State Grammars with Stores

Oscar H. Ibarra, Ian Mcquillan

State Grammars with Stores *

Oscar H. Ibarra[1] and Ian McQuillan[2]

[1] Department of Computer Science
University of California, Santa Barbara, CA 93106, USA
ibarra@cs.ucsb.edu
[2] Department of Computer Science, University of Saskatchewan
Saskatoon, SK S7N 5A9, Canada
mcquillan@cs.usask.ca

Abstract. State grammars are context-free grammars where the productions have states associated with them, and can only be applied to a nonterminal if the current state matches the state in the production. Once states are added to grammars, it is natural to add various stores, similar to machine models. With such extensions, productions can only be applied if both the state and the value read from each store matches between the current sentential form and the production. Here, generative capacity results are presented for different derivation modes, with and without additional stores. In particular, with the standard derivation relation, it is shown that adding reversal-bounded counters does not increase the capacity, and states are enough. Also, state grammars with reversal-bounded counters that operate using leftmost derivations are shown to coincide with languages accepted by one-way machines with a pushdown and reversal-bounded counters, and these are surprisingly shown to be strictly weaker than state grammars with the standard derivation relation (and no counters). Complexity results of some decision problems involving state grammars with counters are also studied.

Keywords: grammars, reversal-bounded counters, automata models, matrix grammars

1 Introduction

State grammars were created by Kasai [11], and have context-free grammar rules with additional state components. As originally defined, they consist of a set of nonterminals V, a set of terminals Σ, an initial nonterminal $S \in V$, a set of states Q, an initial state $q_0 \in Q$, and a set of productions P. Instead of normal context-free productions of the form $A \to w$, where $A \in V, w \in (V \cup \Sigma)^*$, now productions are of the form $(q, A) \to (p, w)$, where $q, p \in Q$, and w was forced to be non-empty in Kasai's original formulation. Sentential forms are of the form (q, α) where $q \in Q, \alpha \in (V \cup \Sigma)^*$. A production is only applicable to a sentential form if

* The research of O. H. Ibarra was supported, in part, by NSF Grant CCF-1117708. The research of I. McQuillan was supported, in part, by Natural Sciences and Engineering Research Council of Canada Grant 2016-06172.

the state of the production matches the state of the sentential form. The original derivation relation considered by Kasai (later called the *leftish* derivation relation in [13] which we will call it here as well), was as follows: $(q, uAv) \Rightarrow_{\text{lt}} (p, uwv)$ if $(q, A) \to (p, w) \in P$, and A is the leftmost nonterminal in the sentential form that has a production that is applicable from the current state. A word is generated if there is some leftish derivation starting at the initial state and initial nonterminal that produces a word over Σ^*. The family of languages generated by such systems with λ-free rules, denoted by $\mathcal{L}_{\text{lt}}(\lambda\text{-free-CFG-S})$, was shown to be equal to the family of context-sensitive languages [11]. Later, it was shown that when including λ rules, the family produced, $\mathcal{L}_{\text{lt}}(\text{CFG-S})$, is equal to the family of recursively enumerable languages [15].

The definition of state grammars was extended shortly afterwards by Moriya [12] to also include a final state set F. Furthermore, he defined another derivation relation called the *free interpretation*, whereby any nonterminal can be rewritten that has a production defined on the current state, rather than the leftmost. With this derivation relation, the family of languages generated by state grammars, $\mathcal{L}(\text{CFG-S})$, was proven to equal the languages generated by matrix grammars (or λ-free matrix grammars for λ-free state grammars) [1].

The notion of combining grammars with states is a powerful one. It is then easy to add various stores to grammars that operate like machine models. It can also enable the study of trade-offs between numbers of states, nonterminals, productions, and stores, relevant to the area of descriptional complexity. Changing the derivation relation and the rules allowed can also significantly change the families generated, obtaining many important language families as special cases.

In this paper, we will collate some of the existing generative capacity results on state grammars. In doing so, we provide a shorter alternative proof that state grammars (with the free interpretation) generate the same family as matrix grammars by using context-free grammars with regular control. A new derivation mode is defined where all nonterminals are rewritten from left-to-right until the last nonterminal, then this repeats starting again at the first nonterminal. State grammars with this mode are found to generate the recursively enumerable languages (or context-sensitive languages for λ-free grammars). We will then consider adding multiple reversal-bounded counters to state grammars (with the free interpretation) and find that this does not change the capacity beyond only having states. However, this system provides quite an easy way of describing languages. Furthermore, it is shown that leftmost derivations for state grammars are strictly weaker than leftmost derivations for state grammars with counters, which are then strictly weaker than state grammars with no counters using the free interpretation. Lastly, some complexity results are presented on state grammars with counters. Many proofs are omitted due to space constraints.

2 Preliminaries

Here, some notation used in the paper is presented; we refer to [7] for an introductory treatment of automata and formal languages. We assume knowledge

of deterministic and nondeterministic finite automata, context-free grammars, context-sensitive languages, and the recursively enumerable languages.

An *alphabet* Σ is a finite set of symbols, a *word* over Σ is a finite sequence of symbols $a_1 \cdots a_n$, $n \geq 0, a_i \in \Sigma, 1 \leq i \leq n$, and Σ^* (respectively Σ^+) is the set of all words (non-empty words) over Σ. Then, Σ^* contains the empty word, denoted by λ. Given a word $w \in \Sigma^*$, the length of w is denoted by $|w|$, for $a \in \Sigma$, $|w|_a$ is the number of a's in w, and for subsets X of Σ, $|w|_X = \sum_{a \in X} |w|_a$. The set of letters occurring in w, $\text{alph}(w) = \{a \in \Sigma \mid |w|_a > 0\}$. Given $\Sigma = \{a_1, \ldots, a_k\}$, the Parikh map of w is $\psi(w) = (|w|_{a_1}, \ldots, |w|_{a_k})$, extended to languages L, $\psi(L) = \{\psi(w) \mid w \in L\}$. The commutative closure of L, $\text{comm}(L) = \{v \in \Sigma^* \mid \psi(v) = \psi(w) \text{ for some } w \in L\}$. We will not define the notion of semilinear sets and languages here, but an equivalent definition is that a language L is semilinear if and only if it has the same commutative closure as some regular language [5]. Given $u, v \in \Sigma^*$, the shuffle of u and v, denoted by $u \sqcup\!\sqcup v$ is $\{u_1 v_1 \cdots u_n v_n \mid u = u_1 u_2 \cdots u_n, v = v_1 v_2 \cdots v_n, u_i, v_i \in \Sigma^*, 1 \leq i \leq n\}$.

The context-free languages are denoted by $\mathcal{L}(\mathsf{CFG})$, the linear languages are denoted by $\mathcal{L}(\mathsf{LG})$, the context-sensitive languages by $\mathcal{L}(\mathsf{CS})$, and the right linear (regular languages) are denoted by $\mathcal{L}(\mathsf{REG})$.

Moreover, we will discuss other families and grammars systems summarized in [1], such as matrix grammars. The languages generated by matrix grammars are denoted by $\mathcal{L}(\mathsf{M})$, and the languages generated by λ-free matrix grammars are denoted by $\mathcal{L}(\lambda\text{-free-}\mathsf{M})$.

3 Grammars with States

To start, we will formally define state grammars, following the notation of [12] with final states.

Definition 1. *A* state grammar *(*CFG-S*), is a 7-tuple $G = (V, \Sigma, P, S, Q, q_0, F)$, where V is the finite nonterminal alphabet, Σ is the finite terminal alphabet, $S \in V$ is the initial nonterminal, Q is the finite set of states (V, Σ, Q are disjoint), $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and P is a finite set of productions of the form:*

$$(q, A) \to (p, w),$$

where $A \in V, w \in (V \cup \Sigma)^*, q, p \in Q$. The grammar is said to be linear (and is an LG-S*) if, for all productions $(q, A) \to (p, w)$, $w \in \Sigma^*(V \cup \{\lambda\})\Sigma^*$. The grammar is said to be right linear (and is a* RLG-S*) if, for all productions $(q, A) \to (p, w)$, $w \in \Sigma^*(V \cup \{\lambda\})$. In all cases, G is λ-free if all productions are to some (p, w) where $w \in (V \cup \Sigma)^+$.*

A sentential form *of G is any element of $Q \times (V \cup \Sigma)^*$. Four different methods of derivation will be defined. They are as follows:*

1. *The* free interpretation *derivation relation is defined such that $(q, uAv) \Rightarrow (p, uxv)$ if, $(q, A) \to (p, x) \in P$, and $u, v \in (V \cup \Sigma)^*$. This is extended to the reflexive, transitive closure $\Rightarrow^*$. The language generated by G is*

$$L(G) = \{w \mid (q_0, S) \Rightarrow^* (f, w), f \in F, w \in \Sigma^*\}.$$

2. *The* leftmost derivation *relation is defined such that* $(q, uAv) \Rightarrow_{\mathrm{lm}} (p, uxv)$ *if,* $(q, A) \to (p, x) \in P$, $(q, uA) \Rightarrow (p, ux)$, *and* $u \in \Sigma^*$. *This is extended to the reflexive, transitive closure* $\Rightarrow^*_{\mathrm{lm}}$. *The leftmost language generated by* G *is*

$$L_{\mathrm{lm}}(G) = \{w \mid (q_0, S) \Rightarrow^*_{\mathrm{lm}} (f, w), f \in F, w \in \Sigma^*\}.$$

3. *The* leftish *derivation relation is defined such that* $(q, uAv) \Rightarrow_{\mathrm{lt}} (p, uxv)$ *if,* $(q, A) \to (p, x) \in P$, $(q, uA) \Rightarrow (p, ux)$, *and for all* $B \in \mathrm{alph}(u)$ *with* $B \in V$, *then there is no production from* (q, B). *This is extended to the reflexive, transitive closure* $\Rightarrow^*_{\mathrm{lt}}$. *The leftish language generated by* G *is*

$$L_{\mathrm{lt}}(G) = \{w \mid (q_0, S) \Rightarrow^*_{\mathrm{lt}} (f, w), f \in F, w \in \Sigma^*\}.$$

4. *The* circular *derivation relation is, for* $v_0 A_1 v_1 \cdots A_n v_n, A_i \in V, v_i \in \Sigma^*$, $0 \leq i \leq n$,

$$(p_0, v_0 A_1 v_1 A_2 \cdots A_n v_n) \Rightarrow_\circ (p_1, v_0 x_1 v_1 A_2 \cdots A_n v_n) \Rightarrow_\circ$$
$$(p_2, v_0 x_1 v_1 x_2 v_2 A_3 \cdots A_n v_n) \Rightarrow_\circ \cdots \Rightarrow_\circ (p_n, v_0 x_1 v_1 x_2 v_2 \cdots x_n v_n),$$

where $(p_i, A_{i+1}) \to (p_{i+1}, x_{i+1}) \in P$ *for all* i, $0 \leq i < n$. *In this case, it is written*

$$(p_0, v_0 A_1 v_1 A_2 \cdots A_n v_n) \Rightarrow_\bullet (p_n, v_0 x_1 v_1 x_2 v_2 \cdots x_n v_n).$$

This is extended to $\Rightarrow^*_\bullet$, *the reflexive, transitive closure of* $\Rightarrow_\bullet$. *Therefore, this relation rewrites all nonterminals from left-to-right, then repeats in a circular fashion. The circular language generated by* G *is*

$$L_\bullet(G) = \{w \mid (q_0, S) \Rightarrow^*_\bullet (f, w), f \in F, w \in \Sigma^*\}.$$

We also sometimes associate labels P_Σ *in bijective correspondence with* P, *and write* $u \underset{p}{\Rightarrow} v$, *if production* p *was applied from* u *to* v *(and similarly for the other derivation relations).*

The family of languages generated by CFG-S *grammars with the free interpretation (respectively the leftmost, leftish, and circular) derivation relation is denoted by* $\mathcal{L}$(CFG-S) *(respectively* $\mathcal{L}_{\mathrm{lm}}$(CFG-S), $\mathcal{L}_{\mathrm{lt}}$(CFG-S), $\mathcal{L}_\bullet$(CFG-S)*). For each of these families, we precede the family with* λ*-free to represent those language generated by* λ*-free systems; e.g.* $\mathcal{L}_\bullet$(λ*-free-*CFG-S). *Similarly, replacing* CFG-S *with* LG-S *in these (or* RLG-S*) restricts the families to grammars where the rules are linear (or right linear).*

Example 2. Let $k \geq 2$, $\Sigma = \{a_1, b_1, \ldots, a_k, b_k\}$, and $G_k = (V, \Sigma, P, S, Q, q_0, F)$ where $Q = \{q_0, \ldots, q_{k-1}\}, F = \{q_0\}$, and P contains:

- $(q_0, S) \to (q_0, A_1 A_2 \cdots A_k)$,
- $(q_{i-1}, A_i) \to (q_i, a_i A_i b_i) \mid (q_i, a_i b_i)$, for $1 \leq i < k$,
- $(q_{k-1}, A_k) \to (q_0, a_k A_k b_k) \mid (q_0, a_k b_k)$.

In any successful derivation using the free interpretation, states must follow a pattern in $q_0(q_0 \cdots q_{k-1})^+ q_0$, and from q_{i-1}, only productions on A_i can be applied, and they all must terminate on the last pass. Hence, $L(G_k) = \{a_1^n b_1^n \cdots a_k^n b_k^n \mid n > 0\}$.

For the first comparison, we see that the different derivation relations for linear and right linear grammars with states are the same.

Proposition 3. *The following are true:*

- $\mathcal{L}(\mathsf{LG}) = \mathcal{L}(\mathsf{LG\text{-}S}) = \mathcal{L}_{\mathrm{lm}}(\mathsf{LG\text{-}S}) = \mathcal{L}_{\mathrm{lt}}(\mathsf{LG\text{-}S}) = \mathcal{L}_{\bullet}(\mathsf{LG\text{-}S})$,
- $\mathcal{L}(\mathsf{REG}) = \mathcal{L}(\mathsf{RLG\text{-}S}) = \mathcal{L}_{\mathrm{lm}}(\mathsf{RLG\text{-}S}) = \mathcal{L}_{\mathrm{lt}}(\mathsf{RLG\text{-}S}) = \mathcal{L}_{\bullet}(\mathsf{RLG\text{-}S})$.

Proof. It is obvious that the method of derivation does not matter for linear and right linear grammars.

A linear grammar (resp., a right linear grammar) can easily be simulated by such a grammar with one state. The converse follows by creating nonterminals in $V \times Q$. Then, for all productions of the form $(q, A) \to (p, uBv), q, p \in Q, A, B \in V, u, v \in \Sigma^*$, create a normal production $(q, A) \to u(p, B)v$ (ie. the state stays on the nonterminal; and for all terminating productions of the form $(q, A) \to (p, u), q, p \in Q, A \in V, u \in \Sigma^*$, create $(q, A) \to u$ if and only if $p \in F$. It is clear that the languages generated are the same. □

The following was mentioned in [13], and it follows by considering the standard simulation of context-free grammars with pushdown automata, but using the state of the pushdown to simulate the state of the state grammar.

Proposition 4. $\mathcal{L}_{\mathrm{lm}}(\mathsf{CFG\text{-}S}) = \mathcal{L}(\mathsf{CFG})$.

As proven in [12], the family of languages generated by matrix grammars (respectively λ-free matrix grammars) is equal to the family generated by state grammars (respectively λ-free state grammars) with the free interpretation. An alternate, shorter proof can also be demonstrated by showing equivalence of state grammars to context-free grammars with regular control [1]. It is known that such grammars are equivalent to matrix grammars [1].

Proposition 5. $\mathcal{L}(\mathsf{CFG\text{-}S}) = \mathcal{L}(\mathsf{M})$, *and* $\mathcal{L}(\lambda\text{-}\mathit{free}\text{-}\mathsf{CFG\text{-}S}) = \mathcal{L}(\lambda\text{-}\mathit{free}\text{-}\mathsf{M})$.

When circular derivations are used, then it will be seen next that CFG-S grammars already generate all recursively enumerable languages. We use the notion of a complete derivation tree t of a context-free grammar [7], which is a tree where all nodes are labelled by either a nonterminal, a terminal, or λ, the root is labelled by the initial nonterminal, if a parent is labelled by A and its children are labelled by $A_1, \ldots, A_k$ from left to right, then $A \to A_1 \cdots A_k$ is a production, if a node is labelled by λ, then it is the only child of its parent, and all leaves are labelled by terminals. The yield of a derivation tree, $\mathrm{yd}(t)$, is the sequence of terminals obtained via a preorder traversal. Given such a tree t, level i is all nodes at distance i from the root, and the level-i word is the sequence of labels on the nodes of level i concatenated together from left to right. It is

known that the set of yields of complete derivation trees of a grammar is exactly the language generated by the grammar [7].

State grammars with the circular derivation will be shown equivalent to *tree controlled grammars* which are defined as follows. A tree controlled grammar is a tuple $G = (V, \Sigma, P, S, R)$, where $G' = (V, \Sigma, P, S)$ is a CFG, and R is a regular language over $V \cup \Sigma$. When considering context-free derivation trees in G', a restriction on the trees is used as follows: the language generated by G, $L(G)$, is equal to

$$\{\mathrm{yd}(t) \mid t \text{ is a complete derivation tree of } G', \text{ for all levels } i \text{ but the last, the level-}i \text{ word is in } R\}.$$

Let $\mathcal{L}(\mathsf{TREE})$ (respectively $\mathcal{L}(\lambda\text{-free-TREE})$) be the family of languages generated by (respectively λ-free) tree controlled grammars. It is known that tree controlled grammars generate all recursively enumerable languages, and λ-free tree controlled grammars generate the context-sensitive languages [1]. Equivalence to tree controlled grammars therefore implies:

Proposition 6. *The following are true:*

- $\mathcal{L}_\bullet(\mathsf{CFG\text{-}S}) = \mathcal{L}_{\mathrm{lt}}(\mathsf{CFG\text{-}S}) = \mathcal{L}(\mathsf{TREE}) = \mathcal{L}(\mathsf{RE})$,
- $\mathcal{L}_\bullet(\lambda\text{-}\textit{free}\text{-}\mathsf{CFG\text{-}S}) = \mathcal{L}_{\mathrm{lt}}(\lambda\text{-}\textit{free}\text{-}\mathsf{CFG\text{-}S}) = \mathcal{L}(\lambda\text{-}\textit{free}\text{-}\mathsf{TREE}) = \mathcal{L}(\mathsf{CS})$.

Hence, the following hierarchies are obtained:

Corollary 7. *The following are true:*

- $\mathcal{L}(\mathsf{CFG}) = \mathcal{L}_{\mathrm{lm}}(\mathsf{CFG\text{-}S}) \subsetneq \mathcal{L}(\mathsf{CFG\text{-}S}) = \mathcal{L}(\mathsf{M}) \subsetneq \mathcal{L}_\bullet(\mathsf{CFG\text{-}S}) = \mathcal{L}_{\mathrm{lt}}(\mathsf{CFG\text{-}S}) = \mathcal{L}(\mathsf{RE})$,
- $\mathcal{L}(\lambda\text{-}\textit{free}\text{-}\mathsf{CFG}) = \mathcal{L}_{\mathrm{lm}}(\lambda\text{-}\textit{free}\text{-}\mathsf{CFG\text{-}S}) \subsetneq \mathcal{L}(\lambda\text{-}\textit{free}\text{-}\mathsf{CFG\text{-}S}) = \mathcal{L}(\lambda\text{-}\textit{free}\text{-}\mathsf{M}) \subsetneq \mathcal{L}_\bullet(\lambda\text{-}\textit{free}\text{-}\mathsf{CFG\text{-}S}) = \mathcal{L}_{\mathrm{lt}}(\lambda\text{-}\textit{free}\text{-}\mathsf{CFG\text{-}S}) = \mathcal{L}(\mathsf{CS})$.

4 State Grammars with Stores

Now that states are attached to grammars, it is quite natural to attach one or more stores as well, just like machine models. Then, store contents can be part of sentential forms just as states are with state grammars. For example, one could define context-free grammars with states plus a pushdown store. This would be represented with a tuple just like a CFG-S but with an additional word over the pushdown alphabet Γ and a bottom-of-pushdown marker Z_0. In particular, the productions would be of the form $(q, X, A) \to (p, \alpha, w)$, where q, p are states, A is a nonterminal, w is over the nonterminal and terminal alphabets, X is the topmost symbol of the pushdown, and α is the string to replace the topmost symbol of the pushdown. Sentential forms are therefore in $Q \times \Gamma^+ \times (V \cup \Sigma)^*$, and the derivation relation is defined in the obvious way. Here, we will attach multiple reversal-bounded counters as stores as they are defined with reversal-bounded counter machines [9]. Explained briefly, a one-way k-counter machine is an NFA with k counters, each containing some non-negative integer,

and the transition function can detect whether each counter is empty or not, and can increment, keep the same, or decrement each counter by one. Such a machine is r-reversal-bounded if the number of times each counter switches between non-decreasing and non-increasing is at most r. Then $\mathcal{L}(\mathsf{NCM})$ is the family of languages accepted by machines that are r-reversal-bounded k-counter machines, for some $k, r \geq 1$.

Since grammars with states using either circular or leftish derivations already generate all recursively enumerable languages, we will not consider those derivation relations with stores.

Denote the set of all context-free grammars with states and some number of reversal-bounded counters by CFG-SC, and the languages they generate with the free interpretation and the leftmost derivation modes by $\mathcal{L}(\mathsf{CFG\text{-}SC})$ and $\mathcal{L}_{\mathrm{lm}}(\mathsf{CFG\text{-}SC})$ respectively. For each such grammar $G = (V, \Sigma, P, S, Q, q_0, F)$ with k counters, productions are of the form $(q, i_1, \ldots, i_k, A) \to (p, l_1, \ldots, l_k, w)$, where $p, q \in Q, i_j \in \{0, 1\}$ (a production with $i_j = 0$ is applied if and only if counter j is 0), $l_j \in \{-1, 0, +1\}$ (which changes the counter), $A \in V, w \in (V \cup \Sigma)^*$.

Example 8. Let $G = (V, \{a, b\}, P, S, Q, q_0, \{q_f\})$ be a CFG-SC with 2 counters accepting $\{w\$w \mid |w|_a = |w|_b \geq 0\}$, where P is as follows:

- $(q_0, 0, 0, S) \to (q_0, 0, 0, A_1A_2)$,
- $(q_0, i, j, A_1) \to (q_a, 1, 0, aA_1) \mid (q_b, 0, 1, bA_1)$, for $i, j \in \{0, 1\}$,
- $(q_a, i, j, A_2) \to (q_0, 0, 0, aA_2)$, $(q_b, i, j, A_2) \to (q_0, 0, 0, bA_2)$, for $i, j \in \{0, 1\}$,
- $(q_0, i, i, A_1) \to (q_1, 0, 0, A_1)$, for $i \in \{0, 1\}$,
- $(q_1, 1, 1, A_1) \to (q_1, -1, -1, A_1)$,
- $(q_1, 0, 0, A_2) \to (q_1, 0, 0, \lambda)$, $(q_1, 0, 0, A_1) \to (q_f, 0, 0, \$)$.

To start, G switches to $(q_0, 0, 0, A_1A_2)$. Then the derivation repeatedly guesses either that A_1 derives an a or a b; if it guesses it derives an a, it switches to q_a and increases the first counter, and then from q_a, only A_2 can be rewritten and it must derive an a (similarly with the b case using q_b and the second counter). Therefore, A_1 derives some sequence of terminals w and A_2 must derive the same sequence, and the first counter contains $|w|_a$ and the second contains $|w|_b$. At any point while in state q_0, G can switch to q_1 which repeatedly decreases both counters in parallel until both are verified to be zero at the same time, at which point both A_2 and A_1 are erased.

Before studying the generative capacity of CFG-SC, a definition is required. A CFG-SC G is in normal form if each counter makes exactly 1 reversal (once they decrease, they can no longer increase), and a terminal string is successfully generated when G enters a unique accepting state f and all the counters are zero. Moreover, at each step at most one counter is changed (i.e. , $+1$ or -1). We also assume that the state remembers when a counter enters a decreasing mode. So, e.g., when counter i enters the decreasing mode, the state remembers that from that point on, counter i can no longer increase. When another counter j enters the decreasing mode, the state now remembers that counters i and j can no longer increase, etc.

Lemma 9. *Let G be an* CFG-SC*. We can effectively construct a* CFG-SC *G' in normal form such that $L(G) = L(G')$.*

Hence, we may assume that a CFG-SC is in normal form.

Next, it will be seen that reversal-bounded counters do not increase the generative capacity.

Proposition 10. $\mathcal{L}(\text{CFG-S}) = \mathcal{L}(\text{CFG-SC}) = \mathcal{L}(\text{M})$.

Proof. By [12] (and Proposition 5), $\mathcal{L}(\text{CFG-S}) = \mathcal{L}(\text{M})$, and clearly $\mathcal{L}(\text{CFG-S}) \subseteq \mathcal{L}(\text{CFG-SC})$.

Let G be such a CFG-SC. Assume without loss of generality that G is in normal form, and it therefore has k 1-reversal bounded counters. Make a state grammar G' (without counters) over $\Sigma \cup \Delta$ where $\Delta = \{c_1, d_1, \cdots, c_k, d_k\}$ are new symbols. Then, G' simulates G but, whenever it adds from counter i, it instead outputs terminal symbol c_i, and whenever it decreases from counter i, it outputs d_i. The states of G' also verify that G' starts by, for each counter i, simulating only productions associated with counter i being empty until it adds to the counter for the first time, then it simulates productions defined on counter i being positive (while outputting c_i's), then simulates productions on counter i being positive (while outputting d_i's), until some nondeterministically guessed spot after outputting some d_i, where it guesses that the counter is now empty, and then it only simulates productions on counter i being empty while not outputting any more c_i's and d_i's. G' operates in this fashion, as states were specifically marked in the normal form. Therefore, G' operates just like G, where it simulates all of the counters, making sure that for each counter, all additions occur before any subtractions, but it does not do any of the counting. If one then restricted the derivations of G' to those where the number of increases is the same as the number of decreases for each counter, then after erasing the letters of Δ, it would give $L(G)$. But, consider the following regular language $R = (c_1 d_1)^* \cdots (c_k d_k)^* \Sigma^*$, and the commutative closure of R, $\text{comm}(R)$. Let h be a homomorphism that erases all letters of Δ and fixes all letters of Σ. Then, $h(L(G') \cap \text{comm}(R))$ is exactly this language, where $L(G') \cap \text{comm}(R)$ restricts words to only those that have the same number of c_i's as d_i's (and hence the same number of increases as decreases for each counter), and h erases the letters of Δ. Hence, $L(G) = h(L(G') \cap \text{comm}(R))$.

Since G' is a normal grammar with states, it can be converted to a matrix grammar G'' by Proposition 5. It is known that matrix grammars are closed under intersection with $\mathcal{L}(\text{NCM})$ [16] (there, they used closure under the BLIND multicounter languages which is equivalent to $\mathcal{L}(\text{NCM})$ [3]). Also, the commutative closure of every regular language is in $\mathcal{L}(\text{NCM})$ [8]. So $L''' = L(G'') \cap \text{comm}(R)$ is a language generated by a matrix grammar. Lastly, erasing all c_i's and d_i's with h gives $L(G)$, and matrix grammars are closed under homomorphism [1]. Since this is a matrix grammar, it can be converted back to a normal state grammar by Proposition 5 generating the same language as $L(G)$. □

To note, in the proof above, in G, despite the counters being 1-reversal-bounded, productions can be applied to any nonterminal in the sentential form. Thus, some counter additions could occur when rewriting a nonterminal to the right of other nonterminals that get rewritten with a production that decreases. Hence, when intersecting with an $\mathcal{L}(\mathsf{NCM})$ language, it must not enforce that all c_i's occur before any d_i's.

The following corollary is true, since the results are known to be true for matrix grammars [6].

Corollary 11. *The following are true:*

1. *Every unary language generated by a* CFG-SC *is regular.*
2. *The emptiness problem for* CFG-SC *is decidable.*

Next, we will study leftmost derivations of CFG-SC's. We will show that $\mathcal{L}_{\rm lm}(\mathsf{CFG\text{-}SC}) = \mathcal{L}(\mathsf{NPCM}) \subsetneq \mathcal{L}(\mathsf{CFG\text{-}SC})$, where NPCMs are one-way nondeterministic pushdown automata augmented by reversal-bounded counters [9]. To help, we need the notion of a CFG with monotonic counters introduced in [10]. This is a simpler model of grammars with counters that do not have states. At each step in the derivation, the counters can be incremented by 0 or +1, but not decremented. A derivation in this grammar starts with the counters having value zero. A terminal string w is in the language of the grammar if there is a derivation of w that ends with all counters having the same value.

A CFG with *monotonic counters* (CFG-MC) is a 5-tuple, $G = (\Sigma, V, S, k, P)$, where Σ is the set of terminals, V is the set of nonterminals, $S \in V$ is the initial nonterminal, k is the number of monotonic counters, all are initially set to 0, and P is the set of rules of the form: $A \to (z, c_1, \ldots, c_k)$, where $A \in V$, $z \in (V \cup \Sigma)^*$ and $c_i = 0$ or $+1$.

The language defined is $L(G) = \{w \mid w \in \Sigma^*, (S, 0, \ldots, 0) \Rightarrow^* (w, n, \ldots, n) \text{ for some } n \geq 0\}$.

The following result was shown in [10]:

Proposition 12. $\mathcal{L}(\mathsf{NPCM})$ *is equal to the family of languages generated by* CFG-MC*s (using either the leftmost derivation relation or the normal derivation relation).*

From this, the following can be shown:

Lemma 13. $\mathcal{L}(\mathsf{NPCM}) = \mathcal{L}_{\rm lm}(\mathsf{CFG\text{-}SC})$.

Proof. Every CFG-MC G with a leftmost derivation can be simulated by a CFG-SC with a leftmost derivation. It starts by simulating with one state. Then, before terminating, it guesses all counters are equal, decreases them all to verify, then terminates. Thus, $\mathcal{L}(\mathsf{NPCM}) \subseteq \mathcal{L}_{\rm lm}(\mathsf{CFG\text{-}SC})$. For the reverse containment, consider the standard simulation on a CFG with an NPDA with a leftmost derivation. This same construction can work with states while the counters of the CFG-SC can be simulated by the counters of the NPCM. □

Lemma 14. $\mathcal{L}(\mathsf{NPCM}) \subsetneq \mathcal{L}(\mathsf{M})$.

Proof. It is clear that $\mathcal{L}(\mathsf{CFL}) \subseteq \mathcal{L}(\mathsf{M})$, it is known that $\mathcal{L}(\mathsf{M})$ is closed under intersection with $\mathcal{L}(\mathsf{NCM})$ [16], and homomorphism [1]. Recently, a Chomsky-Schützenberger-like theorem was shown that demonstrates that every language in NPCM can be obtained by some Dyck language (which is context-free) intersected with an $\mathcal{L}(\mathsf{NCM})$ language, then mapped via a homomorphism [10]. Therefore, $\mathcal{L}(\mathsf{NPCM}) \subseteq \mathcal{L}(\mathsf{M})$.

It is known that every NPCM language is semilinear [9]. Now $\mathcal{L}(\mathsf{CFG\text{-}SC}) = \mathcal{L}(\mathsf{M})$ by Proposition 5. It known that matrix grammars can generate non-semilinear languages, e.g., a matrix grammar can generate the non-semilinear language [1], $L = \{a^n b^m \mid 1 \leq n < m \leq 2^n\}$. It follows that $\mathcal{L}(\mathsf{NPCM}) \subsetneq \mathcal{L}(\mathsf{M})$. □

Therefore, the following hierarchy exists by Propositions 4, 5, 6, 10 and Lemmas 13 and 14.

Proposition 15. *The following is true:*

$$\mathcal{L}(\mathsf{CFG}) = \mathcal{L}_{\mathrm{lm}}(\mathsf{CFG\text{-}S}) \subsetneq \mathcal{L}_{\mathrm{lm}}(\mathsf{CFG\text{-}SC}) = \mathcal{L}(\mathsf{NPCM}) \subsetneq \mathcal{L}(\mathsf{CFG\text{-}S}) = \mathcal{L}(\mathsf{CFG\text{-}SC}) = \mathcal{L}(\mathsf{M}) \subsetneq \mathcal{L}_{\mathrm{lt}}(\mathsf{CFG\text{-}S}) = \mathcal{L}_{\bullet}(\mathsf{CFG\text{-}S}) = \mathcal{L}(\mathsf{RE}).$$

Lastly, we study the emptiness problem for a restriction of CFG-SCs. A CFG-S $G = (V, \Sigma, P, S, Q, q_0, F)$ is of index m, $(m \geq 1)$ if, for every $w \in L(G)$, there exists some derivation $(p_0, \alpha_0) \Rightarrow (p_1, \alpha_1) \Rightarrow \cdots \Rightarrow (p_n, \alpha_n), p_0 = q_0, \alpha_0 = S, p_n \in F, \alpha_n = w$ with $|\alpha_i|_V \leq m$, for all $0 \leq i \leq n$. If it is index m for some m, then it is finite-index. This property is more general than requiring that every derivation of a word in the language has at most m nonterminals, a notion that is called uncontrolled index m, or uncontrolled finite-index. For context-free grammars, the languages generated by uncontrolled finite-index grammars corresponds to pushdown automata with a reversal-bounded pushdown [2], which cannot accept languages such as $\{a^n b^n \mid n > 0\}^*$ that can be generated by an index 2 grammar [14]). Hence, finite-index CFG-S are quite general. Clearly, the definitions easily extend to m-index CFG-SC (finite-index CFG-SC).

Proposition 16. *Let $m, k \geq 1$ be fixed. The emptiness problem for m-index* CFG-SC *with at most k 1-reversal counters is decidable in polynomial time.*

Proof. Let G be an m-index CFG-SC with at most k 1-reversal counters. We first construct from G a grammar G' where all terminal symbols are mapped to λ. Clearly, $L(G')$ is empty if and only if $L(G)$ is empty. Then all rules in G' are of the form:

$$(q, i_1, \ldots, i_k, A) \to (p, l_1, \ldots, l_k, u),$$

where $i_j \in \{0, 1\}, l_j \in \{-1, 0, 1\}$ for $1 \leq j \leq k$, u is string of nonterminals of length at most m (possibly λ; there are no terminals used). Now from G', we construct an NCM M with k 1-reversal counters as follows: Its initial state is $[q_0, S]$, where q_0 is the initial state of G' and S is the start nonterminal of G'. The other states of M are of the form $[q, w]$, where q is a state of G', and w is a string of at most m nonterminals (possibly λ).

Then M starts in state $[q_0, S]$ with all its k counters zero. A move of M is defined by: if G' has a rule

$$(q, i_1, \ldots, i_k, A) \to (p, l_1, \ldots, l_k, v),$$

$i_j \in \{0, 1\}, l_j \in \{-1, 0, 1\}$ for $1 \leq j \leq k$, then in M, for all strings xAy with $|xAy| \leq m$ and $|xvy| \leq m$, create transitions from state $[q, xAy]$ and counter status $i_1, \ldots, i_k$ on λ, that goes to state $[p, xvy]$ and updates the counters by $l_1, \ldots, l_k$. The accepting states of M are of the form $[p, \lambda]$, where p is an accepting state of G'.

Clearly, since G (and, hence, G') is m-index and m is fixed, the size of M is polynomial in the size of G' (hence, of G). Since M is an NCM with a fixed (k) number of 1-reversal counters and it is known that the emptiness problem for NCM with a fixed number of 1-reversal counters is decidable in polynomial time [4], the result follows. □

Note that if the index of G is not fixed, the size of M is no longer polynomial in the size of G' (and, hence, of G). We also note that Proposition 16 can be generalized to hold for many m-index grammar systems with k 1-reversal counters (for fixed m and k).

5 Conclusions and Future Directions

State grammars were studied, and it was shown that with a new circular derivation relation, they generate all recursively enumerable languages. Then, using the free interpretation and the leftmost derivation relations, additional stores were added to state grammars; in particular, some number of reversal-bounded counters. When using the free interpretation derivation relation, the counters do not add any generative capacity, and only states are needed. When using leftmost derivations, the class coincides with the machine model NPCM (pushdown automata with reversal-bounded counters). This leads to the result that state grammars with counters and leftmost derivations are strictly weaker than state grammars with no counters and the free interpretation derivation relation.

There are many interesting problems of descriptional complexity that are open. For example, do state grammars form an infinite hierarchy with the number of states? We conjecture that in Example 2, for each $k \geq 2$, it is impossible to generate L_k with a state grammar with fewer than k states, which would form such a hierarchy.

References

1. Dassow, J., Păun, G.: Regulated Rewriting in Formal Language Theory. EATCS Monographs on Theoretical Computer Science, Springer (1989)
2. Ginsburg, S., Spanier, E.: Finite turn pushdown automata. SIAM Journal of Control 4(3), 429–453 (1966)

3. Greibach, S.: Remarks on blind and partially blind one-way multicounter machines. Theoretical Computer Science 7, 311–324 (1978)
4. Gurari, E.M., Ibarra, O.H.: The complexity of decision problems for finite-turn multicounter machines. Journal of Computer and System Sciences 22(2), 220–229 (1981)
5. Harrison, M.: Introduction to Formal Language Theory. Addison-Wesley series in computer science, Addison-Wesley Pub. Co. (1978)
6. Hauschildt, D., Jantzen, M.: Petri net algorithms in the theory of matrix grammars. Acta Informatica 31(8), 719–728 (1994)
7. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, MA (1979)
8. Ibarra, O., McQuillan, I.: The effect of end-markers on counter machines and commutativity. Theoretical Computer Science 627, 71–81 (2016)
9. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM 25(1), 116–133 (1978)
10. Ibarra, O.H.: Grammatical characterizations of NPDAs and VPDAs with counters. In: Han, Y.S., Salomaa, K. (eds.) Lecture Notes in Computer Science. 21st International Conference on Implementation and Application of Automata, CIAA 2016, Seoul, South Korea, vol. 9705, p. 11 (2016), invited abstract, journal version submitted
11. Kasai, T.: An hierarchy between context-free and context-sensitive languages. Journal of Computer and System Sciences 4(5), 492–508 (1970)
12. Moriya, E.: Some remarks on state grammars and matrix grammars. Information and Control 23, 48–57 (1973)
13. Moriya, E., Hofbauer, D., Huber, M., Otto, F.: On state-alternating context-free grammars. Theoretical Computer Science 337(1), 183–216 (2005)
14. Rozenberg, G., Vermeir, D.: On the effect of the finite index restriction on several families of grammars. Information and Control 39, 284–302 (1978)
15. Salomaa, A.: Matrix grammars with a leftmost restriction. Information and Control 20(2), 143–149 (1972)
16. Stiebe, R.: Slender matrix languages, pp. 375–385. World Scientific (2000)