NON-SELF-EMBEDDING GRAMMARS AND DESCRIPTIONAL COMPLEXITY

Giovanni Pighizzini Luca Prigioniero

Dipartimento di Informatica, Università degli Studi di Milano
via Comelico 39/41, 20135 Milano, Italy
{pighizzini, prigioniero}@di.unimi.it

Abstract
Non-self-embedding grammars are a subclass of context-free grammars which only generate regular languages. The size costs of the conversion of non-self-embedding grammars into equivalent finite automata are studied, by proving optimal bounds for the number of states of nondeterministic and deterministic automata equivalent to given non-self-embedding grammars. In particular, each non-self-embedding grammar of size s can be converted into an equivalent nondeterministic automaton which has an exponential size in s and into an equivalent deterministic automaton which has a double exponential size in s. These costs are shown to be optimal. Moreover, they do not change if the larger class of quasi-non-self-embedding grammars, which still generate only regular languages, is considered. In the case of letter bounded languages, the cost of the conversion of non-self-embedding grammars and quasi-non-self-embedding grammars into deterministic automata reduces to an exponential of a polynomial in s.

1. Introduction

In formal language theory, the most investigated classes are probably those of regular and context-free languages. The interest for them is not purely theoretical, but it is also related to their practical applications as, for instance, the definition of programming language syntax and the construction of lexical and syntactic analyzers [1].

As well-known, the class of context-free languages properly contains the class of regular ones. Roughly speaking, the difference between these two classes is related to the representation of recursive structures (e.g., nested parentheses, nested blocks in programming languages, arithmetic expressions in infix notations) which is possible in context-free languages but not in regular ones. This difference can be emphasized, in terms of recognizers, by observing that regular languages are equivalent to finite automata, while context-free languages are equivalent to pushdown automata, namely finite automata extended with a pushdown store, i.e., the memory structure which allows to implement the recursion.

In one of the first of his pioneering papers on grammars, Chomsky formalized this difference in terms of grammars, by studying the *self-embedding* property [4]. A variable A in a context-free

grammar is *self-embedded* if it is able to reproduce itself in a sentential form, enclosed between two nonempty strings α and β, in symbols $A \stackrel{\star}{\Longrightarrow} \alpha A \beta$. This means that the variable A can generate a "true" recursion that needs an auxiliary memory (typically a stack) to be implemented (in contrast with tail or head recursions, corresponding to the cases in which α or β are empty, respectively, that can be easily eliminated). Chomsky proved that context-free grammars *without* self-embedded variables, namely *non-self-embedding grammars*, only generate regular languages. Hence, the "true" recursion given by self-embedded variables is the additional capability which makes the class of context-free languages larger than the class of regular ones.

The proof given by Chomsky of this result is constructive, namely it provides a method for obtaining a finite automaton equivalent to a given non-self-embedding grammar [4, 5]. A different constructive proof of the same result was given by Anselmo, Giammarresi, and Varricchio [3], by showing a decomposition of non-self-embedding grammars in regular grammars and then iteratively applying regular substitutions to obtain equivalent finite automata. In the same paper, the authors also proved that the size gap between non-self-embedding grammars and equivalent finite automata is at least exponential, by showing the existence of a language (defined over a one-letter alphabet) described by a non-self-embedding grammar of size $O(s)$ for which any equivalent nondeterministic finite automaton requires 2^s many states.

In this paper we continue the investigation of the relationships between the sizes of non-self-embedding grammars, together with one extension of them, and of equivalent finite automata.

It is worthwhile to mention that, in 1971, Meyer and Fischer proved that for any recursive function f and arbitrarily large integer n, there exists a context-free grammar whose description has size n and which generates a regular language, such that any equivalent finite automaton requires at least $f(n)$ states [12]. This means that it is not possible to obtain a recursive bound relating the size of context-free grammars generating regular languages with the number of states of equivalent deterministic finite automata. It is important to notice that the result of Meyer and Fischer was obtained by considering grammars with a two-letter terminal alphabet. The unary, i.e., one-letter, case was studied in 2002 by Pighizzini, Shallit, and Wang, who obtained optimal recursive bounds [15].

In this paper, we show that also in the case of non-self-embedding grammars, the bounds are recursive, independently on the alphabet size. In particular, by inspecting and refining the construction presented in [3], we show that each non-self-embedding grammar of size s can be converted into equivalent nondeterministic and deterministic automata with $2^{O(s)}$ and $2^{2^{O(s)}}$ states, respectively. We also present a family of languages that witness that these gaps cannot be reduced.

We already mentioned the unary case. We remind the reader that restricted to a one-letter terminal alphabet, context-free languages are regular [8]. So one could ask what happens if we consider context-free grammars where the only variables which are allowed to be self-embedded are those generating only unary strings. Let us call such grammars *quasi-non-self-embedding*. Andrei, Cavadini, and Chin proved that quasi-non-self-embedding grammars generate only regular languages, as non-self-embedding ones [2]. Here, we prove that the size costs of the

conversion of quasi-non-self-embedding grammars into equivalent nondeterministic and deterministic finite automata are the same of the conversion of non-self-embedding grammars.

Since the size gap of the conversion into deterministic automata is witnessed by a family of languages defined over a binary alphabet, where in any language each factor of a certain length is required to appear in some strings, it is interesting to ask if the bound can be reduced, when there is not such possibility. For this reason, in the last part of the paper, we consider the case of *letter-bounded languages*, i.e., subsets of $a_1^*a_2^*\cdots a_m^*$, for fixed pairwise distinct letters $a_1, a_2, \ldots, a_m$. While the cost of the conversion from quasi-non-self-embedding grammars of size s into nondeterministic automata remains $2^{O(s)}$, the cost of the conversion into deterministic automata reduces to $2^{O(s^2)}$. As a consequence of the result on the unary case in [15], also these upper bounds are optimal.

2. Preliminaries

Given a set S, let us denote by $\#S$ its cardinality, and by 2^S the family of all its subsets. A strongly connected component (SCC, for short) of a directed graph $G = (V, E)$ is a maximal subset V' of V such that for each pair of vertices $u, v \in V'$, G contains a path from u to v. If $V' = V$, i.e., all the states of the graph form a unique SCC, then G is said to be *strongly connected*. An SCC is *trivial* if it does not contain any loop, namely, it is a single vertex v without the edge (v, v). Otherwise, it is said to be *nontrivial*.

We assume that the reader is familiar with basic notions from automata and formal language theory. The empty string is denoted by ε. Given a string $w \in \Sigma^*$, $w = a_1a_2\cdots a_n$, $a_i \in \Sigma$, $i = 1, \ldots, n$, let us denote by w^R the *reverse* of w, namely the string $a_n\cdots a_2a_1$. For any $h \geq 0$, Σ^h denotes the set of strings of length h over the alphabet Σ. We use DFA and NFA as abbreviations for *deterministic* and *nondeterministic finite automaton*, respectively. A unary automaton (language, respectively) is defined over a one-letter alphabet. A language $L \subseteq \Sigma^*$ is said to be *bounded* if it is a subset of $w_1^*w_2^*\cdots w_m^*$, for some words $w_1, w_2, \ldots, w_m \in \Sigma^*$. In the case these words are pairwise different letters, i.e., $L \subseteq a_1^*a_2^*\cdots a_m^*$, with $a_1, a_2, \ldots, a_m \in \Sigma$ and $a_i \neq a_j$ for $i \neq j$, then L is said to be *letter-bounded*.

A *context-free grammar* (CFG, for short) is a tuple $G = (V, \Sigma, P, S)$, where V is the set of *variables*, Σ is the set of *terminals*, with $V \cap \Sigma = \emptyset$, $S \in V$ is the *initial symbol* and P is the finite set of *productions*, of the form $A \to \alpha$, where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$. The relations $\Rightarrow$ and $\stackrel{\star}{\Longrightarrow}$ are defined in the usual way. By $L(G)$ we denote the language generated by G. For each $A \in V$, $G_{|A} = (V, \Sigma, P, A)$ denotes the grammar obtained from G by taking A as initial symbol. Hence $L(G_{|A})$ is the language generated starting from A.

As a measure for the size of G we consider the total number of symbols used to specify the grammar G, defined as $\mathrm{Symb}(G) = \sum_{(A\to\alpha)\in P}(2 + |\alpha|)$ (cf. [11]).

The *production graph* $\mathcal{P}(G)$ of G is a direct graph which has V as vertex set and contains an edge from A to B, $A, B \in V$, if and only if there is a production $A \to \alpha B\beta$ in P, for

some $\alpha, \beta \in (V \cup \Sigma)^*$.

The grammar G is said to be *right-linear* (*left-linear*, resp.), if each production in P is either of the form $A \to wB$ ($A \to Bw$, resp.), or of the form $A \to w$, for some $A, B \in V$, $w \in \Sigma^*$.

By standard constructions, from each left-linear (or right-linear) grammar G of size s it is possible to obtain an equivalent NFA with $O(s)$ many states. Notice that, by changing the initial symbol in G, the obtained NFA can be different (in the case of a right-linear grammar, we need to change the initial state, while in the case of left-linear grammars we have to change final states).

We are now going to introduce the main notions considered into this paper.

Definition 2.1 *Let $G = (V, \Sigma, P, S)$ be a context-free grammar. A variable $A \in V$ is said to be* self-embedded *when there are two strings $\alpha, \beta \in (V \cup \Sigma)^+$ such that $A \stackrel{\star}{\Longrightarrow} \alpha A \beta$. The grammar G is* self-embedding *(*SE*, for short) if it contains at least one self-embedded variable, otherwise G is* non-self-embedding *(*NSE*, for short). G is* quasi-non-self-embedding *(*qNSE*, for short) when each self-embedded variable generates a unary language.*

Note that in qNSE grammars the self-embedded variables could generate different unary languages on different symbols of Σ.

Chomsky proved that NSE grammars generate only regular languages [4, 5], i.e., they are no more powerful than finite automata. This result has been extended to qNSE grammars in [2], where qNSE grammars are called *one-letter factorizable*.

We point out that, as shown in [3], given a grammar G it is possible to decide in polynomial time whether or not it is NSE. With an easy modification, the same technique can be used to decide if G is qNSE.

The following operation will be fundamental in the paper:

Definition 2.2 ([3]) *Let $G_1 = (V_1, \Sigma_1, P_1, S_1)$ and $G_2 = (V_2, \Sigma_2, P_2, S_2)$, with $V_1 \cap V_2 = \emptyset$ be two* CFG*s. The* $\oplus$-composition *of G_1 and G_2 is the grammar $G = G_1 \oplus G_2 = (V, \Sigma, P, S)$, where $V = V_1 \cup V_2$, $\Sigma = (\Sigma_1 \setminus V_2) \cup \Sigma_2$, $P = P_1 \cup P_2$, and $S = S_1$.*

Intuitively, the grammar $G = G_1 \oplus G_2$ generates all the strings which can be obtained by replacing in any string $w \in L(G_1)$ each symbol $A \in \Sigma_1 \cap V_2$ with a string in Σ_2^* which can be derived from A in the grammar G_2, i.e., which is in the language $L(G_{2|A})$ (notice that the definition of $G_1 \oplus G_2$ does not depend on the initial symbol S_2 of G_2, i.e., by changing the initial symbol of G_2 the resulting grammar G does not change).

The following properties have been stated in [3]:

- If $\Sigma_1 \cap V_2 = \emptyset$ then $L(G_1 \oplus G_2) = L(G_1)$.
- $\oplus$-composition is associative.

- If G_1 and G_2 are NSE grammars, then $G_1 \oplus G_2$ is an NSE grammar.

We also observe that given two CFGs G_1 and G_2, if $L(G_1)$ is regular and, for each $A \in \Sigma_1 \cap V_2$, $L(G_{2|A})$ is regular, then $L(G)$ is regular (regular substitution). In particular, given an NFA M_1 accepting $L(G_1)$ and NFAs M_A accepting $L(G_{2|A})$, for $A \in \Sigma_1 \cap V_2$, we can obtain an NFA M accepting $L(G_1 \oplus G_2)$ by "substituting" each automaton M_A in M_1, namely by replacing in M_1 each transition from a state p to a state q on the symbol A with a copy of the NFA M_A. More precisely, using ε-moves, in M the initial state of the copy of M_A can be reached from the state p while the state q can be reached from each final state in the copy. If M_1 has s_1 states (hence $O(s_1^2)$ transitions) and each M_A has at most s_2 states, then the resulting NFA M has $O(s_1^2 s_2)$ states. This number can be reduced to $O(s_1 s_2)$ by using, for each state p, only one copy of M_A for all outgoing transitions from p on the symbol A, and connecting with ε-transitions the final states of the copy to all the states q which in M_1 are reachable from p by transitions on A.

3. Converting NSE Grammars into Automata

In [3] the authors gave an interesting alternative proof of the above mentioned Chomsky's result on the regularity of languages generated by NSE grammars. In particular, they obtained the result as a consequence of the following theorem:

Theorem 3.1 ([3, Thm. 2]) *Let $G = (V, \Sigma, P, S)$ be an NSE grammar. Then there exist n grammars $G_1, G_2, \ldots, G_n$, $n > 0$, such that $G = G_1 \oplus G_2 \oplus \cdots \oplus G_n$ and, for $i = 1, \ldots, n$, G_i is either left-linear or right-linear.*

The proof of Theorem 3.1 is constructive: it presents a method for obtaining the grammars $G_1, G_2, \ldots, G_n$ from the given NSE grammar G. Since this method is important to obtain the state upper bounds for NFAs and DFAs equivalent to NSE grammars presented in this section and other results in the paper, we now summarize how grammars $G_1, G_2, \ldots, G_n$ are obtained in [3] (for a detailed presentation of a related decomposition see [9, Sect. 3.5]).

- Let n be the number of SCCs in the production graph $\mathcal{P}(G)$.
- If $\mathcal{P}(G)$ is strongly connected then G is either a left-linear or a right-linear grammar, hence $G = G_1$ is regular.
- Otherwise, the SCCs of $\mathcal{P}(G)$ are considered in some topological order and, for $i = 1, \ldots, n$, the grammar $G_i = (V_i, \Sigma_i, P_i, S_i)$ is defined as follows:
  - V_i is the set of variables in the ith SCC,
  - $\Sigma_i = \Sigma \cup \bigcup_{j>i} V_j$, for technical reasons we also set $\Sigma_0 = \{S\}$.
  - P_i is obtained by restricting P to productions whose left-hand side variables are in V_i,
  - S_i is an element in $V_i \cap \Sigma_{i-1}$.

  Since $\mathcal{P}(G_i)$ is strongly connected, G_i is either a left-linear or a right-linear grammar, for $i = 1, \ldots, n$.

We are now going to estimate the size of NFAs and DFAs equivalent to the NSE grammar G given in Theorem 3.1. Let us suppose that the size of G is s. Then $n \leq s$. Furthermore, $s = s_1 + s_2 + \cdots + s_n$, where s_i is the size of G_i, $i = 1, \ldots, n$.

Now, from each grammar G_i, we obtain a family of NFAs $\{M_{i,A} \mid A \in \Sigma_{i-1} \cap V_i\}$, such that each NFA $M_{i,A}$ has size $O(s_i)$ and generates the language $L(G_{i|A})$.

To obtain an NFA M accepting $L(G)$, we iteratively construct automata M_i, for $i = 1, \ldots, n$, accepting $L(G_1 \oplus G_2 \oplus \cdots \oplus G_i)$ as follows. Let us start by taking $M_1 = M_{1,S}$. For $i > 1$, the automaton M_i is obtained by substituting in the automaton M_{i-1} each transition labeled by $A \in \Sigma_{i-1} \cap V_i$ with the NFA $M_{i,A}$, as explained in Section 2.

At the end of this process, we finally obtain $M = M_n$, which accepts $L(G)$ and has $O(s_1 s_2 \cdots s_n)$ many states. Since $s_1 + \cdots + s_n = s$, we get that $s_1 \cdots s_n = 2^{\log(s_1 \cdots s_n)} = 2^{\log s_1 + \cdots + \log s_n} \leq 2^s$. Hence, we conclude that M has $2^{O(s)}$ many states. Considering also the cost of the conversion of NFAs into equivalent DFAs, we obtain the following:

Theorem 3.2 *Let G be an* NSE *grammar of size s. Then there exist an* NFA *and a* DFA *accepting $L(G)$ with $2^{O(s)}$ and $2^{2^{O(s)}}$ many states, respectively.*

4. Optimality

In this section we prove that the exponential and double exponential state upper bounds for the conversion of NSE grammars into NFAs and DFAs given in Theorem 3.2 cannot be reduced. To this aim, we now introduce a family of witness languages.

Given an integer $h > 0$, consider the language $L_h \subseteq \{a, b\}^*$ defined as the set of strings composed of k blocks $w_1 w_2 \cdots w_k$ each of length h, for some $k > 1$, such that the last block w_k is the reverse of one of the first $k - 1$ blocks. Formally,

$$L_h = \{w_1 w_2 \cdots w_{k-1} w_k \mid k > 1, w_i \in \{a, b\}^h, i = 1, \ldots, k, \text{ and } \exists j, 1 \leq j < k, \text{ s.t. } w_j = w_k^R\}.$$

Let us define the grammar $G_h = (V, \Sigma, P, S)$ generating L_h, with $V = \{S, C_1, \ldots, C_h, A_1, \ldots, A_h\}$, $\Sigma = \{a, b\}$, and the productions are:

- $S \to C_1 A_1 \mid A_1$
- $C_i \to aC_{i+1} \mid bC_{i+1}$ for $1 \leq i < h$
- $C_h \to a \mid b \mid aC_1 \mid bC_1$
- $A_i \to aA_{i+1}a \mid bA_{i+1}b$ for $1 \leq i < h$
- $A_h \to aa \mid bb \mid aC_1 a \mid bC_1 b$

It can be observed that from the variable C_1 it is possible to derive one or more blocks of h terminal symbols, i.e., for $x \in \Sigma^*$, $C_1 \stackrel{\star}{\Longrightarrow} x$ if and only if $x \in (\Sigma^h)^+$. From the variable A_1, after expanding variables A_i, $i = 1, \ldots, h$, we generate sentential forms as $A_1 \stackrel{\star}{\Longrightarrow} ww^R$ or $A_1 \stackrel{\star}{\Longrightarrow}$

wC_1w^R where $w \in \Sigma^h$. Then A_1 generates all terminal strings of the form $w(\Sigma^h)^*w^R$, where $w \in \Sigma^h$. Using the initial symbol S, it is possible to combine the two variables C_1 and A_1 to obtain an arbitrarily long sequence of blocks of length h followed by the blocks w_j and w_k that enclose another arbitrarily long sequence of blocks of h symbols.

Observe that the size of the grammar G_h is linear in the parameter h, namely $\mathrm{Symb}(G_h) = O(h)$. Using a distinguishability argument, we are going to show that any DFA accepting L_h requires a number of states which is double exponential in h.

Let $w_1, w_2, \ldots, w_{2^h}$ be the list of all the strings in Σ^h in lexicographical order, and $S = 2^{\Sigma^h}$ be the family of all the subsets of Σ^h. For each $s \in S$, we consider the string $v_s = w_{i_1}w_{i_2}\cdots w_{i_k}$, where $1 \leq i_1 < i_2 < \ldots < i_k \leq 2^h$, $k \geq 0$, and $s = \{w_{i_1}, w_{i_2}, \ldots, w_{i_k}\}$. Given two different subsets $s', s'' \in S$, let $x \in \Sigma^h$ be a string such that $x \in (s' \cup s'') \setminus (s' \cap s'')$. Then, the string x^R distinguishes $v_{s'}$ and $v_{s''}$, i.e., exactly one between $v_{s'}x$ and $v_{s''}x$ belongs to L_h. Hence, each DFA accepting L_h needs at least $\#S$ many states. This gives a 2^{2^h} lower bound for the size of any DFA accepting L_h and a 2^h lower bound for the size of any NFA accepting L_h. This allows us to conclude that the gaps given in Theorem 3.2 cannot be reduced.

Let us observe that the above result is strictly dependent on the structure of considered strings. In Section 6 we show that if a given grammar generates a letter-bounded language, then the double exponential gap cannot be reached.

For the sake of completeness we describe how an NFA N_h accepting L_h works.

- N_h starts to scan the input string by skipping the first $j-1$ blocks and by nondeterministically guessing which is the block w_j. To this aim, N_h uses a counter modulo h, that requires h states.

- After that, the automaton has to read and finally save the content of w_j into its finite state control. The transition graph of this part corresponds to a complete binary tree of height h starting from the initial state and in which each of the 2^h leaves represents a string $w \in \Sigma^h$. As a consequence, the number of states required for this part is $\sum_{i=1}^{h} 2^i = 2^{h+1} - 2$.

- After reaching a leaf, it is necessary to skip $k - j + 1$ blocks — again, this can be done in a nondeterministic way — and, finally, N_h has to verify that the last block corresponds to the stored word. This can be done by comparing the next input symbol with the last symbol in the string w stored in the control; if they are different, then the computation stops, otherwise the last symbol of w is removed and the computation continues in the same way (these operations require $O(h)$ states for each leaf).

From the above description, summing up the states of the automaton, it is possible to derive an upper bound on the size of the NFA N_h that is exponential in the size of the NSE grammar G_n. More precisely, we can show that N_h can be implemented by using $h + \sum_{i=1}^{h} 2^i + 2^h(h-1) + \sum_{i=0}^{h-1} 2^i = (2+h)2^h + h - 3$ many states.

5. Converting qNSE-grammars into Automata

It is well-known that unary context-free languages, i.e., languages generated by CFGs with a one-letter terminal alphabet, are regular [8]. Hence, this holds even for unary grammars containing self-embedded variables.

In this section we consider qNSE grammars, namely CFGs in which the only self-embedded variables are unary. As observed in [2], all the languages generated by these grammars are regular. We are now going to describe and refine the idea used to prove that result, in order to extend Theorem 3.2 to qNSE grammars. First, it is useful to have an upper bound for the cost of the conversion of unary CFGs into equivalent finite automata.

Lemma 5.1 *Each unary* CFG G *of size* s *can be transformed into an equivalent* NFA *with* $2^{O(s)}$ *states and into an equivalent* DFA *with* $2^{O(s^2)}$ *states.*

Proof. In [15, Thms. 5, 6], it was proved that for any unary CFG in Chomsky normal form with h variables there exists an equivalent NFA with at most $2^{2h-1}+1$ states and, when $h \geq 2$, an equivalent DFA with less than 2^{h^2} states. By inspecting the arguments used in the proof, it can be observed that these bounds do not change if unary CFGs whose production right-hand sides have length at most 2 are considered. Each CFG can be turned in this form with a linear increase of the size. □

Let $G = (V, \Sigma, P, S)$ be a qNSE grammar of size s. We proceed as follows:

- We can suppose that each production right-hand side is either a sequence of variables or a single terminal, i.e., $\alpha \in V^* \cup \Sigma$ for each $A \to \alpha$ in P. This (at most) linearly increases the size of the grammar.
- Let $G' = (V', \Sigma', P', S)$ and $G'' = (V'', \Sigma, P'', S'')$ be the grammars obtained from G by choosing as V' the set of variables in G which generate at least one nonunary terminal string and as $V'' = V \setminus V'$ the set of *unary variables*, namely variables generating unary languages, $\Sigma' = \Sigma \cup V''$, P' is the set of productions in P having left-hand side in V', $P'' = P \setminus P'$, and S'' is a variable in V''. It can be verified that $G = G' \oplus G''$. Furthermore, if the sizes of G' and G'' are s' and s'', then $s' + s'' = s$. Notice that $\Sigma' = V''$ under the hypothesis that useless variables have been removed from G.
- Since its variables are nonunary, the grammar G' is NSE. Hence, using Theorem 3.2, there exists an NFA M' with $2^{O(s')}$ states accepting $L(G')$.
- For each unary variable $A \in V''$, from the grammar $G''_{|A}$ we obtain a unary NFA M_A accepting $L(G''_{|A})$ with $2^{O(s'')}$ states (Lemma 5.1).
- Finally, we substitute in the automaton M' the automata M_A, as described in Section 2. Hence, we obtain an NFA M with $2^{O(s')}2^{O(s'')}$ states accepting the language $L(G) = L(G' \oplus G'')$. Since $s' + s'' = s$, the total number of states of M is $2^{O(s)}$.

From the previous discussion, we obtain the extension of Theorem 3.2 to qNSE grammars:

Theorem 5.2 *Let* G *be a* qNSE *grammar of size* s*. Then there exist an* NFA *and a* DFA *accepting* $L(G)$ *with* $2^{O(s)}$ *and* $2^{2^{O(s)}}$ *many states, respectively.*

The optimality of the bounds in Theorem 5.2 follows from the optimality of those in Theorem 3.2 presented in Section 4.

6. The Letter-Bounded Case

In this section we consider NSE and qNSE grammars generating letter-bounded languages, namely subsets of $a_1^*a_2^* \cdots a_m^*$, for a given alphabet $\Sigma = \{a_1, a_2, \ldots, a_m\}$. We prove that the double exponential gap in Theorem 5.2 from qNSE grammars to DFAs cannot be reached. Indeed, we reduce it to a simple exponential of the square of the size of the grammar.

Given a qNSE grammar $G = (V, \Sigma, P, S)$ of size s such that $L(G) \subseteq a_1^*a_2^* \cdots a_m^*$ is a letter bounded language, we apply the same procedure described in Section 5, in order to obtain an NFA M with $2^{O(s)}$ states accepting $L(G)$.

At this point we have to turn M into an equivalent DFA. To this aim, we could use the result presented in [10] for the conversion of NFAs accepting letter-bounded languages into equivalent DFAs, which gives a subexponential, but still superpolynomial upper bound that, in the worst case, cannot be reduced. However, this would produce a double exponential upper bound, for the conversion from a qNSE grammar. We will be able to do better after refining the arguments used in the previous sections and by applying some results related to state complexity of unary automata.

We present some observations which will allow to achieve our goal.

- For each left-linear (or right-linear) grammar whose production graph is strongly connected, it is possible to find an equivalent NFA (with a linear number of states in the size of the grammar) such that its transition graph consists of one SCC containing the initial state and of a finite number of paths from the SCC to the unique final state. Each state on these paths (except the beginning one, and including the final state) is a trivial SCC.
- Substituting one transition in a nontrival SCC $\mathcal{C}$ by an NFA (without useless states) produces a unique SCC, namely an "expanded version" of $\mathcal{C}$.
- Substituting one transition connecting two SCCs by an NFA produces a nontrivial SCC if and only if the NFA used in the substitution contains an SCC.

Using these observations, we now inspect the structure of the NFA M in order to see in which way nontrivial SCCs in its transition graph have been introduced (it is useful to remember how M has been obtained according the procedures described in Sections 3 and 5). The transition graph of M *could* contain only the following SCCs:

- An "initial" nontrivial SCC which has been obtained by substitutions starting from the SCC of G containing the initial symbol.
- For each symbol $A \in V$, one or more SCCs which have been obtained starting from a point when a transition labeled by A has been substituted. Note that all the SCCs obtained for the same symbol A are identical.

We now study the cost of making deterministic each NFA that can be obtained from M by fixing a letter $a_j \in \Sigma$ and considering the part of M working on this letter. In particular, we consider the part of the automaton M consisting of all states with in-going or out-going transitions on a_j, besides all states that are connected by ε-transitions to this part of M. The transitions we consider are all a_j-transitions and ε-transitions between the states we selected. Furthermore, we pick up a state as initial state and a set of final states. From the automaton so obtained, we construct an equivalent NFA $\widetilde{M}$ with the same set of states but without ε-transitions (hence, we remove ε-transitions without making the automaton deterministic). We can observe that all the SCC in $\widetilde{M}$ still represent some of above listed SCCs of M. We want to obtain an upper bound for the number of states of a DFA equivalent to $\widetilde{M}$.

To this aim the following result on unary NFAs will be used:

Lemma 6.1 *Let $\widetilde{M}$ be a unary N-state* NFA*, and $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ be the nontrivial* SCC*s in its transition graph. For $i = 1, \ldots, k$, let us choose an integer ℓ_i, with $0 < \ell_i \leq N$, such that there is a state p_i in $\mathcal{C}_i$, with a loop of length ℓ_i from p_i to p_i. Then there exists a* DFA *equivalent to $\widetilde{M}$ with an initial path of $O(N^2)$ states and a loop of ℓ states, where ℓ is the least common multiple of $\ell_1, \ell_2, \ldots, \ell_k$.*

Proof. Suppose $\widetilde{M}$ has set of states Q, input alphabet $\{a\}$, transition function δ, initial state q_I, and set of final states F. As in [7], for $q_1, q_2 \in Q$, $m \geq 0$, we write $q_1 \xrightarrow{a^m} q_2$, to denote a path from state q_1 to state q_2 on input a^m. To prove the lemma we make use of the following result which has been obtained in [7] to present a transformation of unary NFAs into Chrobak normal form [6]:

> **Lemma 6.2 ([7, Lemma 3.1])** *If there exists a computation path $q_1 \xrightarrow{a^\alpha} q \xrightarrow{a^\beta} q_2$ for some states q_1, q, q_2 in $\widetilde{M}$, and if there also exists a loop $q \xrightarrow{a^\lambda} q$, such that $\beta \geq N \cdot \lambda$, then the path $q_1 \xrightarrow{a^\alpha} q \xrightarrow{a^\beta} q_2$ can be replaced by an equivalent path $q_1 \xrightarrow{a^\alpha} q \xrightarrow{a^\lambda} q \xrightarrow{a^{\beta-\lambda}} q_2$.*

From Lemma 6.2 we now derive the following:

> **Claim** *If there is a path $q_1 \xrightarrow{a^m} q_2$ with $m \geq 3N^2 + N$ such that the first state in a nontrivial* SCC *reached in it belongs to the* SCC *$\mathcal{C}_i$, $1 \leq i \leq k$, then there is an equivalent path $q_1 \xrightarrow{a^\alpha} p_i \xrightarrow{a^{\ell_i}} p_i \xrightarrow{a^{\beta-\ell_i}} q_2$, with $\alpha + \beta = m$.*
>
> *Proof.* Let us start by decomposing the given path as $q_1 \xrightarrow{a^\alpha} q \xrightarrow{a^\beta} q_2$, where $\alpha+\beta = m$ and q is the first state which is reached along the path and belongs to a nontrivial SCC. Hence, $0 \leq \alpha < N$ and $\beta > 3N^2$. Let $\mathcal{C}_i$ be the SCC which contains the state q, and $0 < \lambda \leq N$ be the length of a simple loop from q to q. By Lemma 6.2, there exists an equivalent path $q_1 \xrightarrow{a^\alpha} q \xrightarrow{a^\lambda} q \xrightarrow{a^{\beta-\lambda}} q_2$. In the case $q = p_i$ and $\lambda = \ell_i$, the proof of the claim is completed. Otherwise, as observed in [13], we can repeatedly apply Lemma 6.2, to obtain equivalent paths of the form
>
> $$q_1 \xrightarrow{a^\alpha} q \underbrace{\xrightarrow{a^\lambda} q \xrightarrow{a^\lambda} \cdots \xrightarrow{a^\lambda}}_{t \text{ times}} q \xrightarrow{a^{\beta-t\lambda}} q_2 \tag{1}$$

for each t such that $\beta - (t-1)\lambda \geq N\lambda$.

Now consider two simple paths $q \xrightarrow{a^{\lambda'}} p_i$ and $p_i \xrightarrow{a^{\lambda''}} q$ in $\mathcal{C}_i$. Then $0 < \lambda', \lambda'' < N$ and in $\mathcal{C}_i$ there exists a (not necessarily simple) loop of length $\lambda' + \lambda'' < 2N - 1$ from q to q which visits p_i. By choosing $t = \lambda' + \lambda''$ in (1), we obtain $\lambda' + \lambda''$ repetitions of the loop of length λ from q to q. We can replace them by λ repetitions of the loop of length $\lambda' + \lambda''$ from q to q that visits p_i, and decompose the path so obtained as

$$q_1 \xrightarrow{a^{\alpha}} q \xrightarrow{a^{\lambda'}} p_i \xrightarrow{a^{\lambda''}} q \xrightarrow{a^{(\lambda-1)(\lambda'+\lambda'')}} q \xrightarrow{a^{\beta-\lambda(\lambda'+\lambda'')}} q_2 \tag{2}$$

So, replacing α by $\alpha+\lambda'$ and β by $\beta-\lambda'$, we have $\alpha+\beta = m$, $q_1 \xrightarrow{a^{\alpha}} p_i \xrightarrow{a^{\beta}} q_2$. Since β is large enough, according to Lemma 6.2, there exists an equivalent path $q_1 \xrightarrow{a^{\alpha}} p_i \xrightarrow{a^{\ell_i}} p_i \xrightarrow{a^{\beta-\ell_i}} q_2$. This completes the proof of the claim. □

For $q_1, q_2 \in Q$, $1 \leq i \leq k$, we now consider the language $L^{(i)}_{q_1,q_2}$ consisting of all strings a^m having a path $q_1 \xrightarrow{a^m} q_2$ such that the first state in a nontrivial SCC reached in it belongs to $\mathcal{C}_i$. As a consequence of the previous claim, we get that $a^m \in L^{(i)}_{q_1,q_2}$ if and only if $a^{m+\ell_i} \in L^{(i)}_{q_1,q_2}$ for each $m \geq 3N^2 + N$. Hence, the language $L^{(i)}_{q_1,q_2}$ is a accepted by a DFA with an initial path and a loop consisting of $3N^2 + N$ and ℓ_i many states, respectively.

To complete the proof we observe that the language L accepted by $\widetilde{M}$ can be expressed as

$$L = L_0 \cup \bigcup_{i=1}^{k} \bigcup_{q_f \in F} L^{(i)}_{q_I,q_f}$$

where L_0 is the language consisting of strings of length less than $3N^2+N$ belonging to L, which is accepted by a DFA with no more than $3N^2 + N$ states. From the results concerning the state complexity of the union of languages accepted by unary DFAs [14, Thm. 4], we conclude that there exists a DFA equivalent to $\widetilde{L}$ with an initial path of $3N^2 + N$ states and a loop of ℓ many states, where ℓ is the least common multiple of $\ell_1, \ell_2, \ldots, \ell_k$. □

To obtain an upper bound for the number of states of a DFA equivalent to the automaton $\widetilde{M}$ we are considering, we first observe that $\widetilde{M}$ has $N = 2^{O(s)}$ many states. In each nontrivial SCC $\mathcal{C}_i$ of its transition graph, we choose a loop length ℓ_i, with $0 < \ell_i \leq N$. Since all SCCs which have been obtained starting from a same variable A are identical, we can choose for them a same length ℓ_A. So, we obtain $O(s)$ many different lengths. Furthermore, we have to choose a further loop length for the SCC originated from S, if it is not trivial. According to Lemma 6.1, we can obtain a DFA equivalent to $\widetilde{M}$ with an initial path of $2^{O(s)}$ states, and a loop of ℓ states, where ℓ is the least common multiple of the loop lengths we have selected. Since we have chosen $O(s)$ many different lengths and each one of them is bounded by $N = 2^{O(s)}$, we obtain that ℓ is bounded by $2^{O(s^2)}$. Hence there exists a DFA equivalent to $\widetilde{M}$, with at most $2^{O(s^2)}$ states.

To complete the construction, we now transform the NFA M accepting $L(G)$ into an equivalent DFA, by combining the previous arguments with the "incremental" technique developed in [10]. Let us start by modifying M to make deterministic the section which works on symbol a_1. Then we modify the section which works on a_2 by determinizing each automaton which

corresponds to entering this section from a different final state of the previous section, and so on. For the state estimation, we can proceed as in [10], with the difference that, according to the previous discussion, each deterministic automaton we introduce has $2^{O(s^2)}$ states. This finally leads to get a total number of states for the final DFA which is still of the order of $2^{O(s^2)}$. By summarizing, we got the following result:

Theorem 6.3 *Let G be a* qNSE *grammar of size s generating a letter-bounded language over a fixed alphabet. Then there exist an* NFA *and a* DFA *accepting $L(G)$ with $2^{O(s)}$ and $2^{O(s^2)}$ many states, respectively.*

We point out that in [15] it was proved that for infinitely many integer $h > 0$ there exists a unary language generated by a CFG in Chomsky normal form with h variables such that each equivalent DFA needs at least 2^{ch^2} states, where $c > 0$ is a constant. This gives a lower bound that matches with the upper bound we have stated in Theorem 6.3.

References

[1] A. V. AHO, M. S. LAM, R. SETHI, J. D. ULLMAN, *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[2] S. ANDREI, S. V. CAVADINI, W. CHIN, A New Algorithm for Regularizing One-Letter Context-Free Grammars. *Theor. Comput. Sci.* 306 (2003) 1-3, 113–122.
http://dx.doi.org/10.1016/S0304-3975(03)00215-9

[3] M. ANSELMO, D. GIAMMARRESI, S. VARRICCHIO, Finite Automata and Non-self-Embedding Grammars. In: J. CHAMPARNAUD, D. MAUREL (eds.), *Implementation and Application of Automata, 7th International Conference, CIAA 2002, Tours, France, July 3-5, 2002, Revised Papers*. Lecture Notes in Computer Science 2608, Springer, 2002, 47–56.
http://dx.doi.org/10.1007/3-540-44977-9_4

[4] N. CHOMSKY, On Certain Formal Properties of Grammars. *Information and Control* 2 (1959) 2, 137–167.
https://doi.org/10.1016/S0019-9958(59)90362-6

[5] N. CHOMSKY, A Note on Phrase Structure Grammars. *Information and Control* 2 (1959) 4, 393–395.
http://dx.doi.org/10.1016/S0019-9958(59)80017-6

[6] M. CHROBAK, Finite Automata and Unary Languages. *Theor. Comput. Sci.* 47 (1986) 3, 149–158.
http://dx.doi.org/10.1016/0304-3975(86)90142-8

[7] V. GEFFERT, Magic Numbers in the State Hierarchy of Finite Automata. *Inf. Comput.* 205 (2007) 11, 1652–1670.
https://doi.org/10.1016/j.ic.2007.07.001

[8] S. GINSBURG, H. G. RICE, Two Families of Languages Related to ALGOL. *J. ACM* 9 (1962) 3, 350–371.
http://doi.acm.org/10.1145/321127.321132

[9] M. A. HARRISON, *Introduction to Formal Language Theory*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1978.

[10] A. HERRMANN, M. KUTRIB, A. MALCHER, M. WENDLANDT, Descriptional Complexity of Bounded Regular Languages. In: C. CÂMPEANU, F. MANEA, J. SHALLIT (eds.), *Descriptional Complexity of Formal Systems - 18th IFIP WG 1.2 International Conference, DCFS 2016, Bucharest, Romania, July 5-8, 2016. Proceedings*. Lecture Notes in Computer Science 9777, Springer, 2016, 138–152.
https://doi.org/10.1007/978-3-319-41114-9_11

[11] A. KELEMENOVÁ, Complexity of Normal Form Grammars. *Theor. Comput. Sci.* 28 (1984), 299–314.
http://dx.doi.org/10.1016/0304-3975(83)90026-9

[12] A. R. MEYER, M. J. FISCHER, Economy of Description by Automata, Grammars, and Formal Systems. In: *12th Annual Symposium on Switching and Automata Theory, East Lansing, Michigan, USA, October 13-15, 1971*. IEEE Computer Society, 1971, 188–191.
https://doi.org/10.1109/SWAT.1971.11

[13] G. PIGHIZZINI, Investigations on Automata and Languages Over a Unary Alphabet. *Int. J. Found. Comput. Sci.* 26 (2015) 7, 827–850.
http://dx.doi.org/10.1142/S012905411540002X

[14] G. PIGHIZZINI, J. SHALLIT, Unary Language Operations, State Complexity and Jacobsthal's Function. *Int. J. Found. Comput. Sci.* 13 (2002) 1, 145–159.
https://doi.org/10.1142/S012905410200100X

[15] G. PIGHIZZINI, J. SHALLIT, M. WANG, Unary Context-Free Grammars and Pushdown Automata, Descriptional Complexity and Auxiliary Space Lower Bounds. *J. Comput. Syst. Sci.* 65 (2002) 2, 393–414.
http://dx.doi.org/10.1006/jcss.2002.1855