# Converting Nondeterministic Automata and Context-Free Grammars into Parikh Equivalent Deterministic Automata

Giovanna J. Lavado[1], Giovanni Pighizzini[1], and Shinnosuke Seki[2]

[1] Dipartimento di Informatica, Università degli Studi di Milano
via Comelico 39, I-20135, Milano, Italy
{giovanna.lavado,giovanni.pighizzini}@unimi.it
[2] Department of Information and Computer Science, Aalto University,
P.O. Box 15400, FI-00076, Aalto, Finland
shinnosuke.seki@aalto.fi

**Abstract.** We investigate the conversion of nondeterministic finite automata and context-free grammars into Parikh equivalent deterministic finite automata, from a descriptional complexity point of view.

We prove that for each nondeterministic automaton with $n$ states there exists a Parikh equivalent deterministic automaton with $e^{O(\sqrt{n \cdot \ln n})}$ states. Furthermore, this cost is tight. In contrast, if all the strings accepted by the given automaton contain at least two different letters, then a Parikh equivalent deterministic automaton with a polynomial number of states can be found.

Concerning context-free grammars, we prove that for each grammar in Chomsky normal form with $n$ variables there exists a Parikh equivalent deterministic automaton with $2^{O(n^2)}$ states. Even this bound is tight.

**Keywords:** Finite automaton, context-free grammar, Parikh's theorem, descriptional complexity, semilinear set, Parikh equivalence.

## 1 Introduction

It is well-known that the state cost of the conversion of nondeterministic finite automata (NFAs) into equivalent deterministic finite automata (DFAs) is exponential: using the classical subset construction [13], from each $n$-state NFA we can build an equivalent DFA with $2^n$ states. Furthermore, this cost cannot be reduced.

In all examples witnessing such a state gap (e.g., [8–10]), input alphabets with at least two letters and proof arguments strongly relying on the structure of strings are used. As a matter of fact, for the unary case, namely the case of the one letter input alphabet, the cost reduces to $e^{\Theta(\sqrt{n \cdot \ln n})}$, as shown by Chrobak [1].

What happens if we do not care of the order of symbols in the strings, i.e., if we are interested only in obtaining a DFA accepting a set of strings which are equal, after permuting the symbols, to the strings accepted by the given NFA?

H.-C. Yen and O.H. Ibarra (Eds.): DLT 2012, LNCS 7410, pp. 284–295, 2012.

This question is related to the well-known notions of Parikh image and Parikh equivalence [11]. Two strings over a same alphabet $\Sigma$ are Parikh equivalent if and only if they are equal up to a permutation of their symbols or, equivalently, for each letter $a \in \Sigma$ the number of occurrences of $a$ in the two strings is the same. This notion extends in a natural way to languages (two languages $L_1$ and $L_2$ are Parikh equivalent when for each string in $L_1$ there is a Parikh equivalent string in $L_2$ and vice versa) and to formal systems which are used to specify languages as, for instance, grammars and automata. Notice that in the unary case Parikh equivalence is just the standard equivalence. So, in the unary case, the answer to our previous question is given by the above mentioned result by Chrobak.

Our first contribution in this paper is an answer to that question in the general case. In particular, we prove that the state cost of the conversion of $n$-state NFAs into Parikh equivalent DFAs is the same as in the unary case, i.e., it is $e^{\Theta(\sqrt{n \cdot \ln n})}$. More surprisingly, we prove that this is due to the unary parts of languages. In fact, we show that if the given NFA accepts only nonunary strings, i.e., each accepted string contains at least two different letters, then we can obtain a Parikh equivalent DFA with a polynomial number of states in $n$. Hence, while in standard determinization the most difficult part (with respect to the state complexity) is the nonunary one, in the "Parikh determinization" this part becomes easy and the most complex part is the unary one.

In the second part of the paper we consider context-free grammars (CFGs). Parikh Theorem [11] states that each context-free language is Parikh equivalent to a regular language. We study this equivalence from a descriptional complexity point of view. Recently, Esparza, Ganty, Kiefer, and Luttenberger proved that each CFG in Chomsky normal form with $h$ variables can be converted into a Parikh equivalent NFA with $O(4^h)$ states [2]. In [7] it was proven that if $G$ generates a bounded language then we can obtain a DFA with $2^{h^{O(1)}}$ states, i.e., a number exponential in a polynomial of the number of variables. In this paper, we are able to extend such a result by removing the restriction to bounded languages. We also reduce the upper bound to $2^{O(h^2)}$. A milestone for obtaining such a result is the conversion of NFAs to Parikh equivalent DFAs presented in the first part of the paper. By suitably combining that result (in particular the polynomial conversion in the case of NFAs accepting nonunary strings) with the above mentioned result from [2] and with a result by Pighizzini, Shallit, and Wang [12] concerning the unary case, we prove that each context-free grammar in Chomsky normal form with $h$ variables can be converted into a Parikh equivalent DFA with $2^{O(h^2)}$ states. From the results concerning the unary case, it follows that this bound is tight.

Even for this simulation, as for that of NFAs by Parikh equivalent DFAs, the main contribution to the state complexity of the resulting automaton is given by the unary part.

## 2 Preliminaries

Let $\Sigma = \{a_1, a_2, \ldots, a_m\}$ be an alphabet of $m$ letters. Let us denote by $\Sigma^*$ the set of all words over $\Sigma$ including the empty word $\varepsilon$. Given a word $w \in \Sigma^*$, $|w|$ denotes its length and, for a letter $a \in \Sigma$, $|w|_a$ denotes the number of occurrences of $a$ in $w$. For a word $u \in \Sigma^*$, $w$ is a prefix of $u$ if $u = wx$ for some word $x \in \Sigma^*$. We denote by $\mathrm{Pref}(u)$ the set of all prefixes of $u$, and for a language $L \subseteq \Sigma^*$, let $\mathrm{Pref}(L) = \bigcup_{u \in L} \mathrm{Pref}(u)$.

A language $L$ has the *prefix property* or, equivalently, is said to be *prefix-free* if and only if for each string $x \in L$, each proper prefix of $x$ does not belong to $L$. Given two sets $A, B$ and a function $f : A \to \Sigma^*$, we say that $f$ has the prefix property on $B$ if and only if the language $f(A \cap B)$ has the prefix property.

We denote the set of integers by $\mathbb{Z}$ and the set of nonnegative integers by $\mathbb{N}$. Then $\mathbb{Z}^m$ and $\mathbb{N}^m$ denote the corresponding sets of $m$-dimensional integer vectors including the *null vector* $\mathbf{0} = (0, 0, \ldots, 0)$. For $1 \leq i \leq m$, we denote the $i$-th component of a vector $\boldsymbol{v}$ by $\boldsymbol{v}[i]$.

Given $k$ vectors $\boldsymbol{v_1}, \ldots, \boldsymbol{v_k} \in \mathbb{Z}^m$, we say that they are *linearly independent* if and only if for all $n_1, \ldots n_k \in \mathbb{Z}$, $n_1\boldsymbol{v_1} + \cdots + n_k\boldsymbol{v_k} = \mathbf{0}$ implies $n_1 = \ldots = n_k = 0$. It is well-know that, in this case, $k$ cannot exceed $m$. The following result will be used in the paper.

**Lemma 1.** *Given $k$ linearly independent vectors $\boldsymbol{v_1}, \ldots, \boldsymbol{v_k} \in \mathbb{Z}^m$ there are $k$ pairwise different integers $t_1, \ldots, t_k \in \{1, \ldots, m\}$ such that $\boldsymbol{v_j}[t_j] \neq 0$, for $j = 1, \ldots, k$.*

*Proof.* Let $W$ be the $m \times k$ matrix which has $\boldsymbol{v_1}, \ldots, \boldsymbol{v_k}$ as columns. Since the given vectors are linearly independent, $k \leq m$. Furthermore, by suitably deleting $m - k$ rows from $W$, we can obtain a $k \times k$ matrix $V$ whose determinant $d(V)$ is nonnull.

If $k = 1$ then the result is trivial. Otherwise, we can compute $d(V)$ along the last column as $d(V) = \sum_{i=1}^{k} (-1)^{i+k} \boldsymbol{v_k}[i] d_{i,k}$, where $d_{i,k}$ is the determinant of the matrix $V_{i,k}$ obtained by removing from $V$ the row $i$ and the column $k$. Since $d(V) \neq 0$, there is at least one index $i$ such that $\boldsymbol{v_k}[i]$ and $d_{i,k} \neq 0$. Hence, as $t_k$ we take such $i$. Using an induction on the matrix $V_{t_k,k}$, we can finally obtain the sequence $t_1, \ldots, t_k$ satisfying the statement of the theorem. □

A vector $\boldsymbol{v} \in \mathbb{Z}^m$ is *unary* if it contains at most one nonzero component, i.e., $\boldsymbol{v}[i], \boldsymbol{v}[j] \neq 0$ for some $1 \leq i, j \leq m$ implies $i = j$; otherwise, it is *nonunary*. By definition, the null vector is unary.

In the sequel, we reserve $\preceq$ for the componentwise partial order on $\mathbb{N}^m$, i.e., $\boldsymbol{u} \preceq \boldsymbol{v}$ if and only if $\boldsymbol{u}[k] \leq \boldsymbol{v}[k]$ for all $1 \leq k \leq m$. For a vector $\boldsymbol{v} \in \mathbb{N}^m$, let $\mathrm{Pred}(\boldsymbol{v}) = \{\boldsymbol{u} \mid \boldsymbol{u} \preceq \boldsymbol{v}\}$. For $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^m$, $\boldsymbol{v} - \boldsymbol{u}$ is defined to be a vector $\boldsymbol{w}$ with $\boldsymbol{w}[k] = \boldsymbol{v}[k] - \boldsymbol{u}[k]$ for all $1 \leq k \leq m$. Note that $\boldsymbol{v} - \boldsymbol{u}$ is a vector in $\mathbb{N}^m$ if and only if $\boldsymbol{u} \preceq \boldsymbol{v}$.

A *semilinear set* in $\mathbb{N}^m$ is a finite union of *linear sets* of the form $\{\boldsymbol{v_0} + \sum_{i=1}^{k} n_i \boldsymbol{v_i} \mid n_1, \ldots, n_k \in \mathbb{N}\}$, where $k \geq 0$ and $\boldsymbol{v_0}, \boldsymbol{v_1}, \ldots, \boldsymbol{v_k} \in \mathbb{N}^m$. The vector $\boldsymbol{v_0}$ is called *offset*, while the vectors $\boldsymbol{v_1}, \ldots, \boldsymbol{v_k}$ are called *generators*.

The *Parikh map* $\psi : \Sigma^* \to \mathbb{N}^m$ associates with a word $w \in \Sigma^*$ the vector $(|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_m})$. Then a word $w \in \Sigma^*$ is *unary* if and only if its Parikh image $\psi(w)$ is a unary vector; otherwise, $w$ is *nonunary*. One can naturally generalize this map for a language $L \subseteq \Sigma^*$ as $\psi(L) = \{\psi(w) \mid w \in L\}$. $\psi(L)$ is called the *Parikh image* of $L$. Languages $L, L' \subseteq \Sigma^*$ are said to be *Parikh equivalent* to each other if $\psi(L) = \psi(L')$.

We assume the readers to be familiar with the notions of deterministic and nondeterministic finite automata (abbreviated as DFA and NFA), context-free grammar (CFG), and context-free language (CFL) as well as their basic properties (see [5] for them). A CFG $G$ is denoted by a quadruple $(V, \Sigma, P, S)$, where $V$ is the set of variables, $P$ the set of productions, and $S \in V$ the start variable. By $L(G)$, we denote the set of all words in $\Sigma^*$ that have at least one derivation from $S$. $G$ is said to be in *Chomsky normal form* if all of its productions are in one of the three simple forms, either $B \to CD$, $B \to a$, or $S \to \varepsilon$, where $a \in \Sigma$, $B \in V$, and $C, D \in V \setminus \{S\}$. CFGs in Chomsky normal form are called *Chomsky normal form grammars* (CNFGs). According to the discussion in [4], we employ the number of variables of CNFGs as a "reasonable" measure of descriptional complexity for CFLs.

Parikh equivalence can be defined not only between languages but among languages, grammars, and finite automata. A CFG $G$ is *Parikh equivalent* to a language $L$ if $\psi(L(G)) = \psi(L)$. Likewise, for a finite automaton $A$, we can say that $A$ is *Parikh equivalent* to $L$ if $\psi(L(A)) = \psi(L)$, where $L(A)$ is the set of words accepted by $A$.

Parikh's theorem [11] states that the Parikh image of any context-free language is a semilinear set. Thus, it has the following immediate consequence.

**Theorem 1 ([11]).** *Every context-free language is Parikh equivalent to a regular language.*

It is immediate to observe that in the case of *unary languages*, i.e., languages defined over a one letter alphabet, two languages are Parikh equivalent if and only if they are equal. Hence, as a consequence of Theorem 1, each unary context-free language is regular. This result was firstly proved, without using Parikh's Theorem, by Ginsburg and Rice [3]. The equivalence between unary context-free and regular languages has been studied from the descriptional complexity point of view in [12], where the following result was proved:

**Theorem 2 ([12]).** *For any* CNFG $G$ *with* $h$ *variables that generates a unary language, there exists an equivalent* DFA $M$ *with less than* $2^{h^2}$ *states.*

In the paper we will also make use of the transformation of unary NFAs into DFAs.

**Theorem 3 ([1]).** *The state cost of the conversion of* $n$*-state unary* NFA*s into equivalent* DFA*s is* $e^{\Theta(\sqrt{n \cdot \ln n})}$.

## 3 From NFAs to Parikh Equivalent DFAs

In this section we present our first main contribution. From each $n$-state NFA $A$ we derive a Parikh equivalent DFA $A'$ with $e^{O(\sqrt{n \cdot \ln n})}$ states. Furthermore, we prove that this cost is tight.

Actually, as a preliminary step we obtain a result which is interesting *per se*: if each string accepted by the given NFA $A$ contains at least two different symbols, i.e., it is nonunary, then the Parikh equivalent DFA $A'$ can be obtained with polynomially many states. Hence, the superpolynomial blowup is due to the unary part of the accepted language.

A fundamental tool which will be used in this section is the following normal form for the Parikh image of NFAs, which is based on a result from [6, 14].

**Lemma 2.** *Given an alphabet $\Sigma = \{a_1, a_2, \ldots, a_m\}$, there exists a polynomial $p$ such that for each $n$-state* NFA *$A$ over $\Sigma$, if all the words in $L(A)$ are nonunary then $\psi(L(A)) = Y \cup \bigcup_{i \in I} Z_i$ where:*

- *$Y \subseteq \mathbb{N}^m$ is a finite set of vectors whose components are bounded by $p(n)$;*
- *$I$ is a set of at most $p(n)$ indices;*
- *for each $i \in I$, $Z_i \subseteq \mathbb{N}^m$ is a linear set of the form:*

$$Z_i = \{\boldsymbol{v_{i,0}} + n_1 \boldsymbol{v_{i,1}} + \cdots + n_k \boldsymbol{v_{i,k}} \mid n_1, \ldots, n_k \in \mathbb{N}\} \tag{1}$$

*where $0 \leq k \leq m$, the components of $\boldsymbol{v_{i,0}}$ are bounded by $p(n)$, $\boldsymbol{v_{i,1}}, \ldots, \boldsymbol{v_{i,k}}$ are linearly independent vectors from $\{0, 1, \ldots, n\}^m$.*

*Futhermore, for each $i \in I$ we can choose a nonunary vector $\boldsymbol{x_i} \in \mathrm{Pred}(\boldsymbol{v_{i,0}})$ such that all those vectors are pairwise distinct.*

*Proof.* In [6, 14] it was proved that $\psi(L(A))$ can be written as claimed in the first part of the statement of the lemma, with $Y = \emptyset$, $I$ of size $O(n^{m^2+3m+5} m^{4m+6})$, and the components of each offset $\boldsymbol{v_{i,0}}$ bounded by $O(n^{3m+5} m^{4m+6})$ (the result holds also if unary words are accepted). We notice that, since the language accepted by $A$ does not contain any unary word, all the offsets $\boldsymbol{v_{i,0}}$ are nonunary.

If for each $i \in I$ we can choose $\boldsymbol{x_i} \in \mathrm{Pred}(\boldsymbol{v_{i,0}})$ such that all $\boldsymbol{x_i}$'s are pairwise different, then the proof is completed.

Otherwise, we proceed as follows. For a vector $\boldsymbol{v}$, let us denote by $\|\boldsymbol{v}\|$ its *infinite norm*, i.e., the value of its maximum component. Let us suppose $I \subseteq \mathbb{N}$ and we denote as $N_I$ the maximum element of $I$.

By proceeding in increasing order, for $i \in I$ we choose a nonunary vector $\boldsymbol{x_i} \in \mathrm{Pred}(\boldsymbol{v_{i,0}})$ such that $\|\boldsymbol{x_i}\| \leq i$ and $\boldsymbol{x_i}$ is different from all already chosen $\boldsymbol{x_j}$, i.e., $\boldsymbol{x_i} \neq \boldsymbol{x_j}$ for all $j \in I$ with $j < i$. The extra condition $\|\boldsymbol{x_i}\| \leq i$ will turn out to be useful later.

When for an $i \in I$ it is not possible to find such $\boldsymbol{x_i}$, we replace $Z_i$ by some suitable sets. Essentially, those sets are obtained by enlarging the offsets using sufficiently long "unrollings" of the periods. In particular, for $j = 1, \ldots, k$, we consider the set

$$Z_{N_I+j} = \{(\boldsymbol{v_{i,0}} + h_j \boldsymbol{v_{i,j}}) + n_1 \boldsymbol{v_{i,1}} + \cdots + n_k \boldsymbol{v_{i,k}} \mid n_1, \ldots, n_k \in \mathbb{N}\} \tag{2}$$

where $h_j$ is an integer satisfying the inequalities

$$N_I + j \leq \|\boldsymbol{v_{i,0}} + h_j \boldsymbol{v_{i,j}}\| < N_I + j + n \tag{3}$$

Due to the fact that $\boldsymbol{v_{i,j}} \in \{0, \ldots, n\}^m$, we can always find such $h_j$. Furthermore, we consider the following set

$$Y_i = \{\boldsymbol{v_{i,0}} + n_1 \boldsymbol{v_{i,1}} + \cdots + n_k \boldsymbol{v_{i,k}} \mid 0 \leq n_1 < h_1, \ldots, 0 \leq n_k < h_k\} \tag{4}$$

It can be easily verified that $Z_i = Y_i \cup \bigcup_{j=1}^{k} Z_{N_I+j}$.

Now we replace the set of indices $I$ by the set $\widehat{I} = I - \{i\} \cup \{N_I+1, \ldots, N_I+k\}$ and the set $Y$ by $\widehat{Y} = Y \cup Y_i$. We continue the same process by considering the next index $i$.

We notice that, since we are choosing each vector $\boldsymbol{x_i} \in \mathrm{Pred}(\boldsymbol{v_{i,0}})$ in such a way that $\|\boldsymbol{x_i}\| \leq i$, when we will have to choose the vector $\boldsymbol{x_{N_I+j}}$ for a set $Z_{N_I+j}$ introduced at this stage, by the condition (3) we will have at least one possibility (a vector with one component equal to $N_I+j$ and another component equal to 1; we remind the reader that, since the given automaton accepts only nonunary strings, all offsets are nonunary). This implies that after examining all sets $Z_i$ corresponding to the original set $I$, we do not need to further modify the sets introduced during this process.

We finally observe that for each $Z_i$ in the initial representation, we introduced at most $m$ sets. Hence, the cardinality $\widetilde{N}$ of the set of indices resulting at the end of this process is $O(n^{m^2+3m+5} m^{4m+7})$.

By (3) the components of the offsets which have been added in this process cannot exceed $\widetilde{N} + n$. Hence, it turns out that $m \cdot (\widetilde{N} + n)$ is an upper bound to the components of vectors in $Y_i$.

This permit us to conclude that $p(n) = O(n^{m^2+3m+5} m^{4m+8})$ is upper bound for all these amounts. □

Now we are able to consider the case of automata accepting only words that are nonunary.

**Theorem 4.** *For each $n$-state* NFA *accepting a language none of whose words are unary, there exists a Parikh equivalent* DFA *with a number of states polynomial in $n$.*

*Proof.* Let $A$ be the given $n$-state NFA. We can express $\psi(L(A))$ as $Y \cup \bigcup_{i \in I} Z_i$ according to Lemma 2. Starting from this representation, we will build a DFA $A_{\mathrm{non}}$ Parikh equivalent to $A$. To this end, we first build for each $Z_i$ a DFA $A_i$ that accepts a language whose Parikh image is equal to $Z_i$, and then, from the automata $A_i$'s so obtained, we derive the DFA $A'$ Parikh equivalent to $\bigcup_{i \in I} Z_i$. We will also give a DFA $A''$ accepting a language Parikh equivalent to $Y$. Hence, by the standard construction for the union, we will finally get a DFA $A_{\mathrm{non}}$ Parikh equivalent to $A$.

First, we handle the generators of $Z_i$. Let us introduce a function $g : \mathbb{N}^m \to \Sigma^*$ as: for a vector $\boldsymbol{v} = (i_1, \ldots, i_m)$, $g(\boldsymbol{v}) = a_1^{i_1} a_2^{i_2} \cdots a_m^{i_m}$. Using this function, we map the generators $\boldsymbol{v_{i,1}}, \ldots, \boldsymbol{v_{i,k}}$ into the words $s_{i,1}, \ldots, s_{i,k}$; that is,

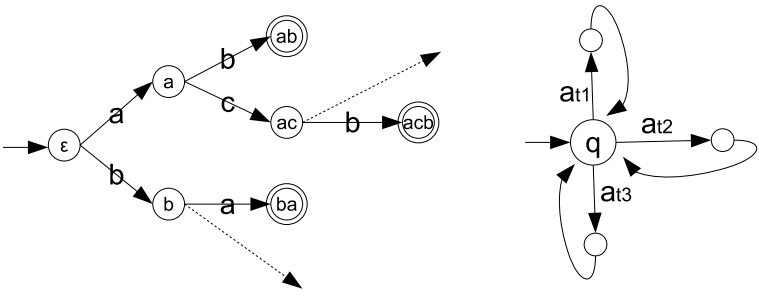


**Fig. 1.** (Left) A construction of DFA $A_W$, where the state $q_u$ is simply denoted by $u$ for clarity. (Right) A construction of DFA $A_i$ that accepts $\{w_{i,1}, w_{i,2}, \ldots, w_{i,k}\}^*$ for $k = 3$. In the construction of the final DFA $A_{\text{non}}$, if $w_{i,0} = acb$, then the initial state of $A_i$ is merged with the state $q_{acb}$ of $A_W$.

$s_{i,j} = g(\boldsymbol{v_{i,j}})$. It is easy to define a finite automaton accepting the language $\{s_{i,1}, s_{i,2}, \ldots, s_{i,k}\}^*$, which consists of a start state $q$ with $k$ loops labeled with $s_{i,1}, s_{i,2}, \ldots, s_{i,k}$, respectively. The state $q$ is the only accepting state. However, this automaton is nondeterministic. To avoid this problem, we modify the language by replacing each $s_{i,j}$, for $j = 1, \ldots, k$, with a Parikh equivalent word $w_{i,j}$ in such a way that for all pairwise different $j, j'$ the corresponding words $w_{i,j}$ and $w_{i,j'}$ begin with different letters. This is possible due to the fact, being $\boldsymbol{v_{i,1}}, \ldots, \boldsymbol{v_{i,k}}$ linearly independent, according to Lemma 1 we can find $k$ different letters $a_{t_1}, a_{t_2}, \ldots, a_{t_k} \in \Sigma$ such that $\boldsymbol{v_{i,j}}[t_j] > 0$ for $j = 1, \ldots, k$. For $j = 1, \ldots, k$, we "rotate" each $s_{i,j}$ by a cyclic shift so that the resulting word, $w_{i,j}$, begins with an occurrence of the letter $a_{t_j}$. Then $w_{i,j}$ is Parikh equivalent to $s_{i,j}$. For example, if $s_{i,j} = a_1^3 a_2^4 a_3$ and $t_j = 2$, then $w_{i,j}$ should be chosen as $a_2^4 a_3 a_1^3$. The construction of a DFA $A_i$ with one unique accepting state $q$ that accepts $\{w_{i,1}, w_{i,2}, \ldots, w_{i,k}\}^*$ must be now clear; $q$ with $k$ loops labeled with these respective $k$ words (see Fig. 1 (Right)). Furthermore, due to the limitations deriving from Lemma 2, the length of these loops is at most $mn$ so that this DFA contains at most $1 + m(mn - 1)$ states.

Next, we handle all the offsets $\boldsymbol{v_{i,0}}$ for $i \in I$. For that, we use the function $f : \mathbb{N}^m \to \Sigma^*$ defined as: for $\boldsymbol{v} \in \mathbb{N}^m$, $f(\boldsymbol{v}) = \hookleftarrow(g(\boldsymbol{v}))$, where $\hookleftarrow$ denotes the 1-step left circular shift. For example, $f(4, 1, 2, 0, \ldots, 0) = a_1^3 a_2 a_3^2 a_1$. It can be verified that the 1-step left circular shift endows $f$ with the prefix property *over the nonunary vectors*, that is, for any $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{N}^m$ that are nonunary, if $f(\boldsymbol{u})$ is a prefix of $f(\boldsymbol{v})$, then $\boldsymbol{u} = \boldsymbol{v}$. Let $w_{i,0} = f(\boldsymbol{x_i})g(\boldsymbol{v_{i,0}} - \boldsymbol{x_i})$, where $\boldsymbol{x_i}$ is given by Lemma 2. We now consider the finite language $W = \{w_{i,0} \mid i \in I\}$. Because both $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ are nonunary and $f$ has the prefix property over nonunary words, the language $W$ is prefix-free. We build a (partial) DFA that accepts $W$, which is denoted by $A_W = (Q_W, \Sigma, q_\varepsilon, \delta_W, F_W)$, where $Q_W = \{q_u \mid u \in \text{Pref}(W)\}$ and $F_W = \{q_u \mid u \in W\}$. Its transition function $\delta_W$ is defined as: for $u \in \text{Pref}(W)$ and $a \in \Sigma$, if $ua \in \text{Pref}(W)$, then $\delta(q_u, a) = q_{ua}$, while $\delta(q_u, a)$ is undefined

otherwise. See Fig. 1 (Left). Clearly, this accepts $W$. Since the longest word(s) in $W$ is of length $m \cdot p(n)$, this DFA contains at most $1 + |I| \cdot m \cdot p(n) \leq O(p^2(n))$.

It goes without saying that each accepting state of this DFA is only for one word in $W$. In other words, it accepts two distinct words in $W$ at distinct two states. Now, based on $A_W$ and the DFAs $A_i$ with $i \in I$, we can build a finite automaton that accepts the language $\bigcup_{i \in I} w_{i,0} L(A_i)$ *without introducing any new state*. This is simply done by merging $q_{w_{i,0}}$ with the start state of $A_i$. Given an input $u$, the resulting automaton $A'$ simulates the DFA $A_W$, looking for a prefix $w$ of $u$ such that $w \in W$. When such a prefix is found, $A'$ starts simulating $A_i$ on the remaining suffix $z$, where $i$ is the index such that $w = w_i$. Since $W$ is prefix-free, we need only to consider one decomposition of the input as $u = wz$. This implies that $A'$ is deterministic. Finally, we observe that $A'$ contains at most $O(p^2(n)) + |I|(1 + m(mn - 1)) = O(p^2(n))$ states, i.e., a number which is polynomial in $n$.

We now sketch the construction of a DFA $A''$ accepting a language $L_Y$ whose Parikh image is $Y$. We just take $L_Y = \{g(\boldsymbol{v}) \mid \boldsymbol{v} \in Y\}$. Let $M$ be the maximum of the components of vectors in $Y$. With each $\boldsymbol{v} \in \{0, \ldots, M\}^m$, we associate a state $q_{\boldsymbol{v}}$ which is reachable from the initial state by reading the string $g(\boldsymbol{v})$. Final states are those corresponding to vectors in $Y$. The automaton $A''$ so obtained has $M^m = p^m(n)$ states, a number polynomial in $n$.

Finally, by applying the standard construction for the union, from automata $A'$ and $A''$ we obtain the DFA $A_{\text{non}}$ Parikh equivalent to the given NFA $A$. □

We now switch to the general case. We prove that for each input alphabet the state cost of the conversion of NFAs into Parikh equivalent DFAs is the same as for the unary alphabet.

**Theorem 5.** *For each $n$-state* NFA*, there exists a Parikh equivalent* DFA *with $e^{O(\sqrt{n \cdot \ln n})}$ states. Furthermore, this cost is tight.*

*Proof.* From a given $n$-state NFA $A$ with input alphabet $\Sigma = \{a_1, a_2, \ldots, a_m\}$, for each $i = 1, \ldots, m$, we first build an $n$-state NFA $A_i$ accepting the unary language $L(A) \cap a_i^*$. Using Theorem 3, we convert $A_i$ into an equivalent DFA $A'_i$ with $e^{O(\sqrt{n \cdot \ln n})}$ states. We can assume that the state sets of the resulting automata are pairwise disjoint.

We define $A_u$ that accepts $\{w \in L(A) \mid w \text{ is unary}\}$ consisting of one copy of each of these DFAs and a new state $q_s$, which is its start state. In reading the first letter $a_i$ of an input, $A_u$ transits from $q_s$ to the state $q$ in the copy of $A'_i$ if $A'_i$ transits from its start state to $q$ on $a_i$ (such $q$ is unique because $A'_i$ is deterministic). These transitions from $q_s$ do not introduce any nondeterminism because $A'_1, \ldots, A'_m$ are defined over pairwise distinct letters. After thus entering the copy, $A_u$ merely simulates $A'_i$. The start state $q_s$ should be also an accepting state if and only if $\varepsilon \in L(A'_i)$ for some $1 \leq i \leq m$. Being thus built, $A_u$ accepts $\{w \in L(A) \mid w \text{ is unary}\}$ and contains at most $m \cdot e^{O(\sqrt{n \cdot \ln n})} + 1$ states.

On the other hand, the language $\{w \in L(A) \mid w \text{ is not unary}\}$ can be accepted by an $O(n)$-state NFA $A_0$, and Theorem 4 converts this NFA into a Parikh

equivalent DFA $A_n$ with a number of states $r(n)$, polynomial in $n$. The standard product construction is applied to $A_u$ and $A_n$ in order to build a DFA accepting $L(A_u) \cup L(A_n)$. The number of states of the DFA thus obtained is bounded by the product $e^{O(\sqrt{n \cdot \ln n})} \cdot r(n) = e^{O(\sqrt{n \cdot \ln n})} \cdot e^{O(\ln n)} = e^{O(\sqrt{n \cdot \ln n} + \ln n)}$, which is still bounded by $e^{O(\sqrt{n \cdot \ln n})}$.

Finally, we observe that by Theorem 3, in the unary case $e^{\Theta(\sqrt{n \cdot \ln n})}$ is the tight cost of the conversion from $n$-state NFAs to DFAs. This implies that the upper bound we obtained here cannot be reduced. □

## 4 From CFGs to Parikh Equivalent DFAs

In this section we extend the results of Section 3 to the conversion of CFGs in Chomsky normal form to Parikh equivalent DFAs. Actually, Theorem 4 will play an important role in order to obtain the main result of this section. The other important ingredient is the following result proved by Esparza *et al.* [2], which gives the cost of the conversion of CNFGs into Parikh equivalent NFAs.

**Theorem 6 ([2]).** *For a* CNFG *with $h$ variables, there exists a Parikh equivalent* NFA *with $O(4^h)$ states.*

By combining Theorem 6 with the main result of the previous section, i.e., Theorem 5, we can immediately obtain a double exponential upper bound in $h$ for the size of DFAs Parikh equivalent to CNFGs with $h$ variables. However, in the next theorem we show how to do better. In fact, we are able to reduce the upper bound to a single exponential in a polynomial of $h$.

**Theorem 7.** *For any* CNFG *with $h$ variables, there exists a Parikh equivalent* DFA *with at most $2^{O(h^2)}$ states.*

*Proof.* Let us denote the given CNFG by $G = (V, \Sigma, P, S)$, where $|V| = h$ and $\Sigma = \{a_1, a_2, \ldots, a_m\}$ for some $m \geq 1$.

In the case $m = 1$ (unary alphabet), one can employ Theorem 2 (note that, over a unary alphabet, two languages $L_1, L_2$ are Parikh equivalent if and only if they are equivalent). Hence, from now on we assume $m \geq 2$.

Let us give an outline of our construction first:

1. From $G$, we first create CNFGs $G_1, G_2, \ldots, G_m$ with at most $h$ variables each, and $G_{\text{non}}$ with $mh - m + 1$ variables such that, for $1 \leq i \leq m$, $G_i$ generates $L(G) \cap a_i^*$ and $G_{\text{non}}$ generates the rest, i.e., $L(G_{\text{non}}) = \{w \in L(G) \mid w \text{ is not unary}\}$.
2. The grammars $G_1, G_2, \ldots, G_m$ are converted into respectively equivalent unary DFAs $A_1, A_2, \ldots, A_m$. From these DFAs, a DFA $A_{\text{unary}}$ accepting the set of all unary words in $L(G)$ is constructed.
3. The grammar $G_{\text{non}}$ is converted into a Parikh equivalent NFA.
4. From the NFA so obtained, a Parikh equivalent DFA $A_{\text{non}}$ is built.
5. Finally, from $A_{\text{unary}}$ and $A_{\text{non}}$, a DFA that accepts the union of $L(A_{\text{unary}})$ and $L(A_{\text{non}})$ is obtained.

Observe that $L(A_{\text{unary}}) = \{w \in L(G) \mid w \text{ is unary}\}$ and $L(A_{\text{non}})$ is Parikh equivalent to $L(G_{\text{non}}) = \{w \in L(G) \mid w \text{ is not unary}\}$. Thus, the DFA which is constructed in this way is Parikh equivalent to the given grammar $G$.

We already have all the tools we need to implement Steps 2-5. (Step 1 will be discussed later.) In particular:

2. According to Theorem 2, for $i = 1, \ldots, m$, grammar $G_i$ is converted into a DFA $A_i$ with less than $2^{h^2}$ states. Using the same strategy presented in the proof of Theorem 5, from $A_1, \ldots, A_m$, we define $A_{\text{unary}}$ consisting of one copy of each of these DFAs and a new state $q_s$, which is its start state. Hence, the number of states of $A_{\text{unary}}$ does not exceed $m2^{h^2}$.
3. This step is done using Theorem 6. The number of the states of the resulting NFA is exponential in the number of the variables of the grammar $G_{\text{non}}$ and, hence, exponential in $h$.
4. This step is done using Theorem 4. The number of states of the resulting DFA $A_{\text{non}}$ is polynomial in the number of states of the NFA obtained in Step 3 and, hence, exponential in $h$.
5 The final DFA can be obtained as the product of two automata $A_{\text{unary}}$ and $A_{\text{non}}$. Considering the bounds obtained in Step 2 and 4 we conclude that the number of states in exponential in $h^2$.

To complete the proof, we have to discuss Step 1, where we design from $G$ with $h$ variables the CNFGs $G_1, G_2, \ldots, G_m$ and $G_{\text{non}}$ with only linear blowup in the number of their variables. The design of $G_i$ is simply done by deleting from $P$ all productions of the form $B \to a_j$ with $i \neq j$. Built in this manner, it is impossible for $G_i$ to contain more than $h$ variables.

Giving such a linear upper bound on the number of variables for $G_{\text{non}}$ is slightly more involved. It is clear that any production of one letter or $\varepsilon$ directly from $S$ is contrary to the purpose of $G_{\text{non}}$. This observation enables us to focus on the derivations by $G$ that begins with replacing $S$ by two variables. Consider a derivation $S \Rightarrow_G BC \Rightarrow_G^+ uC \Rightarrow_G^+ uv$ for some non-empty words $u, v \in \Sigma^+$ and $S \to BC \in P$. $G_{\text{non}}$ simulates $G$, but also requires extra feature to test whether $u$ and $v$ contain respectively letters $a_i$ and $a_j$ for some $i \neq j$ and make only derivations that pass this test valid. To this end, we let the start variable $S'$ of $G_{\text{non}}$ make guess which of the two distinct letters in $\Sigma$ have to derive from $B$ and $C$, respectively. We encode this guess into the variables in $V \setminus \{S\}$ as a subscript like $B_i$ (this means that, for $w \in \Sigma^*$, $B_i \Rightarrow_{G_{\text{non}}}^+ w$ if and only if $B \Rightarrow_G^+ w$ and $w$ contains at least one $a_i$).

Now, we give a formal definition of $G_{\text{non}}$ as a quadruple $(V', \Sigma, P', S')$, where $V' = \{S'\} \cup \{B_i \mid B \in V \setminus \{S\}, 1 \leq i \leq m\}$ and $P'$ consists of the following production rules:

1. $\{S' \to B_i C_j \mid S \to BC \in P \text{ and } 1 \leq i, j \leq m \text{ with } i \neq j\}$;
2. $\{B_i \to C_i D_j, B_i \to C_j D_i \mid B \to CD \in P \text{ with } B \neq S \text{ and } 1 \leq i, j \leq m\}$;
3. $\{B_i \to a_i \mid B \to a_i \in P \text{ and } 1 \leq i \leq m\}$.

We conclude the proof by checking that $L(G_{\text{non}}) = \{w \in L(G) \mid w \text{ is not unary}\}$. It must be obvious for trained readers, and hence, they can skip the check and directly go after the end of the proof.

**Lemma 3.** *Let $B_i$ be a variable of $G_{\mathrm{non}}$ that is different from the start variable. For $w \in \Sigma^*$, $B_i \Rightarrow^+_{G_{\mathrm{non}}} w$ if and only if $B \Rightarrow^+_G w$ and $w$ contains at least one occurrence of $a_i \in \Sigma$.*

*Proof.* Both inclusions will be proved using induction on the length of derivations.

($\Rightarrow$): If $B_i \Rightarrow_{G_{\mathrm{non}}} w$ (single-step derivation), then $w$ must be $a_i$ and $B \to a_i \in P$ according to the type-3 production in $P'$. Hence, the base case is correct. The longer derivations must begin with either $B_i \to C_iD_j$ or $B_i \to C_jD_i$ for some $B \to CD \in P$ and some $1 \le j \le m$. It is enough to investigate the former case. Then we have $B_i \Rightarrow_{G_{\mathrm{non}}} C_iD_j \Rightarrow^+_{G_{\mathrm{non}}} w_1D_j \Rightarrow^+_{G_{\mathrm{non}}} w_1w_2 = w$ for some $w_1, w_2 \in \Sigma^+$. By induction hypothesis, $C \Rightarrow^+_G w_1$, $w_1$ contains $a_i$, and $D \Rightarrow^+_G w_2$. Hence, $B \Rightarrow_G CD \Rightarrow^+_G w_1D \Rightarrow^+_G w_1w_2 = w$ is a valid derivation by $G$.

($\Leftarrow$): The base case is proved as for the direct implication. If $B \Rightarrow^+_G w$ is not a single-step derivation, then it must start with applying to $B$ some production $B \to CD \in P$. Namely, $B \Rightarrow_G CD \Rightarrow^+_G w_1'D \Rightarrow^+_G w_1'w_2' = w$ for some nonempty words $w_1', w_2' \in \Sigma^+$. Thus, either $w_1'$ or $w_2'$ contains $a_i$; let us say $w_1'$ does. By induction hypothesis, $C_i \Rightarrow^+_{G_{\mathrm{non}}} w_1'$. A letter $a_j$ occurring in $w_2'$ is chosen, and the hypothesis gives $D_j \Rightarrow^+_{G_{\mathrm{non}}} w_2'$. As a result, the derivation $B_i \Rightarrow_{G_{\mathrm{non}}} C_iD_j \Rightarrow^+_{G_{\mathrm{non}}} w_1D_j \Rightarrow^+_{G_{\mathrm{non}}} w_1'w_2' = w$ is valid. □

Let us check that $L(G_{\mathrm{non}}) = \{w \in L(G) \mid w \text{ is not unary}\}$ holds. For the direct implication, assume that $u \in L(G_{\mathrm{non}})$. Its derivation should be $S' \Rightarrow_{G_{\mathrm{non}}} B_iC_j \Rightarrow^+_{G_{\mathrm{non}}} u_1C_j \Rightarrow^+_{G_{\mathrm{non}}} u_1u_2 = u$ for some $S' \to B_iC_j \in P'$ and $u_1, u_2 \in \Sigma^+$. Ignoring the subscripts $i, j$ in this derivation brings us with $S \Rightarrow^*_G u$. Moreover, Lemma 3 implies that $u_1$ and $u_2$ contain $a_i$ and $a_j$, respectively. Thus, $u$ is a nonunary word in $L(G)$.

Conversely, consider a nonunary word $w \in L(G)$. Being nonunary, $|w| \ge 2$, and this means that its derivation by $G$ must begin with a production $S \to BC$. Since $B, C \ne S$, they cannot produce $\varepsilon$, and hence, we have $S \Rightarrow_G BC \Rightarrow^+_G w_1C \Rightarrow^+_G w_1w_2 = w$ for some nonempty words $w_1, w_2 \in \Sigma^+$. Then we can find a letter $a_i$ in $w_1$ and a letter $a_j$ in $w_2$ such that $i \ne j$. Now, Lemma 3 implies $B_i \Rightarrow^+_{G_{\mathrm{non}}} w_1$ and $C_j \Rightarrow^+_{G_{\mathrm{non}}} w_2$. Since $S' \to B_iC_j \in P'$, the derivation $S' \Rightarrow_{G_{\mathrm{non}}} B_iC_j \Rightarrow^+_{G_{\mathrm{non}}} w_1C_j \Rightarrow^+_{G_{\mathrm{non}}} w_1w_2 = w$ is a valid one by $G_{\mathrm{non}}$.

Note that, being thus designed, $G_{\mathrm{non}}$ contains $mh-m+1$ variables. □

We finally observe that in [12] it was proven that there is a constant $c > 0$ such that for infinitely many $h > 0$ there exists a CNFG with $h$ variables generating a unary language such that each equivalent DFA requires at least $2^{ch^2}$ states. This implies that the upper bound given in Theorem 7 cannot be improved.

## 5 Conclusion

We proved that the state cost of the conversion of $n$-state NFAs into Parikh equivalent DFAs is $e^{\Theta(\sqrt{n \cdot \ln n})}$. This is the same cost of the conversion of unary

NFAs into equivalent DFAs. Since in the unary case Parikh equivalence is just equivalence, this result can be seen as a generalization of the Chrobak conversion [1] to the nonunary case. More surprisingly, such a cost is due to the unary parts of the languages. In fact, as shown in Theorem 4, for each $n$-state unary NFA accepting a language which does not contain any unary word there exists a Parikh equivalent DFA with polynomially many states. Hence, while for the transformation from NFAs to equivalent DFAs we need at least two different symbols to prove the exponential gap from $n$ to $2^n$ states and we have a smaller gap in the unary case, for Parikh equivalence the worst case is due only to unary strings.

Even in the proof of our result for CFGs (Theorem 7), the separation between the unary and nonunary parts was crucial. Also in this case, it turns out that the most expensive part is the unary one.

## References


1. Chrobak, M.: Finite automata and unary languages. Theoretical Computer Science 47, 149–158 (1986); Corrigendum, ibid. 302, 497–498 (2003)
2. Esparza, J., Ganty, P., Kiefer, S., Luttenberger, M.: Parikh's theorem: A simple and direct automaton construction. Information Processing Letters 111(12), 614–619 (2011)
3. Ginsburg, S., Rice, H.G.: Two families of languages related to ALGOL. J. ACM 9, 350–371 (1962)
4. Gruska, J.: Descriptional complexity of context-free languages. In: Proceedings of 2nd Mathematical Foundations of Computer Science, pp. 71–83 (1973)
5. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
6. Kopczyński, E., To, A.W.: Parikh images of grammars: Complexity and applications. In: Symposium on Login in Computer Science, pp. 80–89 (2010)
7. Lavado, G.J., Pighizzini, G.: Parikh's Theorem and Descriptional Complexity. In: Bieliková, M., Friedrich, G., Gottlob, G., Katzenbeisser, S., Turán, G. (eds.) SOFSEM 2012. LNCS, vol. 7147, pp. 361–372. Springer, Heidelberg (2012)
8. Lupanov, O.: A comparison of two types of finite automata. Problemy Kibernet. 9, 321–326 (1963) (in Russian); German translation: Über den Vergleich zweier Typen endlicher Quellen. Probleme der Kybernetik 6, 329–335 (1966)
9. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: FOCS, pp. 188–191. IEEE (1971)
10. Moore, F.R.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. IEEE Transactions on Computers C-20(10), 1211–1214 (1971)
11. Parikh, R.J.: On context-free languages. Journal of the ACM 13(4), 570–581 (1966)
12. Pighizzini, G., Shallit, J., Wang, M.: Unary context-free grammars and pushdown automata, descriptional complexity and auxiliary space lower bounds. Journal of Computer and System Sciences 65(2), 393–414 (2002)
13. Rabin, M., Scott, D.: Finite automata and their decision problems. IBM J. Res. Develop. 3, 114–125 (1959)
14. To, A.W.: Parikh images of regular languages: Complexity and applications, arXiv:1002.1464v2 (February 2010)