19

Further Properties of Path-Controlled Grammars

Carlos Martín-Vide and Victor Mitrana

Abstract

A path controlled grammar consists in a pair of CFGs, G_1, G_2, where G_2 generates a language over the total alphabet of G_1. A word w generated by G_1 is accepted only if there is a derivation tree τ of w with respect to G_1 such that there exists a path in τ, from the root to a leaf node, having the nodes marked by a word x which is in $L(G_2)$. We show that these grammars are slightly more powerful than CFGs, but they preserve all the properties of CFGs: closure properties, pumping lemmas, decision properties, polynomial recognition. Moreover, many of these properties are mainly based on the corresponding properties of CFGs.

Keywords path-controlled grammar, closure properties, decision property, polynomial recognition.

This work is a continuation of the investigation started in Marcus et al. (2001), where a new-old type of control on context-free grammars is considered. This type of control is extracted and abstracted from a paper (Bellert (1965)) with very solid linguistic motivations. The goal of this paper is to complete the picture of path-controlled grammars started in Marcus et al. (2001) with some mathematical properties which are missing from the aforementioned work: closure and decidability properties, including a polynomial recognition algorithm.

19.1 Introduction

The work by Bellert (1965) is very well motivated, starting from the assertion that the natural language is not context-free. The paper not only motivates a theoretical study by this assertion, but, after intro-

FG-MoL 2005.
James Rogers (ed.).

ducing the “relational grammars", a consistent case study is examined, using the new formalism for modelling several constructions from Polish. Bellert applies her relational grammar to the generation of Polish kernel sentences. In this application, 13 relations are considered, mainly concerning the agreement in gender, number or case between the noun and the predicate.

Informally, the framework is the following one. One considers a context-free grammar (CFG for short) $G = (N, T, S, P)$ and a way to select from all possible derivation trees in G only those trees with certain properties, specified by a set of *tuple-grammars* and a *relation* on the rules of these tuple-grammars. More specifically, one gives several k-tuples of CFGs, $(G_1, \ldots, G_k)$, of the form $G_i = (N_i, N \cup T, S_i, P_i)$. Note that these grammars have as the terminal alphabet the total alphabet of G. The grammars from such a tuple work in parallel, over separate sentential forms, also observing the restriction imposed by a relation $\rho \subseteq P_1 \times \cdots \times P_k$. In a derivation of the k-tuple of grammars $(G_1, \ldots, G_k)$ we have to synchronously use k rules related by ρ. The k-tuples $(w_1, \ldots, w_k)$ of words generated in this way by $(G_1, \ldots, G_k)$ are then used for selecting from the derivation trees of G only those trees which have paths marked by $w_1, \ldots, w_k$, markers being assigned to nodes.

The definition of relational grammars contains an idea which has been investigated later in the regulated rewriting area: to impose some restriction on the paths present in a derivation tree of a CFG. One takes two CFGs, G_1, G_2, such that the rules of G_1 are labeled in a one-to-one manner with labels from a finite set of labels V_L and G_2 generates a language over V_L. A word w generated by G_1 is accepted only if there is a derivation tree τ of w with respect to G_1 such that all independent paths in τ are labeled (labels are now assigned to edges) by words in $L(G_2)$. In Khabbaz (1974) one defines an infinite hierarchy of languages based on this idea. The second class in this hierarchy is exactly the class of languages defined by a pair of grammars (G_1, G_2) as above with G_1 a linear grammar. Since at each point in a derivation of a linear CFG there is at most one nonterminal to expand, every derivation is associated with a single control word. This approach has been extended to arbitrary CFGs in Weir (1992). Both formalisms are special cases of a more general formalism proposed by Ginsburg and Spanier (1968). The concept of *tree controlled grammar* introduced in Culik and Maurer (1977) appears to have some weak connections with this approach; a tree controlled grammar is also a pair of CFGs (G, G') but a derivation tree τ of a word in G is “accepted” if the words labelling the nodes considered from left to right on each level in τ, excepting the

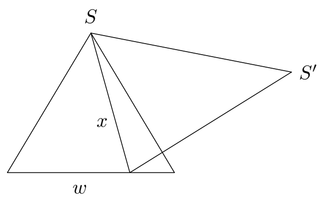

FIGURE 1: A Word Generated by G_1 Under the Partial Control of G_2

last one, can be generated by G'.

In Marcus et al. (2001) one proposes a bit different approach: one takes two CFGs, G_1, G_2, where G_2 generates a language over the total alphabet of G_1. A word w generated by G_1 is accepted only if there is a derivation tree τ of w with respect to G_1 such that there exists a path in τ, from the root to a leaf node, having the nodes marked by a word x which is in $L(G_2)$. For an intuitive representation of this idea, one should have in mind the fact that a word w is generated by G_1 under the "partial control" of G_2 if there exist two derivation trees, τ_1, τ_2, the first one with respect to G_1 and the second one with respect to G_2, such that the first tree has w as its frontier, while the second tree, having x as its frontier, is "orthogonal" to the first tree and its frontier word x describes a path in the first tree. Figure 1 illustrates this idea.

A pair of two grammars as above is called a path-controlled grammar in Marcus et al. (2001).

The main difference between the "partial control" defined above and the control on all independence paths proposed in Khabbaz (1974), Weir (1992), and Palis and Shende (1995) does not consists in the way of assigning words to paths (this translation can be done by a generalized sequential machine) but in the fact that the "partial control" requires the existence of just one path in the derivation tree labeled by a control word while the other control type requires that all independence paths to be labeled by control words. The relationship between the two formalisms seems an easy question at first sight: each path-controlled grammar can be simulated by a control grammar in the sense of Weir (1992) (more precisely, a labeled distinguished grammar with a control context-free language). However, we were not able to prove or disprove this; actually this depends on the definition of a control grammar. If one allows different labels for the same rule depending on the distinguished symbol identified in the right-hand side of that rule the statement is true. By space reason we skip the proof which is rather simple. However, if this is not permitted, then the problem could not be settled. It is worth mentioning that the properties reported in Palis and Shende (1992) and Palis and Shende (1995), including the complexity of the recognition problem, were proved for this latter case. Consequently, it seems that they cannot be easily extended to path-controlled grammars.

Having this in mind, we considered that it is worth presenting a more complete picture of path-controlled grammars. In Marcus et al. (2001) one mainly investigates the generative power and a very few linguistically oriented properties of path-controlled grammars. These grammars are slightly more powerful than CFGs, but they preserve all the properties of CFGs: closure properties, pumping lemmas, decision properties, polynomial recognition. Moreover, many of these properties are mainly based on the corresponding properties of CFGs. As a consequence, these devices belong to the so-called *mildly context-sensitive grammars* proposed in Joshi (1985). This extra power appears to be adequate for handling various phenomena which require more formal power than CFGs like duplication, crossed dependencies, multiple agreements up to four positions. Path-controlled grammars may also be viewed as a tree-generating formalism with some attractive properties for characterizing the strong generative power of grammars. We are not concerned here with their capacity to characterize the structural description associated with sentences, though this is a natural matter of great importance. The goal of this paper is to complete the picture of path-controlled grammars started in Marcus et al. (2001) with some mathematical properties which are missing from the aforementioned work: closure and decidability properties, including a polynomial recognition algorithm.

19.2 Path-Controlled Grammars

We use the standard formal language theory notions and notations, as available in many monographs, see, e.g., Hopcroft and Ullman (1979). In particular, V^* is the free monoid generated by the alphabet V under the operation of concatenation, λ is the empty word, and $|x|$ is the length of the word $x \in V^*$, As usual, when comparing two languages, the empty word is ignored, that is, we consider L_1 equal to L_2 if $L_1 - \{\lambda\} = L_2 - \{\lambda\}$.

Given a CFG $G = (N, T, S, P)$ without erasing rules, with derivations in G we associate derivation trees in the well-known manner. Let $S \Longrightarrow^* w$ be a terminal derivation in G and τ its associated tree. Each path from the root of τ to a leaf is described by a word of the form $SA_1, A_2, \ldots A_r a$, with $A_i \in N, 1 \leq i \leq r, r \geq 0$, and $a \in T$. We denote by *path*(τ) the language of all these words, describing paths in τ, by *path*(x) the union of all the languages *path*(τ), where τ is a derivation tree for x in G, and by *path*(G) the union of all these languages. Note that we consider that any path in a derivation tree ends by a node labelled by a terminal symbol.

A *path-controlled* grammar is a pair $\gamma = (G, G')$, where $G = (N, T, S, P)$ and $G' = (N', N \cup T, S', P')$ are context-free grammars. The language generated by γ is

$$L_\exists(\gamma) = \{w \in L(G) \mid path(w) \cap L(G') \neq \emptyset\}.$$

In other words, the language generated by γ is the yield of the tree language obtained from the derivation trees of G_1 which are "partially accepted" by G_2.

We denote by $PC_\exists(CFG, CFG)$ the family of languages $L_\exists(\gamma)$, where $\gamma = (G, G')$ is a path-controlled grammar having both component CFGs. It is of no interest to consider a control on paths imposed by regular grammars as this control does not increase the power of CFGs as shown in Marcus et al. (2001)

19.3 Closure Properties of $PC_\exists(CFG, CFG)$

Proposition 1 *$PC_\exists(CFG, CFG)$ is closed under the following operations: union, intersection with regular languages, left and right concatenation with context-free languages, substitution with λ-free context-free languages, non-erasing homomorphism. It is not closed under intersection.*

Proof We skip the proof for *union* and *right/left concatenation* with context-free languages which is trivial.

Intersection with regular languages: Let $\gamma = (G, G')$ be a path-controlled grammar and $\mathcal{A}$ be a finite automaton. By the standard proof one constructs the CFG G_1 generating $L(G) \cap L(\mathcal{A})$. Remember that the nonterminals of this grammars are triples of the form (q, A, q'), where A is a nonterminal of G and q, q' are states of $\mathcal{A}$. We now construct a grammar G'_1 generating the language $s(L(G'))$, where s is a finite substitution defined by

$$s(X) = \begin{cases} \{(q, X, q') \mid q, q' \text{ are states of } \mathcal{A}\}, & X \text{ is a nonterminal of } G \\ \{X\}, & X \text{ is a terminal of } G. \end{cases}$$

It is rather plain that $L_\exists(G_1, G'_1) = L_\exists(\gamma) \cap L(\mathcal{A})$.

Substitution with λ-free context-free languages: Let $\gamma = (G, G')$ be a path-controlled grammar with $G = (N, T, S, P)$, $T = \{a_1, a_2, \ldots, a_n\}$, and $s : T^* \longrightarrow 2^{U^*}$ be a substitution such that $s(a_i) = L(G_i)$, with $G_i = (N_i, U, S_i, P_i)$ being a CFG without λ-rules for all $1 \leq i \leq n$.

If $\widehat{G}$ is the CFG generating the language $s(L(G))$ and $\widetilde{G}$ is a grammar generating the language $g(L(G'))$, where g is a substitution which keeps unchanged any symbol from N and replace each a_i by a word in the regular set $S_i N_i^* U$, then $L_\exists(\widehat{G}, \widetilde{G}) = s(L_\exists(\gamma))$ holds. As a direct consequence, the closure of $PC_\exists(CFG, CFG)$ under non-erasing homomorphism follows.

Intersection: In Marcus et al. (2001), one proves that path-controlled grammars with linear CFG components "can count to four", that is they can generate the language $\{a_1^n a_2^n a_3^n a_4^n \mid n \geq 1\}$, but even path-controlled grammars with CFGs components "cannot count to five". By the closure under left and right concatenation with context-free languages, the statement follows. $\square$

We finish this section with a brief discussion about the significance of closure properties of a class of languages with respect to natural language. It should be emphasized that, though closure of operations is a very natural and elegant mathematical property, however non-closure properties seem to be natural too. One may argue that it is possible to construct a grammatical formalism for a language in the following way. One looks for a decomposition of the language into some independent fragments, constructs auxiliary grammars for such fragments, and then obtains the desired grammar by closure under union of the auxiliary grammars. Following this idea, the grammatical model should satisfy the closure property under union, which is fulfilled by almost all mildly context-sensitive devices including ours. Along the same lines, we can imagine another way of constructing a grammatical formalism for a language L. It consists of identifying a set of conditions $c_1, c_2, \ldots, c_n$ which define the correctness of L. Then, one tries to define a grammar generating the set $L(c_i)$ of all words observing the condition c_i and ignoring the other conditions. It is expected that the intersection $\bigcap_{i=1}^n L(c_i)$ gives exactly the language L. Now, the closure property which should be verified by the model is the one under intersection, which is not verified by many mildly context-sensitive mechanisms, among them the one investigated in this paper.

As far as the closure under concatenation is concerned, some authors consider that this is basic to natural languages while others have the opposite opinion, see, e.g., Kudlek et al. (2003) for such a discussion. It is worth mentioning here that the closure under concatenation of our grammatical formalism remains unsettled.

19.4 Decision Properties of $PC_{\exists}(CFG, CFG)$

We start this section by recalling a very simple result from Thatcher (1967) which will be useful in the sequel. Since we shall refer to this construction, for sake of completeness we prefer to give a construction as well

Lemma 1 *For any context-free grammar G, $path(G)$ is regular.*

Proof Let $G = (N, T, S, P)$ be a CFG, for later purposes we assume that it is reduced, namely each nonterminal is reachable from S and produces a terminal word. We construct the deterministic finite automaton $\mathcal{A}_G = (Q, N \cup T, \delta, q_0, \{q_f\})$, where $Q = \{q_0, q_f\} \cup \{[A] \mid A \in N\}$ and δ defined by

$$\delta(q_0, S) = [S], \quad \begin{array}{l}\delta([A], B) = [B], \\ \text{if } A \to uBv \in P,\end{array} \quad \begin{array}{l}\delta([A], a) = q_f, \\ \text{if } A \to uav \in P,\end{array}$$

where $u, v \in (N \cup T)^*$ and $a \in T$. Clearly, $path(G) = L(\mathcal{A})$. $\square$

In the same work one proves that each language generated by a path-controlled grammar is a matrix language (see Abraham (1965) and Dassow and Păun (1989) for the definition and properties of matrix languages). Since the emptiness problem is decidable for matrix grammars via a reduction to the reachability problem in vector addition systems shown to be decidable in Kosaraju (1982), we have:

Proposition 2 *The emptiness problem is decidable for path-controlled grammars.*

However, a direct proof can be obtained as an immediate consequence of the above lemma. Indeed, it is obvious that for any $\gamma = (G, G')$, where G is reduced, $L_{\exists}(\gamma) \neq \emptyset$ if and only if $path(G) \cap L(G') \neq \emptyset$. It suffices to remember that the class of context-free languages is effectively closed under intersection with regular sets and the emptiness problem is decidable for CFGs.

In Marcus et al. (2001) one presents a pumping lemma for languages generated by path-controlled grammars very similar to that for context-free languages. Since the membership problem is decidable for matrix grammars without erasing rules, by the pumping lemma from Marcus et al. (2001) one immediately infers that:

Proposition 3 *The finiteness problem is decidable for path-controlled grammars.*

However, we now present a more efficient algorithm for this problem. Let $\gamma = (G, G')$ be a path-controlled grammar; without loss of generality we may assume that G is reduced and without chain rules of the form $A \to B$, where A and B are nonterminals. If this is not the case, we eliminate these rules by the well known algorithm and then construct a *gsm* M which transforms each word in $L(G')$ as follows: when M scans a nonterminal A of G, it nondeterministically either lets it unchanged, or removes it provided that there is a rule $B \to A$ in G and B was just scanned before A. Therefore, subwords which correspond to recurrent derivations in G which produce nothing are either erased or not (the second case prevents the possibility to also have productive derivations with the same nonterminals). Because the class of context-free languages is closed under *gsm* mappings, the language $M(L(G'))$ is still context-free.

Clearly, if $path(G) \cap L(G')$ is an infinite set, then $L_{\exists}(\gamma)$ is infinite as well. The converse does not hold, since one may have an arbitrarily long path described by a word in $path(G)$ such that a prefix of this word is the prefix of a word in $path(G) \cap L(G')$. Therefore, if $path(G) \cap L(G')$ is finite but non-empty, we construct the two sets $\langle A\rangle_N$ and $\langle A\rangle_T$ for each nonterminal $A \in N$ as follows:

– $\langle A\rangle_N$ contains all nonterminals C such that there exists a rule $A \to \alpha \in P$ satisfying the following condition: There is a nonterminal B (which may be C as well) such that both B and C appear in α (if $B = C$, then α contains at least two occurrences of B) and the language accepted by the automaton $\mathcal{A}_G$ with the initial state $[B]$ is infinite.

– $\langle A\rangle_T$ contains all terminals under the same circumstances as above. Now it is easy to see that $L_{\exists}(\gamma)$ is infinite if and only if either $path(G) \cap L(G')$ is infinite or

$$\bigcup_{A \in N} ((path(G) \cap L(G')) \cap (N^*\{A\}\langle A\rangle_N N^* T \cup N^*\{A\}\langle A\rangle_T)) \neq \emptyset.$$

By the closure and decision properties of the class of context-free languages this condition can be algorithmically checked.

We now present a undecidable problem which, in its turn, is based on the undecidability of the universality problem for CFGs.

Proposition 4 *One cannot algorithmically decide whether or not the language generated by a given path-controlled grammar is context-free.*

Proof Let $L \subseteq V^*$ be an arbitrary context-free language and $L' \subseteq U^*$ be a non-context-free language in $PC_{\exists}(CF, CF)$. Assume that $V \cap U = \emptyset$. By the closure properties in Proposition 1 the language $LU^* \cup V^*L'$ can be generated by a path-controlled grammar γ. If $L = V^*$, then $L_{\exists}(\gamma) = V^*U^*$, hence it is context-free. If $L \neq V^*$, then we consider $w \in V^* \setminus L$ and observe that $L_{\exists}(\gamma) \cap \{w\}U^* = \{w\}L'$, hence $L_{\exists}(\gamma)$ cannot be context-free. Indeed, if $L_{\exists}(\gamma)$ were context-free, then $\{w\}L'$ would be context-free, hence L' would be context-free, a contradiction.

Consequently, $L_\exists(\gamma)$ is context-free if and only if $L = V^*$ which is undecidable. $\square$

19.5 Polynomial Recognition

The most important decision property for any class of grammars which aims to be considered as a model of syntax is the membership (recognition) problem, and especially the complexity of this problem provided that it is decidable. In this section we present two polynomial algorithms for recognizing languages generated by path-controlled grammars. We stress that such an algorithm might not be directly inferred from the work of Palis and Shende (1992) since it is likely that no efficient algorithm for simulating a path-controlled grammar with a control grammar in the sense of Palis and Shende (1992) exists.

We first give an algorithm arising from Propositions 1 and 2. Let $\gamma = (G, G')$ be a path-controlled grammar and w be a word of length n over the terminal alphabet of G. Assume that $\mathcal{A}$ is a finite automaton recognizing w; grammars G_1 and G'_1 from the first part of the proof of Proposition 1 can be constructed in polynomial time. According to Proposition 2,

$$w \in L_\exists(\gamma) \text{ iff } path(G_1) \cap L(G'_1) \neq \emptyset.$$

By Lemma 1 the automaton recognizing $path(G_1)$ as well as the CFG generating $path(G_1) \cap L(G'_1)$ can be constructed in polynomial time. Now, the emptiness problem for a language generated by a given CFG can be algorithmically solved in polynomial time. The worst case complexity of this algorithm, which is $O(n^{12})$, can be inferred from the proof of the next statement which proposes a bit more efficient algorithm.

Proposition 5 *The language generated by a given path-controlled grammar can be recognized in $O(n^{10})$ time.*

Proof The proof is mainly based on the well-known Cocke-Younger-Kasami (CYK for short) algorithm for the context-free languages recognition. This algorithm appears in many places, it may be found in the pioneering works of Kasami (1965) and Younger (1967). Let $\gamma = (G, G')$ be a path-controlled grammar with $G = (N, T, S, P)$ a reduced CFG in the Chomsky Normal Form (CNF) and $L(G') \subseteq SN^*T$. This supposition does not decrease the generality of the reasoning since if G is not in the CNF, then we can transform it by the classic algorithm and G' accordingly. More precisely, in the first step we allow only terminal rules of the form $X_a \to a$, where X_a is a new nonterminal for each $a \in T$. Each word in $L(G')$ is modified by inserting X_a before its last a. In the second step we eliminate all the chain rules and transform G' as shown in the finiteness algorithm. In the last step, each rule $r : A \to B_1B_2 \ldots B_k \in P$ with $k \geq 3$ is transformed equivalently in the set of rules $\{A \to B_1X_1^{(r)}, X_1^{(r)} \to B_2X_2^{(r)}, \ldots, X_{k-2}^{(r)} \to B_{k-1}B_k\}$. Note that the new nonterminals identify exactly the rule for which they have been used. We construct a *gsm* which nondeterministically inserts between any two consecutive nonterminal symbols it scans either nothing or a sequence $X_1^{(p)}X_2^{(p)} \ldots X_q^{(p)}$, where p is the label of a rule in P and $1 \leq q \leq m-2$, $m \geq 3$ being the length of the right-hand side of the rule p.

For a given word $w = a_1a_2 \ldots a_n \in T^+$ we construct the deterministic finite automaton $\mathcal{A}_w = (Q, N \cup T, \delta, q_0, \{q_f\})$, where

$$Q = \{q_0, q_f\} \cup \{[A, i, j] \mid A \in N, 1 \leq i \leq j \leq n\}.$$

For any pair $1 \leq i \leq j \leq n$ we denote by $N(i,j) = \{A \in N \mid A \Longrightarrow^* a_ia_{i+1} \ldots a_j\}$; these sets can be computed by the CYK algorithm. Then we set

$$\begin{array}{rl}
(i) & \delta([A,i,i],a_i) = q_f \text{ for each } A \in N(i,i),\ 1 \leq i \leq n,\\
(ii) & \delta([A,i,j),B) = [B,i,k] \text{ and } \delta([A,i,j],C) = [C,k+1,j]\\
& \qquad \text{for each } A \in N(i,j),\ 1 \leq i \leq k < j \leq n, \text{ such that}\\
& \qquad B \in N(i,k), C \in N(k+1,j), A \to BC \in P,\\
(iii) & \delta(q_0, A) = [A,1,n] \text{ for all } A \in N(1,n).
\end{array}$$

Clearly, $w \in L_\exists(\gamma)$ iff $L(\mathcal{A}_w) \cap L(G') \neq \emptyset$. As far as the complexity of this algorithm is concerned, the automaton $\mathcal{A}_w$ (which has $O(n^2)$ states) can be constructed in time $O(n^3)$, the number of nonterminals and rules in the context-free grammar G'' generating $L(\mathcal{A}_w) \cap L(G')$ is $O(n^4)$ and $O(n^6)$, respectively. Applying the well-known algorithm, see, e.g., Hopcroft and Ullman (1979), to solve the emptiness of $L(G'')$ we are done. □

19.6 Conclusion

This note tries to complete the picture of path-controlled grammars started in Marcus et al. (2001) with some mathematical properties which are missing from the aforementioned work: closure and decidability properties, including a polynomial recognition algorithm.

There is an interesting difference between our formalism and that considered in Weir (1992) and a series of subsequent works in that the grammars studied here require only the existence of a path labeled with a word in the control set while Weir's formalism requires, in essence, all independence paths to be labeled with words in the control set. On the other hand, all of the properties reported here as well as in Marcus et al. (2001) are quite similar to those of Weir's corresponding classes. In spite of this fact, we were not able to emphasize the substantial differences, if any, between the two formalisms.

Acknowledgments. Some critical suggestions and comments of the referees which improved the presentation are gratefully acknowledged.

References

Abraham, S. 1965. Some Questions of Phrase-Structure Grammars. *Computational Linguistics* 4:61–70.

Bellert, I. 1965. Relational Phrase Structure Grammar and Its Tentative Applications. *Information and Control* 8:503–530.

Culik, K. and H. A. Maurer. 1977. Tree Controlled Grammars. *Computing* 19:120–139.

Dassow, J. and G. Păun. 1989. *Regulated Rewriting in Formal Language Theory*. Berlin: Springer-Verlag.

Ginsburg, S. and E. H. Spanier. 1968. Control Sets on Grarnrnars. *Math. Systems Theory* 2:159–177.

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

Joshi, A. K. 1985. How Much Context-Sensitivity Is Required to Provide Reasonable Structural Descriptions? Tree Adjoining Grammars. In D. R. D. et al, ed., *Natural Language Processing: Psycholinguistic, Computational, and Theoretic Perspectives*. New York: Cambridge Univ. Press.

Kasami, T. 1965. *An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages*. Bedford Mass.: Air Force Cambridge Research Laboratory.

Khabbaz, N. A. 1974. A Geometric Hierarchy of Languages. *J. Comput. System Sci.* 8:142–157.

Kosaraju, S. R. 1982. Decidability of Reachability in Vector Addition Systems. In *14th Symposium on Theory of Computing*.

Kudlek, M., C. Martín-Vide, A. Mateescu, and V. Mitrana. 2003. Contexts and the Concept of Mild Context-Sensitivity. *Linguistics and Philosophy* 26:703–725.

Marcus, S., C. Martín-Vide, V. Mitrana, and G. Păun. 2001. A New-Old Class of Linguistically Motivated Regulated Grammars. In W. Daelemans, K. Sima'an, J. Veenstra, and J. Zavrel, eds., *7th International Conference* *on Computational Linguistics in Netherlands*, Language and Computers-Studies in Practical Linguistics. Amsterdam: Rodopi.

Palis, M. A. and S. M. Shende. 1992. Upper Bounds on Recognition of a Hierarchy of Non-Context-Free Languages. *Theoret. Comput. Sci.* 98:289–319.

Palis, M. A. and S. M. Shende. 1995. Pumping Lemmas for the Control Language Hierarchy. *Math. Systems Theory* 28:199–213.

Thatcher, J. W. 1967. Characterizing Derivation Trees of Context-Free Grammars Through a Generalization of Finite Automata Theory. *J. Comput. System Sci.* 1:317–322.

Weir, D. J. 1992. A Geometric Hierarchy Beyond Context-Free Languages. *Theoret. Comput. Sci.* 104:235–261.

Younger, D. H. 1967. Recognition and Parsing of Context-Free Languages in Time n^3. *Information and Control* 10:189–208.