJW92] O.H. Ibarra, T. Jiang, and H. Wang. A characterization of exponential-time languages by alternating context-free grammars. *Theoretical Computer Science*, 99: 301–313, 1992.

[Kas70] T. Kasai. An infinite hierarchy between context-free and context-sensitive languages. *Journal of Computer and System Sciences*, 4: 492–508, 1970.

[LLS78] R.E. Ladner, R.J. Lipton, and L.J. Stockmeyer. Alternating pushdown automata. In *Proceedings of the 19th FOCS*, pp.92–106. IEEE Computer Society Press, 1978.

[LLS84] R.E. Ladner, R.J. Lipton, and L.J. Stockmeyer. Alternating pushdown and stack automata. *SIAM Journal on Computing*, 13: 135–155, 1984.

[Lan02] M. Lange. Alternating contex-free languages and linear time μ-calculus with sequential composition. *Electronic Notes in Theor. Comput. Sci.*, 68, 2002.

[Mat64] Matthew, G. H. A note on asymmetry in phrase structure grammars. *Inform. Contr.*, 7, 360–365, 1964.

[Mau73] Maurer,H.A. Simple matrix languages with a leftmost restriction. *Inform. Contr.* **23**, 128–139, 1973.

[Mor89] E. Moriya. A grammatical characterization of alternating pushdown automata. *Theoretical Computer Science*, 67: 75–85, 1989.

[MN97] E. Moriya and S. Nakayama. Grammatical characterizations of alternating pushdown automata and linear bounded automata. *Gakujutsu Kenkyu*, Series of Math, 45: 13–24, School of Education, Waseda Univ., 1997 (in Japanese).

[MHHO05] E. Moriya, D. Hofbauer, M. Huber, F. Otto. On State-Alternating Context-Free Grammars. *Theoretical Computer Science*, 337: 183–216, 2005.

[Okh01] A. Okhotin. Conjunctive grammars. *Journal of Automata, Languages and Combinatorics*, 6: 519–535, 2001.

[Sal72] A. Salomaa. Matrix grammars with a leftmost restriction. *Information and Control*, 20: 143–149, 1972.

[Sud75] H. Sudborough. A note on tape-bounded complexity classes and linear context-free languages. *Journal of the Assoc. for Comput. Mach.*, 22: 499–500, 1975.