

Parikh's Theorem: A simple and direct automaton construction

Javier Esparza^a, Pierre Ganty^{b,1,*}, Stefan Kiefer^c, Michael Luttenberger^a

^a*Institut für Informatik, Technische Universität München, 85748 Garching, Germany*

^b*IMDEA Software Institute, Madrid, Spain*

^c*Oxford University Computing Laboratory, Oxford, UK*

Abstract

Parikh's theorem states that the Parikh image of a context-free language is semilinear or, equivalently, that every context-free language has the same Parikh image as some regular language. We present a very simple construction that, given a context-free grammar, produces a finite automaton recognizing such a regular language.

The *Parikh image* of a word w over an alphabet $\{a_1, \dots, a_n\}$ is the vector $(v_1, \dots, v_n) \in \mathbb{N}^n$ such that v_i is the number of occurrences of a_i in w . For example, the Parikh image of $a_1a_1a_2a_2$ over the alphabet $\{a_1, a_2, a_3\}$ is $(2, 2, 0)$. The Parikh image of a language is the set of Parikh images of its words. Parikh images are named after Rohit Parikh, who in 1966 proved a classical theorem of formal language theory which also carries his name. Parikh's theorem [1] states that the Parikh image of any context-free language is *semilinear*. Since semilinear sets coincide with the Parikh images of regular languages, the theorem is equivalent to the statement that every context-free language has the same Parikh image as some regular language. For instance, the language $\{a^n b^n \mid n \geq 0\}$ has the same Parikh image as $(ab)^*$. This statement is also often referred to as Parikh's theorem, see e.g. [10], and in fact it has been considered a more natural formulation [14].

Parikh's proof of the theorem, as many other subsequent proofs [8, 14, 13, 9, 10, 2], is constructive: given a context-free grammar G , the proof produces (at least implicitly) an automaton or regular expression whose language has the same Parikh image as $L(G)$. However, the constructions are relatively complicated, not given in detail, or they yield crude upper bounds, namely automata

*Corresponding author

Email addresses: esparza@model.in.tum.de (Javier Esparza), pierre.ganty@imdea.org (Pierre Ganty), stefan.kiefer@comlab.ox.ac.uk (Stefan Kiefer), lутtenbe@in.tum.de (Michael Luttenberger)

¹This author was sponsored by the Comunidad de Madrid's Program PROMETIDOS-CM (S2009TIC-1465), by the PEOPLE-COFUND's program AMAROUT (PCOFUND-2008-229599), and by the Spanish Ministry of Science and Innovation (TIN2010-20639).

of size $\mathcal{O}(n^n)$ for grammars in Chomsky normal form with n variables (see Section 4 for a detailed discussion). In this note we present an explicit and very simple construction that yields an automaton with $\mathcal{O}(4^n)$ states for grammars in Chomsky normal form, for a lower bound of $\Omega(2^n)$. An application of the automaton is briefly discussed in Section 3: the automaton can be used to algorithmically derive the semilinear set, and, using recent results on Parikh images of NFAs [16, 11], it leads to the best known upper bounds on the size of the semilinear set for a given context-free grammar.

1. The Construction

We follow the notation of [3, Chapter 5]. Let $G = (V, T, P, S)$ be a context-free grammar with a set $V = \{A_1, \dots, A_n\}$ of *variables* or *nonterminals*, a set T of *terminals*, a set $P \subseteq V \times (V \cup T)^*$ of *productions*, and an *axiom* $S \in V$. We construct a nondeterministic finite automaton (NFA) whose language has the same Parikh image as $L(G)$. The transitions of this automaton will be labeled with words of T^* , but note that by adding intermediate states (when the words have length greater than one) and removing ϵ -transitions (i.e., when the words have length zero), such an NFA can be easily brought in the more common form where transition labels are elements of T .

We need to introduce a few notions. Given $\alpha \in (V \cup T)^*$, we denote by $\Pi_V(\alpha)$ (resp. $\Pi_T(\alpha)$) the Parikh image of α where the components not in V (resp. T) have been projected away. Moreover, we denote by $\alpha_{/V}$ (resp. $\alpha_{/T}$) the projection of α onto V (resp. T). For instance, if $V = \{A_1, A_2\}$, $T = \{a, b, c\}$, and $\alpha = aA_2bA_1A_1$, then $\Pi_V(\alpha) = (2, 1)$, $\Pi_T(\alpha) = (1, 1, 0)$ and $\alpha_{/T} = ab$. Given $\alpha, \beta \in (V \cup T)^*$, let $\mathcal{P}(\alpha, \beta)$ be the set of productions of G that can transform α into β , i.e., $\mathcal{P}(\alpha, \beta) = \{(A \rightarrow \gamma) \in P \mid \exists \alpha_1, \alpha_2 \in (V \cup T)^* : \alpha = \alpha_1 A \alpha_2 \wedge \beta = \alpha_1 \gamma \alpha_2\}$. If $\mathcal{P}(\alpha, \beta) \neq \emptyset$ then we call (α, β) a *step*, denoted by $\alpha \Rightarrow \beta$.

The NFA whose language has the same Parikh image as $L(G)$ will be a member of the following family:

Definition 1.1. Let $G = (V, T, P, S)$ be a context-free grammar, let $n = |V|$, and let $k \geq 1$. The k -Parikh automaton of G is the NFA $M_G^k = (Q, T^*, \delta, q_0, \{q_f\})$ defined as follows:

- $Q = \{(x_1, \dots, x_n) \in \mathbb{N}^n \mid \sum_{i=1}^n x_i \leq k\}$;
- $\delta = \{(\Pi_V(\alpha), \gamma_{/T}, \Pi_V(\beta)) \mid \exists (A \rightarrow \gamma) \in \mathcal{P}(\alpha, \beta) : \Pi_V(\alpha), \Pi_V(\beta) \in Q\}$;
- $q_0 = \Pi_V(S)$;
- $q_f = \Pi_V(\varepsilon) = (0, \dots, 0)$.

It is easily seen that M_G^k has exactly $\binom{n+k}{n}$ states. Figure 1 shows the 3-Parikh automaton of the context-free grammar with productions $A_1 \rightarrow A_1A_2|a$, $A_2 \rightarrow bA_2aA_2|cA_1$ and axiom A_1 . The states are all pairs (x_1, x_2) such that

$x_1 + x_2 \leq 3$. For instance, transition $(0, 2) \xrightarrow{ba} (0, 3)$ comes (among others) from the step $A_2 A_2 \Rightarrow b A_2 a A_2 A_2$, and can be interpreted as follows: applying the production $A_2 \rightarrow b A_2 a A_2$ to a word with zero occurrences of A_1 and two occurrences of A_2 leads to a word with one new occurrence of a and b , zero occurrences of A_1 , and three occurrences of A_2 .

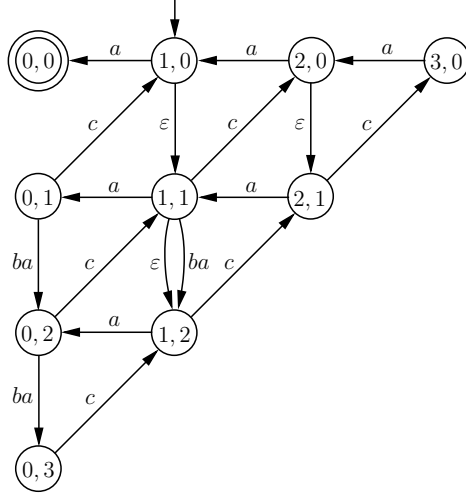


Figure 1: The 3-Parikh automaton of $A_1 \rightarrow A_1 A_2 | a$, $A_2 \rightarrow b A_2 a A_2 | c A_1$ with $S = A_1$.

We define the *degree* of G by $m := -1 + \max\{|\gamma|_V : (A \rightarrow \gamma) \in P\}$; i.e., $m + 1$ is the maximal number of variables on the right hand sides of the productions. For instance, the degree of the grammar in Fig. 1 is 1. Notice that if G is in Chomsky normal form then $m \leq 1$, and $m \leq 0$ iff G is regular.

In the rest of the note we prove:

Theorem 1.1. *If G is a context-free grammar with n variables and degree m , then $L(G)$ and $L(M_G^{nm+1})$ have the same Parikh image.*

For the grammar of Figure 1 we have $n = 2$ and $m = 1$, and Theorem 1.1 yields $L(G) = L(M_G^3)$. So the language of the automaton of the figure has the same Parikh image as the language of the grammar.

Using standard properties of binomial coefficients, for M_G^{nm+1} and $m \geq 1$ we get an upper bound of $2 \cdot (m+1)^n \cdot e^n$ states. For $m \leq 1$ (e.g. for grammars in Chomsky normal form), the automaton M_G^{n+1} has $\binom{2n+1}{n} \leq 2^{2n+1} \in \mathcal{O}(4^n)$ states. On the other hand, for every $n \geq 1$ the grammar G_n in Chomsky normal with productions $\{A_k \rightarrow A_{k-1} A_{k-1} \mid 2 \leq k \leq n\} \cup \{A_1 \rightarrow a\}$ and axiom $S = A_n$ satisfies $L(G_n) = \{a^{2^{n-1}}\}$, and therefore the smallest Parikh-equivalent NFA has $2^{n-1} + 1$ states. This shows that our construction is close to optimal.

2. The Proof

Given $L_1, L_2 \subseteq T^*$, we write $L_1 =_{\Pi} L_2$ (resp. $L_1 \subseteq_{\Pi} L_2$) to denote that the Parikh image of L_1 is equal to (resp. included in) the Parikh image of L_2 . Also, given $w, w' \in T^*$, we abbreviate $\{w\} =_{\Pi} \{w'\}$ to $w =_{\Pi} w'$.

We fix a context-free grammar $G = (V, T, P, S)$ with n variables and degree m . In terms of the notation we have just introduced, we have to prove $L(G) =_{\Pi} L(M_G^{nm+1})$. One inclusion is easy:

Proposition 2.1. *For every $k \geq 1$ we have $L(M_G^k) \subseteq_{\Pi} L(G)$.*

PROOF. Let $k \geq 1$ arbitrary, and let $q_0 \xrightarrow{\sigma} q$ be a run of M_G^k on the word $\sigma \in T^*$. We first claim that there exists a step sequence $S \Rightarrow^* \alpha$ satisfying $\Pi_V(\alpha) = q$ and $\Pi_T(\alpha) = \Pi_T(\sigma)$. The proof is by induction on the length ℓ of $q_0 \xrightarrow{\sigma} q$. If $\ell = 0$, then $\sigma = \varepsilon$, and we choose $\alpha = S$, which satisfies $\Pi_V(S) = q_0$ and $\Pi_T(S) = (0, \dots, 0) = \Pi_T(\varepsilon)$. If $\ell > 0$, then let $\sigma = \sigma' \gamma$ and $q_0 \xrightarrow{\sigma'} q' \xrightarrow{\gamma} q$. By induction hypothesis there is a step sequence $S \Rightarrow^* \alpha'$ satisfying $\Pi_V(\alpha') = q'$ and $\Pi_T(\alpha') = \Pi_T(\sigma')$. Moreover, since $q' \xrightarrow{\gamma} q$ is a transition of M_G^k , there is a production $A \rightarrow \gamma'$ and a step $\alpha_1 A \alpha_2 \Rightarrow \alpha_1 \gamma' \alpha_2$ such that $\Pi_V(\alpha_1 A \alpha_2) = q'$, $\Pi_V(\alpha_1 \gamma' \alpha_2) = q$ and $\gamma'_{/T} = \gamma$. Since $\Pi_V(\alpha') = q' = \Pi_V(\alpha_1 A \alpha_2)$, α' contains at least one occurrence of A , i.e. $\alpha' = \alpha'_1 A \alpha'_2$ for some α'_1, α'_2 . We choose $\alpha = \alpha'_1 \gamma' \alpha'_2$, and get $\Pi_V(\alpha) = \Pi_V(\alpha'_1 \gamma' \alpha'_2) = \Pi_V(\alpha'_1 A \alpha'_2) - \Pi_V(A) + \Pi_V(\gamma') = \Pi_V(\alpha') - \Pi_V(A) + \Pi_V(\gamma') = \Pi_V(\alpha_1 A \alpha_2) - \Pi_V(A) + \Pi_V(\gamma') = \Pi_V(\alpha_1 \gamma' \alpha_2) = q$. Also $\Pi_T(\alpha) = \Pi_T(\alpha'_1 \gamma' \alpha'_2) = \Pi_T(\alpha'_1 A \alpha'_2) + \Pi_T(\gamma') = \Pi_T(\alpha') + \Pi_T(\gamma') = \Pi_T(\sigma') + \Pi_T(\gamma') = \Pi_T(\sigma)$. This concludes the proof of the claim.

Now, let σ be an arbitrary word with $\sigma \in L(M_G^k)$. Then there is a run $q_0 \xrightarrow{\sigma} \Pi_V(\varepsilon)$. By the claim there exists a step sequence $S \Rightarrow^* \alpha$ satisfying $\Pi_V(\alpha) = (0, \dots, 0)$ and $\Pi_T(\alpha) = \Pi_T(\sigma)$. So $\alpha \in T^*$, and hence $\alpha \in L(G)$. Since $\Pi_T(\alpha) = \Pi_T(\sigma)$ we have $\alpha =_{\Pi} \sigma$, and we are done. \square

The proof of the second inclusion $L(G) \subseteq_{\Pi} L(M_G^{nm+1})$ is more involved. To explain its structure we need a definition.

Definition 2.1. A derivation $S = \alpha_0 \Rightarrow \dots \Rightarrow \alpha_{\ell}$ of G has index k if for every $i \in \{0, \dots, \ell\}$, the word $(\alpha_i)_{/V}$ has length at most k . The set of words derivable through derivations of index k is denoted by $L_k(G)$.

For example, the derivation $A_1 \Rightarrow A_1 A_2 \Rightarrow A_1 c A_1 \Rightarrow A_1 c a \Rightarrow a c a$ has index two. Clearly, we have $L_1(G) \subseteq L_2(G) \subseteq L_3(G) \dots$ and $L(G) = \bigcup_{k \geq 1} L_k(G)$.

The proof of $L(G) \subseteq_{\Pi} L(M_G^{nm+1})$ is divided into two parts. We first prove the *Collapse Lemma*, Lemma 2.3, stating that $L(G) \subseteq_{\Pi} L_{nm+1}(G)$, and then we prove, in Lemma 2.4, that $L_k(G) \subseteq_{\Pi} L(M_G^k)$ holds for every $k \geq 1$. A similar result has been proved in [7] with different notation and in a different context. We reformulate its proof here for the reader interested in a self-contained proof.

The Collapse Lemma. We need a few preliminaries. We assume the reader is familiar with the fact that every derivation can be parsed into a *parse tree* [3, Chapter 5], whose *yield* is the word produced by the derivation. We denote the yield of a parse tree t by $Y(t) \in T^*$, and the set of yields of a set \mathcal{T} of trees by $Y(\mathcal{T}) \subseteq \Sigma^*$. Figure 2 shows the parse tree of the derivation $A_1 \Rightarrow A_1A_2 \Rightarrow aA_2 \Rightarrow acA_1 \Rightarrow aca$. We introduce the notion of dimension of a parse tree.

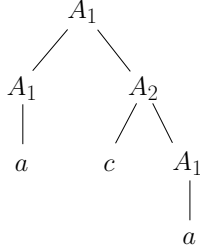


Figure 2: A parse tree of $A_1 \rightarrow A_1A_2|a$, $A_2 \rightarrow bA_2aA_2|cA_1$ with $S = A_1$

Definition 2.2. Let t be a parse tree. A *child* of t is a subtree of t whose root is a child of the root of t . A child of t is called *proper* if its root is not a leaf, i.e., if it is labeled with a variable. The *dimension* $d(t)$ of a parse tree t is inductively defined as follows. If t has no proper children, then $d(t) = 0$. Otherwise, let t_1, t_2, \dots, t_r be the proper children of t sorted such that $d(t_1) \geq d(t_2) \geq \dots \geq d(t_r)$. Then

$$d(t) = \begin{cases} d(t_1) & \text{if } r = 1 \text{ or } d(t_1) > d(t_2) \\ d(t_1) + 1 & \text{if } d(t_1) = d(t_2). \end{cases}$$

The set of parse trees of G of dimension k is denoted by $\mathcal{T}^{(k)}$, and the set of all parse trees of G by \mathcal{T} .

The parse tree of Fig. 2 has two children, both of them proper. It has dimension 1 and height 3. Observe also the following fact, which can be easily proved by induction.

Fact 2.1. Denote by $h(t)$ the height of a tree t . Then $h(t) > d(t)$.

For the proof of the collapse lemma, $L(G) \subseteq_{\Pi} L_{nm+1}(G)$, observe first that, since every word in $L(G)$ is the yield of some parse tree, we have $L(G) = Y(\mathcal{T})$, and so it suffices to show $Y(\mathcal{T}) \subseteq_{\Pi} L_{nm+1}(G)$. The proof is divided into two parts. We first show $Y(\mathcal{T}) \subseteq_{\Pi} \bigcup_{i=0}^n Y(\mathcal{T}^{(i)})$ in Lemma 2.1, and then we show $\bigcup_{i=0}^n Y(\mathcal{T}^{(i)}) \subseteq L_{nm+1}(G)$ in Lemma 2.2. Actually, the latter proves the stronger result that parse trees of dimension $k \geq 0$ have derivations of index $km + 1$, i.e., $Y(\mathcal{T}^{(k)}) \subseteq L_{km+1}(G)$ for all $k \leq 0$.

Lemma 2.1. $Y(\mathcal{T}) \subseteq_{\Pi} \bigcup_{i=0}^n Y(\mathcal{T}^{(i)})$.

PROOF. In this proof we write $t = t_1 \cdot t_2$ to denote that t_1 is a parse tree except that exactly one leaf ℓ is labelled by a variable, say A , instead of a terminal; the tree t_2 is a parse tree with root A ; and the tree t is obtained from t_1 and t_2 by replacing the leaf ℓ of t_1 by the tree t_2 . Figure 3 shows an example. In the

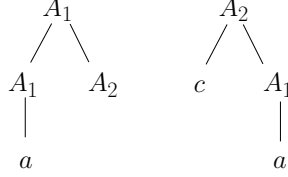


Figure 3: A decomposition t_1, t_2 such that $t = t_1 \cdot t_2$ is the parse tree of Fig. 2

rest of the proof we abbreviate *parse tree* to *tree*.

We need to prove that for every tree t there exists a tree t' such that $Y(t) =_{\Pi} Y(t')$ and $d(t') \leq n$. Say that two trees t, t' are *equivalent* if they have the same number of nodes, the sets of variables appearing in t and t' as well as their root variable coincide, and $Y(t) =_{\Pi} Y(t')$ holds. Say further that a tree t is *compact* if $d(t) \leq K(t)$, where $K(t)$ denotes the number of distinct variables that appear in t . Since $K(t) \leq n$ for every t , it suffices to show that every tree is equivalent to a compact tree.

The proof is by induction on the number of non-leaf nodes in t . In the base case, t has no proper child, and we have $d(t) = 0 \leq K(t)$. So t is compact and we are done.

Consider now the case in which t has $r \geq 1$ proper children t_1, \dots, t_r . Choose x with $1 \leq x \leq r$ such that $K(t_x) = \max_i K(t_i)$. We construct a compact tree that is equivalent to t . To this end we use an iterative procedure whose description begins in the following line.

By the induction hypothesis, each child t_i is equivalent to a compact tree. Replace in t each child by its equivalent compact counterpart. Clearly, the new t and the old t are equivalent. Also note that x still satisfies $K(t_x) = \max_i K(t_i)$. If t is now compact, the procedure terminates successfully. Otherwise, let $1 \leq y \leq r$ such that $d(t_y) = \max_i d(t_i)$. We have

$$\begin{aligned}
 d(t) &\leq d(t_y) + 1 && \text{(by definition of dimension and of } y) \\
 &\leq K(t_y) + 1 && \text{(as } t_y \text{ is compact)} \\
 &\leq K(t_x) + 1 && \text{(as } K(t_x) = \max_i K(t_i)) \\
 &\leq K(t) + 1 && \text{(as } t_x \text{ is a child of } t) \\
 &\leq d(t) && \text{(as } t \text{ is not compact),}
 \end{aligned} \tag{1}$$

so all inequalities in (1) are in fact equalities. In particular, it follows $d(t) = d(t_y) + 1$. The definitions of dimension and of y then imply that there exists $y' \neq y$ with $d(t_{y'}) = d(t_y)$. Hence $x \neq y$ or $x \neq y'$, and w.l.o.g. we can choose y such that $x \neq y$. Further, it follows $K(t_y) \stackrel{(1)}{=} d(t_y) < h(t_y)$ by Fact 2.1, thus

we find a path in t_y from the root to a leaf which passes through at least two nodes with the same variable, say A . So t_y can be factored into $t_y^a \cdot (t_y^b \cdot t_y^c)$ such that the roots of t_y^b and t_y^c are labelled by A . As $K(t) \stackrel{(1)}{=} K(t_x)$, we also find an A -labelled node in t_x , which allows us to write $t_x = t_x^a \cdot t_x^b$ with the root of t_x^b labelled by A . Now we cut out the middle part t_y^b of t_y , and insert it between the two parts t_x^a and t_x^b of t_x , so that we get $t'_x = t_x^a \cdot (t_y^b \cdot t_x^b)$ and $t'_y = t_y^a \cdot t_y^c$. Replace in t the child t_x by t'_x and the child t_y by t'_y . Clearly, the new t and the old t are equivalent. Observe that the child t_x has gained some nodes and not lost any. All other children did not gain any nodes, so we still have $K(t_x) = \max_i K(t_i)$. Now we iterate the procedure.

The whole procedure must terminate eventually, as the child t_x grows in each iteration, but the total number of nodes stays constant. Since the procedure only terminates after having constructed an equivalent compact tree, it eventually constructs an equivalent compact tree. \square

Lemma 2.2. *For every $k \geq 0$: $Y(\mathcal{T}^{(k)}) \subseteq L_{km+1}(G)$.*

PROOF. In this proof we will use the following notation. If D is a derivation $\alpha_0 \Rightarrow \dots \Rightarrow \alpha_\ell$ and $w, w' \in (V \cup T)^*$, then we define wDw' to be the step sequence $w\alpha_0w' \Rightarrow \dots \Rightarrow w\alpha_\ell w'$.

Let t be a parse tree such that $d(t) = k$. We show that there is a derivation for $Y(t)$ of index $km + 1$. We proceed by induction on the number of non-leaf nodes in t . In the base case, t has no proper child. Then we have $k = 0$ and t represents a derivation $S \Rightarrow Y(t)$ of index 1. For the induction step, assume that t has $r \geq 1$ proper children t_1, \dots, t_r where the root of t_i is assumed to be labeled by $A^{(i)}$; i.e., we assume that the topmost level of t is induced by a rule $S \rightarrow \gamma_0 A^{(1)} \gamma_1 \dots \gamma_{r-1} A^{(r)} \gamma_r$ for $\gamma_i \in T^*$. Note that $r - 1 \leq m$. By definition of dimension, at most one child t_i has dimension k , while the other children have dimension at most $k - 1$. W.l.o.g. assume $d(t_1) \leq k$ and $d(t_2), \dots, d(t_r) \leq k - 1$. By induction hypothesis, for all $1 \leq i \leq r$ there is a derivation D_i for $Y(t_i)$ such that D_1 has index $km + 1$, and D_2, \dots, D_r have index $(k - 1)m + 1$. Define, for each $1 \leq i \leq r$, the step sequence

$$D'_i := \gamma_0 A^{(1)} \gamma_1 \dots \gamma_{i-2} A^{(i-1)} \gamma_{i-1} D_i \gamma_i Y(t_{i+1}) \gamma_{i+1} \dots \gamma_{r-1} Y(t_r) \gamma_r.$$

If the notion of index is extended to step sequences in the obvious way, then D'_1 has index $km + 1$, and for $2 \leq i \leq r$, the step sequence D'_i has index $(i - 1) + (k - 1)m + 1 \leq km + 1$. By concatenating the step sequences $S \Rightarrow \gamma_0 A^{(1)} \gamma_1 \dots \gamma_{r-1} A^{(r)} \gamma_r$ and D_r, D_{r-1}, \dots, D_1 in that order, we obtain a derivation for $Y(t)$ of index $km + 1$. \square

Putting Lemma 2.2 and Lemma 2.1 together we obtain:

Lemma 2.3. [Collapse Lemma] $L(G) \subseteq_{\Pi} L_{nm+1}(G)$.

PROOF.

$$\begin{aligned} L(G) &= Y(\mathcal{T}) \\ &\subseteq_{\Pi} \bigcup_{i=0}^n Y(\mathcal{T}^{(i)}) \quad (\text{Lemma 2.1}) \\ &\subseteq L_{nm+1}(G) \quad (\text{Lemma 2.2}) \end{aligned}$$

□

Lemma 2.4. *For every $k \geq 1$: $L_k(G) \subseteq_{\Pi} L(M_G^k)$.*

PROOF. We show that if $S \Rightarrow^* \alpha$ is a prefix of a derivation of index k then M_G^k has a run $q_0 \xrightarrow{w} \Pi_V(\alpha)$ such that $w \in T^*$ and $\alpha_{/T} =_{\Pi} w$. The proof is by induction on the length i of the prefix.

$i = 0$. In this case $\alpha = S$, and since $q_0 = \Pi_V(S)$ and $S_{/T} = \varepsilon$ we are done.

$i > 0$. Since $S \Rightarrow^i \alpha$ there exist $\beta_1 A \beta_2 \in (V \cup T)^*$ and a production $A \rightarrow \gamma$ such that $S \Rightarrow^{i-1} \beta_1 A \beta_2 \Rightarrow \alpha$ and $\beta_1 \gamma \beta_2 = \alpha$. By induction hypothesis, there exists a run of M_G^k such that $q_0 \xrightarrow{w_1} \Pi_V(\beta_1 A \beta_2)$ and $(\beta_1 A \beta_2)_{/T} =_{\Pi} w_1$. Then the definition of M_G^k and the fact that $S \Rightarrow^i \alpha$ is of index k show that there exists a transition $(\Pi_V(\beta_1 A \beta_2), \gamma_{/T}, \Pi_V(\alpha))$, hence we find that $q_0 \xrightarrow{w_1 \cdot \gamma_{/T}} \Pi_V(\alpha)$. Next we conclude from $(\beta_1 A \beta_2)_{/T} =_{\Pi} w_1$ and $\alpha = \beta_1 \gamma \beta_2$ that $\alpha_{/T} =_{\Pi} w_1 \cdot \gamma_{/T}$ and we are done.

Finally, if $\alpha \in T^*$ so that $S \Rightarrow^* \alpha$ is a derivation, then $q_0 \xrightarrow{w} \Pi_V(\alpha) = (0, \dots, 0)$ where $(0, \dots, 0)$ is an accepting state and $\alpha = \alpha_{/T} =_{\Pi} w$. □

We now have all we need to prove the other inclusion.

Proposition 2.2. $L(G) \subseteq_{\Pi} L(M_G^{nm+1})$.

PROOF.

$$\begin{aligned} L(G) &\subseteq_{\Pi} L_{nm+1}(G) && \text{(Collapse Lemma)} \\ &\subseteq_{\Pi} L(M_G^{nm+1}) && \text{(Lemma 2.4)} \end{aligned}$$

□

3. An Application: Bounding the Size of Semilinear Sets

Recall that a set $S \subseteq \mathbb{N}^k$, $k \geq 1$, is *linear* if there is an *offset* $\mathbf{b} \in \mathbb{N}^k$ and *periods* $\mathbf{p}_1, \dots, \mathbf{p}_j \in \mathbb{N}^k$ such that $S = \{\mathbf{b} + \sum_{i=1}^j \lambda_i \mathbf{p}_i \mid \lambda_1, \dots, \lambda_j \in \mathbb{N}\}$. A set is *semilinear* if it is the union of a finite number of linear sets. It is easily seen that the Parikh image of a regular language is semilinear. Procedures for computing the semilinear representation of the language starting from a regular expression or an automaton are well-known (see e.g. [14]). Combined with Theorem 1.1 they provide an algorithm for computing the Parikh image of a context-free language.

Recently, To has obtained an upper bound on the size of the semilinear representation of the Parikh image of a regular language (see Theorem 7.3.1 of [16]):

Theorem 3.1. *Let A be an NFA with s states over an alphabet of ℓ letters. Then $\Pi(L(A))$ is a union of $\mathcal{O}(s^{\ell^2+3\ell+3} \ell^{4\ell+6})$ linear sets with at most ℓ periods; the maximum entry of any offset is $\mathcal{O}(s^{3\ell+3} \ell^{4\ell+6})$, and the maximum entry of any period is at most s .*

Plugging Theorem 1.1 into Theorem 3.1, we get the (to our knowledge) best existing upper bound on the size of the semilinear set representation of the Parikh image of a context-free language. Let $G = (V, T, P, S)$ be a context-free grammar of degree m with $n = |V|$ and $t = |T|$. Let p be the total number of occurrences of terminals in the productions of G , i.e., $p = \sum_{X \rightarrow \alpha \in P} |\alpha|_T$. The number of states of M_G^{nm+1} is $\binom{n+nm+1}{n}$. Recall that the transitions of M_G^{nm+1} are labelled with words of the form $\gamma|_T$, where γ is the right-hand-side of some production. Splitting transitions, adding intermediate states, and then removing ϵ -transitions yields an NFA with $\binom{n+nm+1}{n} \cdot p$ states. So we finally obtain for the parameters s and ℓ in Theorem 3.1 the values $s := \binom{n+nm+1}{n} \cdot p$, and $\ell := t$. This result (in fact a slightly stronger one) has been used in [6] to provide a polynomial algorithm for a language-theoretic problem relevant for the automatic verification of concurrent programs.

4. Conclusions and Related Work

For the sake of comparison we will assume throughout this section that all grammars have degree $m \leq 1$. Given G a context-free grammar with n variables, we have shown how to construct an NFA M with $\mathcal{O}(4^n)$ states such that $L(G)$ and $L(M)$ have the same Parikh image. We compare this result with previous proofs of Parikh's theorem.

Parikh's proof [1] (essentially the same proof is given in [15]) shows how to obtain a Parikh-equivalent regular expression from a finite set of parse trees of G . The complexity of the resulting construction is not studied. By its definition, the regular expression basically consists of the sum of words obtained from the parse trees of height at most n^2 . This leads to the admittedly rough bound that the regular expression consists of at most $\mathcal{O}(2^{2^{n^2-1}})$ words each of length at most $\mathcal{O}(2^{n^2})$.

Greibach [8] shows that a particular substitution operator on language classes preserves semilinearity of the languages. This result implies Parikh's theorem, if the substitution operator is applied to the class of regular languages. It is hard to extract a construction from this proof, as it relies on previously proved closure properties of language classes.

Pilling's proof [14] (also given in [4]) of Parikh's theorem uses algebraic properties of commutative regular languages. From a constructive point of view, his proof leads to a procedure that iteratively replaces a variable of the grammar G by a regular expression over the terminals and the other variables. This procedure finally generates a regular expression which is Parikh-equivalent to $L(G)$. Van Leeuwen [13] extends Parikh's theorem to other language classes, but, while using very different concepts and terminology, his proof leads to the same construction as Pilling's. Neither [14] nor [13] study the size of the resulting regular expression.

Goldstine [9] simplifies Parikh's original proof. An explicit construction can be derived from the proof, but it is involved: for instance, it requires to

compute for each subset of variables, the computation of all derivations with these variables up to a certain size depending on a pumping constant.

Hopkins and Kozen [10] generalize Parikh's theorem to commutative Kleene algebra. Like in Pilling [14] their procedure to compute a Parikh-equivalent regular expression is iterative; but rather than eliminating one variable in each step, they treat all variables in a symmetric way. Their construction can be adapted to compute a Parikh-equivalent finite automaton. Hopkins and Kozen show (by algebraic means) that their iterative procedure terminates after $\mathcal{O}(3^n)$ iterations for a grammar with n variables. In [7] we reduce this bound (by combinatorial means) to n iterations. The construction yields an automaton, but it is much harder to explain than ours. The automaton has size $\mathcal{O}(n^n)$.

In [2] Parikh's theorem is derived from a small set of purely equational axioms involving fixed points. It is hard to derive a construction from this proof.

In [5] Parikh's theorem is proved by showing that the Parikh image of a context-free language is the union of the sets of solutions of a finite number of systems of linear equations. In [17] the theorem is also implicitly proved, this time by showing that the Parikh image is the set of models of an existential formula of Presburger arithmetic. While the constructions yielding the systems of equations and the Presburger formulas are very useful, they are also more complicated than our construction of the Parikh automaton. Also, neither [5] nor [17] give bounds on the size of the semilinear set.

Acknowledgments

We thank two anonymous referees and Anca Muscholl for very useful suggestions.

- [1] R. J. Parikh, On context-free languages, *Journal of the ACM* 13 (4) (1966) 570–581.
- [2] L. Aceto, Z. Ésik, A. Ingólfssdóttir, A fully equational proof of Parikh's Theorem, *ITA* 36 (2) (2002) 129–153.
- [3] J. E. Hopcroft, R. Motwani, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd Edition, Addison-Wesley (2006).
- [4] J. H. Conway, *Regular algebra and finite machines*, Chapman and Hall, 1971.
- [5] J. Esparza, *Petri Nets, Commutative Context-Free Grammars, and Basic Parallel Processes*, *Fundamentæ Informaticæ* (1997).
- [6] J. Esparza, P. Ganty, Complexity of Pattern-Based Verification for Multi-threaded Programs, *POPL, Proceedings, ACM* (2011), 499–510.
- [7] J. Esparza, S. Kiefer, M. Luttenberger, Newtonian program analysis, *Journal of the ACM* 57 (6) (2010) 33:1–33:47.

- [8] S. A. Greibach, A generalization of Parikh's semilinear theorem, *Discrete Mathematics* 2 (4) (1972) 347–355.
- [9] J. Goldstine, A simplified proof of Parikh's Theorem, *Discrete Mathematics* 19 (3) (1977) 235–239.
- [10] M. W. Hopkins, D. C. Kozen, Parikh's Theorem in commutative Kleene algebra, *LICS* (1999), 394–401.
- [11] E. Kopczynski, A. W. To, Parikh Images of Grammars: Complexity and Applications, *LICS, Proceedings, IEEE Computer Society* (2010), 80–89.
- [12] M. Lange, H. Leiß, To CNF or not to CNF? An Efficient Yet Presentable Version of the CYK Algorithm, *Informatica Didactica* (8) (2008–2010).
- [13] J. van Leeuwen, A generalisation of Parikh's Theorem in formal language theory, *ICALP, LNCS* 14 (1974) 17–26.
- [14] D. L. Pilling, Commutative regular equations and Parikh's Theorem, *J. London Math. Soc.* 2 (6) (1973) 663–666.
- [15] A. Salomaa, *Formal Languages*, Academic Press, 1973.
- [16] A. W. To, *Model-Checking Infinite-State Systems: Generic and Specific Approaches*, PhD Thesis, University of Edinburgh (2010).
- [17] K. N. Verma, H. Seidl, T. Schwentick, On the Complexity of Equational Horn Clauses, *CADE, LNCS* 1831 (2005) 337–352.