Determining exponential density and maximal encoding capabilities of a regular language ⋆

Stavros Konstantinidis and Nicolae Santean

Department of Mathematics and Computing Science, Saint Mary's University, Halifax, Nova Scotia, B3H 3C3 Canada,
s.konstantinidis@smu.ca, nic.santean@smu.ca

Abstract. The density of a language is the function that returns, for each n, the number of words in the language of length n. In the first place, we consider deciding whether the density of a given regular language L is exponential. This question can be answered in linear time when L is given via a DFA. We show that the same question can be decided in quadratic time when L is given via an NFA. It turns out that this question is equivalent to whether L is an encoding language: it includes xD^*y, for some nontrivial code D and words x,y. Then, motivated by certain coding techniques for reliable DNA computing, we consider the problem of characterizing nontrivial languages D that are maximal with the property that D^* is contained in the subword closure of a given set S of words of some fixed length k – this closure is simply the set of all words whose subwords of length k must be in S. We provide a deep structural characterization of these languages D, which leads to polynomial time algorithms for computing such languages.

Key words: algorithm, automaton, code, density, maximal, regular language, subword closure

1 Introduction

Following [17], we define the density of a language L to be the function that returns, for every nonnegative integer n, the number of words in L of length n. This concept is of central importance in language theory. In particular, [17] and [14] characterize regular languages of exponential density, where the characterization of [14] leads to a linear time algorithm for deciding whether a regular language is of exponential density when L is given via a deterministic finite automaton (DFA). This characterization is very simple: the DFA has a state that belongs to two different loops – we note that the same idea was used in [4] in the context of encoding data into DNA languages that are described via certain DFAs.

Here, in the first place, we consider the question of characterizing regular languages of exponential density when they are given via nondeterministic finite automata (NFAs). Our characterization is that the NFA has a strongly connected component containing two paths of the same length, starting at the same state, and having different labels. This characterization leads to a quadratic time algorithm deciding whether the given NFA language is of exponential density. We also observe that a language L has exponential density if and only if L is an encoding language – see the next section for precise definitions.

In the second part of the paper we consider the case where $L = S^{\otimes}$. Then L is an encoding language if and and only if $C^* \subseteq L$, for some two-element code C whose words are of the same length. Here $S^{\otimes}$ is the subword closure of a given nonempty set S of words of some fixed length k. This concept has been considered recently in the context of coding for reliable DNA computing – see [11], [3], [4] – as well as in state complexity considerations [1]. The language $S^{\otimes}$ is simply the set of all words whose subwords (factors) of length k must be in S. In relevant applications, the set S represents a subword constraint and the fact that some language is a subset of $S^{\otimes}$ means that the language satisfies that constraint. In DNA computing, for instance, it is often desirable that no two, possibly long, DNA molecules in the test tube contain two short blocks of k pairwise complementary DNA bases, as this would allow a sufficiently strong bond to form between these molecules. In this case, the constraint S is a set of words of length k representing

⋆ Research supported by NSERC.

short molecules that are not pairwise complementary, and the language $S^{\otimes}$ represents all molecules (possibly arbitrarily long) such that there can be no sufficiently strong bond between any two of them.

In [3] and [4], the authors consider the problem of encoding arbitrary sequences of data blocks (fixed-length words in Σ^m, for some alphabet Σ and integer $m \geq 1$) into the words of $S^{\otimes}$, for a given constraint S. More specifically, a bijective mapping (an encoding) from Σ^m to some D is wanted such that $D^* \subseteq S^{\otimes}$. The authors of those papers propose and implement a method for such encodings.

In this paper, motivated by the above problems, but independently of any application, we consider the problem of characterizing *nontrivial* languages D whose words are of length at least k and are maximal with the property "$D^* \subseteq S^{\otimes}$" – the nontrivial requirement for D ensures that D^* is of exponential density and has a good encoding capability. We obtain a *complete* structural characterization of these languages D, which leads to polynomial time algorithms for computing such languages, and a better understanding of the encoding method in [3] and [4]. We note that the more general question of computing maximal regular languages D such that $D^* \subseteq R$, where R is any regular language, has been solved recently in [12] using different tools. However, these tools lead to an algorithm with an exponential number of steps – see also Section 7 for further comments.

The paper is organized as follows. Section 2 contains the basic notation and terminology about regular languages and automata, as well as relevant background on the subword closure operation. In Section 3, we consider the question of whether a given regular language L is of exponential density (equivalently, whether it is an encoding language). Section 4 deals with characterizing structurally nontrivial languages D whose words are of length at least k and are maximal with "$D^* \subseteq S^{\otimes}$". Our characterization is used in Section 5 to evaluate the encoding method of [3] and [4] that was mentioned before. Moreover, this characterization is used in Section 6 to obtain polynomial algorithms for constructing certain maximal languages D. Finally, Section 7 contains a few concluding remarks and suggestions for future research.

2 Basic Notation and Background

We begin this section with notation and concepts on words and languages, and then on finite automata. We use [13] as a general reference.

2.1 Words, Languages, Codes

For a set S, we denote by $|S|$ the cardinality of S. We consider an arbitrary alphabet Σ containing at least two symbols. As usual, the set of all words over Σ is denoted as Σ^*. We write λ for the empty word and Σ^+ for the set of all nonempty words. The length of a word w is the number of alphabet symbols occurring in w and is denoted as $|w|$. For an integer $n \geq 0$, the expression $(w)^n$ is the word consisting of n copies of w. A *prefix* (resp. *suffix*, *subword*) of a word w is any word u such that $w = ux$ (resp. $w = xu$, $w = xuy$) for some words x, y. A subword of w is also called a factor, or infix, of w.

A language is any set of words. A word w is called an L-word if $w \in L$. As usual, for any integer $n \geq 0$, if L is a language then L^n is the language whose words consist of any n concatenated words from L. In particular Σ^n is the set of all words of length n. Also, L^* is the union of L^n, for all $n \geq 0$, and $L^+ = L^* - \{\lambda\}$. For any word x and language L we use the special notation

$$x^{-1}L = \{z \in \Sigma^* \mid xz \in L\}.$$

In particular, if x is a prefix of some word w, then $x^{-1}w$ is the suffix z of w such that $w = xz$. A language C is called a (uniquely decodable) *code* if, for every word $w \in C^+$, there is exactly one sequence $(w_1, \ldots, w_n)$ of C-words whose concatenation is equal to w, that is, $w = w_1 \cdots w_n$. In particular, any language whose words are of some fixed length is always a code (usually called a uniform, or block, code). A language L is called

maximal with respect to some property '$\mathscr{P}$', if any language L' containing L and satisfying '$\mathscr{P}$' is equal to L.

The *density* of a language L is the function d_L that maps every nonnegative integer n to $\mathrm{d}_L(n)$ = the number of L-words of length n. We say that a regular language L has *exponential density* if the density of L is not polynomially upper-bounded – see below for the definition of regular language. This definition is justified by a result of [17] stating that the density of any regular language is either polynomially upper-bounded or has a subsequence of order $\Omega(2^n)$. A language D is called *nontrivial* if it contains two words w_1, w_2 such that $w_1w_2 \neq w_2w_1$. Note that, in this case, the set $\{w_1w_2, w_2w_1\}$ is a two-element fixed-length code, and is a subset of D^* – we shall see later that this property of D ensures that D^* is of exponential density and has a good encoding capability.

2.2 Automata, graphs, cycles

Recall that a complete deterministic finite automaton is a quintuple $M = (\Sigma, K, \delta, s, F)$ such that K is the nonempty set of states, s is the start state, F is the set of final states and $\delta : K \times \Sigma \to K$ is the transition function, which can be extended as $\delta : K \times \Sigma^* \to K$ in the usual way. If the function δ is partial then M is not complete – we simply call it a DFA. A nondeterministic finite automaton (NFA) is a quintuple $M = (\Sigma, K, T, s, F)$ such that K, s, F are as in the case of a deterministic automaton, and T is the finite set of *transitions*, which are triples of the form (p, σ, q) with $\sigma \in \Sigma$ and $p, q \in K$. In this case, we say that the transition is going out of the state p. As usual, a deterministic automaton is a special type of nondeterministic one where $(p, \sigma, q) \in T$ exactly when $\delta(p, \sigma) = q$.

The NFA M can be viewed as a directed labeled graph having K as the set of vertices and any triple (p, σ, q) as a labeled edge exactly when (p, σ, q) is a transition in T. A *path* in M is a sequence

$$(p_0, \sigma_1, p_1, \ldots, \sigma_n, p_n)$$

such that (p_{i-1}, σ_i, p_i) is a transition of M, for each $i = 1, \ldots, n$. In this case, the word $\sigma_1 \cdots \sigma_n$ is called the *label* of the path. As usual, the language $L(M)$ *accepted* by M is the set of all labels that appear in paths as above such that p_0 is the start state and p_n is a final state. These languages constitute the class of *regular* languages – see [18] for more information on regular languages.

The NFA M is called *trim* if every state of M occurs in some path from the start state to a final state. The *size* of M is $|K| + |T|$, that is the number of states plus the number of transitions in M. We note that if M is trim then $|K| \leq |T| + 1$ and, therefore, the size of M is dominated by the number of transitions in M. A state in an automaton is called a *fork state* if there at least two transitions going out of that state. A *cycle* in the NFA is a path in which the first and last states of the path are equal. The special cycle (p), where p is any state, is called trivial. A *strongly connected component* (with respect to an NFA M) is a set $\mathscr{C}$ of states that is maximal with the property that there is a path in M between any pair of states in $\mathscr{C}$. The component $\mathscr{C}$ is called *nontrivial* if there is at least one transition between two states in $\mathscr{C}$. For the sake of simplicity, we shall say that a component $\mathscr{C}$ 'contains' a transition (or a path) to mean that the NFA in which $\mathscr{C}$ exists contains that transition (or path) with all states involved belonging to $\mathscr{C}$.

2.3 The subword closure $S^{\otimes}$ and the DFA $\mathrm{Trie}(S)^{\otimes}$

As mentioned in the introductory section, any nonempty set of words of length k, for some integer $k > 0$, is called a *subword constraint*. It is used to define the language

$$S^{\otimes} = \{w \in \Sigma^* \mid \text{ if } u \text{ is a subword of } w \text{ and } |u| = k \text{ then } u \in S\}.$$

This concept was used in [11] and [4] in the context of coding for reliable DNA computing. Two properties of $S^{\otimes}$ are the following.

- If $w \in S^{\otimes}$ then every subword of w is also in $S^{\otimes}$.
- If $xu, vy \in S^{\otimes}$ and $|u| = |v| = k$ then we have that $xuvy \in S^{\otimes}$ if and only if $uv \in S^{\otimes}$.

In [11] it is shown that every language $S^{\otimes}$ is regular via the DFA $\mathrm{Trie}(S)^{\otimes}$, which is defined as follows. First, let $\mathrm{Trie}(S)$ be the *trie* accepting the set S. Recall [2] this is the complete DFA with states

$$\{[u] \mid u \text{ is a prefix of a word in } S\} \cup \{[\text{sink}]\}$$

such that $[\lambda]$ is the start state and $\{[u] \mid u \in S\}$ is the set of final states. We remark that the notation $[\cdot]$ for states is not necessary, but it is meant to help the reader distinguish easily that u represents a word and $[u]$ represents a state. By extending this notation to sets of states, we can write that the set of final states of $\mathrm{Trie}(S)$ is $[S]$. The transition function of $\mathrm{Trie}(S)$ is δ such that $\delta([u], \sigma) = [u\sigma]$, when $u\sigma$ is a prefix of S of length at most k, and with all the other values of δ being $[\text{sink}]$.

The DFA $\mathrm{Trie}(S)^{\otimes}$ accepting $S^{\otimes}$ is obtained from the tre $\mathrm{Trie}(S)$ as follows [11] – see also Fig. 1. The set of states is the same; the start state is the same; all states now are final; the transition function $\delta^{\otimes}$ of $\mathrm{Trie}(S)^{\otimes}$ is the same as δ except as follows: for each $u \in S$ and $\sigma \in \Sigma$, if $u \in \Sigma u_1$ and $u_1\sigma \in S$, then $\delta^{\otimes}([u], \sigma) = [u_1\sigma]$ – this ensures that the last k symbols read drive the automaton to a state in $[S]$. A few useful properties of $\mathrm{Trie}(S)^{\otimes}$ are the following.

- If $([u], \sigma_1, p_1, \ldots, \sigma_k, p_k)$ is a path in $\mathrm{Trie}(S)^{\otimes}$ and $p_k \neq [\text{sink}]$ then the state p_k must be $[\sigma_1 \cdots \sigma_k]$.
- If $w \in S^{\otimes}$ and $|w| \geq k$ then $\delta^{\otimes}([\lambda], w) = \delta^{\otimes}([x], w_1) = [y]$, where x is the prefix of w of length k, $w_1 = x^{-1}w$, and y is the suffix of w of length k.
- The DFA $\mathrm{Trie}(S)^{\otimes}$ can be computed in linear time with respect to the size of S – the size of a finite set of words is the sum of the lengths of the words in that set.
- Any strongly connected component $[Q]$ of $\mathrm{Trie}(S)^{\otimes}$ is such that $Q \subseteq S$.

Fig. 1 shows a part of the DFA $\mathrm{Trie}(S)^{\otimes}$ accepting the language $S^{\otimes}$, where

$$S = \{0001, 0010, 0100, 0101, 0111, 1000, 1011, 1101, 1110, 1111\}.$$

In particular the figure shows only the transitions involving states $[u]$ with $u \in S$ and the state $[\text{sink}]$.

3 Deciding Exponential Density

In this section we consider the problem of characterizing NFA languages having exponential density. Our characterization leads to a quadratic time algorithm for deciding whether the language accepted by a given NFA is exponential. We also observe that the property of exponential density is equivalent to whether the language in question is an encoding language – see the definition below. In fact our algorithm for deciding exponential density also returns an encoding part of the given language in case where indeed the language is of exponential density.

Definition 1. *A language L is called an* encoding language *if there are words x, y and, for every integer $n \geq 2$, there is a code C_n of cardinality n such that*

$$xC_n^*y \subseteq L.$$

In this case, the code C_n is called an encoding part *of L.*

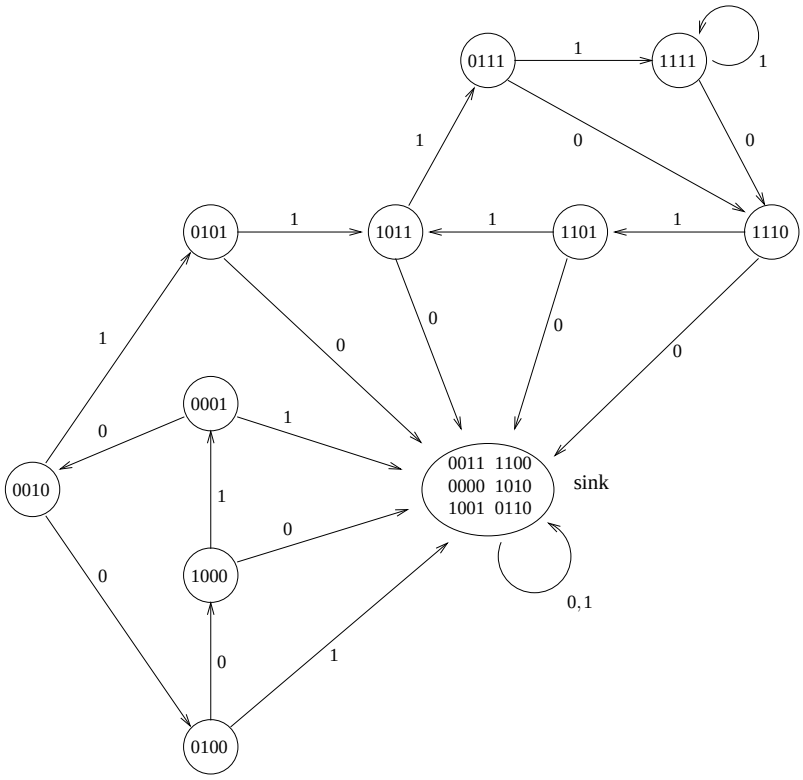

Fig. 1. The part of Trie$(S)^{\otimes}$ involving only states $[u]$ with $u \in S$ and the state [sink].

When we need to encode into L arbitrary sequences made from a certain set $\{t_1,\ldots,t_n\}$ of objects (e.g., data blocks in Σ^m for some alphabet Σ and integer $m \geq 1$), and L is an encoding language then we can pick a code $C_n = \{u_1,\ldots,u_n\}$ and words x,y such that any sequence $(t_{i_1},\ldots,t_{i_r})$ can be encoded with the word $xu_{i_1}\cdots u_{i_r}y \in L$. The use of the fixed words x,y is necessary in general, as C_n^* itself might not be a subset of L – this is not an issue in the case of $L = S^{\otimes}$ because then any subword of $S^{\otimes}$ is an element of $S^{\otimes}$ as well.

Now we consider the following problem.

(P0) Given a regular language L, decide whether L is of exponential density.

It is not difficult to see – see Lemma 1 below – that this problem is equivalent to whether L is an encoding language.

In [14] (see also [15]) the author gives a very simple criterion for testing this property for a regular language L: it has exponential density if and only if any trim deterministic automaton accepting L has a state that belongs to two cycles in the graph corresponding to the automaton. Here we consider the case where the language is given via a nondeterministic automaton. We show that L has an exponential density if and only if any trim nondeterministic automaton accepting L has a strongly connected component containing two paths of the same length, starting at the same state, and whose labels are different. Moreover, we show that our test can be performed in quadratic time. Note that, when the automaton is deterministic, our test is equivalent to whether a strongly connected component contains a fork state (this is of course equivalent to the test of [14]) and can be performed in linear time. The rest of this section deals with the formalities of the above statements.

Lemma 1. *Let L be a regular language. The following statements are equivalent.*

ENC: L is an encoding language.

EXP: L has exponential density.
BL2: There is a two-element block code C and words x,y such that $xC^*y \subseteq L$.
SCC: For every trim NFA A accepting L, there exists a strongly connected component in A containing two paths of the same length, starting at the same state, and whose labels are different.

Proof. First we establish the equivalence ENC $\leftrightarrow$ BL2. For the 'if' part, consider any integer $n \geq 2$ and let $m = \lceil \log n \rceil$. Then, as $x(C^m)^*y \subseteq L$ and C^m contains at least n codewords, it follows that L is an encoding language. For the 'only if' part, let $C_2 = \{v,w\}$ be a two-element code and x,y be words such that $xC_2^*y \subseteq L$. As C_2 is a code we have that C_2^2 is also a code consisting of four codewords. The required block code C is $\{vw,wv\}$.

It is sufficient now to prove the following sequence of statements: BL2 $\rightarrow$ EXP, SCC $\rightarrow$ BL2, EXP $\rightarrow$ SCC.

Part BL2 $\rightarrow$ EXP: Let $C = \{z_1,z_2\}$ and consider, for every $n \geq 0$, all words in L of length $|x|+|y|+\ell n$, where ℓ is the length of z_1 and z_2. As $xC^ny \subseteq L$, there are at least 2^n such words and, therefore, L must have exponential density.

Part SCC $\rightarrow$ BL2: Assume there is a state q in some strongly connected component $\mathscr{C}$, and two paths in $\mathscr{C}$ starting at q, ending at some states q_1 and q_2, and having two different labels u_1,u_2 of the same length. Then, there must be two paths in $\mathscr{C}$, one from q_1 to q and the second from q_2 to q with some labels v_1,v_2, respectively. Then there are two cycles in $\mathscr{C}$ with labels u_1v_1 and u_2v_2. Moreover, it follows that there are two cycles in $\mathscr{C}$ with labels $z_1 = u_1v_1u_2v_2$ and $z_2 = u_2v_2u_1v_1$, which are different and of the same length, say ℓ. As the NFA A is trim, there are two paths, one from the start state to q with some label x, and the other from q to a final state with some label y. Let $C = \{z_1,z_2\}$. Then it follows that

$$xC^*y \subseteq L.$$

Part EXP $\rightarrow$ SCC: We use contraposition by assuming the negation of SCC and showing that the density of $L(A)$ is polynomially upper-bounded. So assume that in every strongly connected component of A, any two paths starting at the same state and having the same length must have equal labels. This implies that there is no state having two outgoing transitions with different labels. First we have the following claim.

Claim 1: The assumption implies that, in every strongly connected component $\mathscr{C}$, for every $n \geq 0$, there are at most $|\mathscr{C}|$ distinct path labels of length n.

To see this, we first note that the claim is obvious if $\mathscr{C}$ is trivial. If $\mathscr{C}$ is nontrivial, then for every state q in $\mathscr{C}$ and any integer $n \geq 0$, there is at least one path in $\mathscr{C}$ of length n starting at state q. At the same time, the assumption implies that, for every state q in $\mathscr{C}$ and integer $n \geq 0$, there is at most one path label of length n. Thus, for every state q in $\mathscr{C}$ and any $n \geq 0$, there is exactly one path label of length n starting at q, and the claim follows easily from this observation.

Next we show that the density of $L(A)$ is polynomially upper-bounded, using induction on the number k, say, of strongly connected components in A. For $k = 1$ this follows immediately from Claim 1. Assume the statement holds when A has at most t components, for some $t \geq 1$, and consider the case where A has $k = t+1$ components. As $k \geq 2$, there must be a strongly connected component $\mathscr{D}$ with no outgoing transitions – also, as A is trim, $\mathscr{D}$ cannot contain the start state. Consider the set L_n of all words of length n accepted by A. Then

$$L_n = M_n \cup K_n,$$

where M_n is the set of words of length n accepted using paths containing no state in $\mathscr{D}$, and K_n is the set of words of length n accepted using paths ending in $\mathscr{D}$. Let $\overline{\mathscr{D}}$ be the set of states not in $\mathscr{D}$, and let $q_1,\ldots,q_r$ be all states in $\overline{\mathscr{D}}$ having transitions going into $\mathscr{D}$. Let $N_1,\ldots,N_r$ be the languages accepted by the part of A that involves no states from $\mathscr{D}$ and has as final states $\{q_1\},\ldots,\{q_r\}$, respectively. Let $N_1',\ldots,N_r'$ be the

languages accepted starting, respectively, from the states $q_1,\ldots,q_r$ and then using only states in $\mathscr{D}$, where the final states of A that are in $\mathscr{D}$ are used as final states. Then, it follows that

$$K_n = (N_1N_1' \cup \cdots \cup N_rN_r') \cap \Sigma^n$$

and then

$$L_n = M_n \cup (N_1N_1' \cap \Sigma^n) \cup \cdots \cup (N_rN_r' \cap \Sigma^n).$$

By the induction hypothesis, $|M_n| = O(n^c)$, for some constant c. Now for each term $N_iN_i' \cap \Sigma^n$ we have

$$N_iN_i' \cap \Sigma^n = \bigcup_{j=0}^{n} (N_i \cap \Sigma^j)(N_i' \cap \Sigma^{n-j}).$$

Again, as N_i is accepted by an NFA having at most t components we have that $|N_i \cap \Sigma^j| = O(j^{c_i})$, for some constant c_i. Also, by Claim 1, $|N_i' \cap \Sigma^{n-j}| = O(1)$. With these observations, it follows that

$$|N_iN_i' \cap \Sigma^n| = O((n+1) \times n^{c_i}) = O(n^{1+c_i}) \text{ and } |L_n| = O(n^{\max(c,1+c_1,\ldots,1+c_r)})$$

and, therefore, $L(A)$ is polynomially upper-bounded. □

Theorem 1. *The following problem is decidable in quadratic time.*

Input: NFA A
Output: YES/NO, depending on whether the density of $L(A)$ is exponential.

Moreover, there is a (quadratic time) algorithm that returns an encoding part of $L(A)$ of cardinality n, for any given NFA A and integer $n \geq 2$, if $L(A)$ is indeed an encoding language.

Proof. We shall use the previous lemma and the well-known product construction for labeled graphs. In particular, for any directed labeled graph $G = (V,E)$, the graph G^2 has vertices all pairs in $V \times V$ and edges all triples of the form $((p_1,p_2),(a_1,a_2),(q_1,q_2))$ such that (p_1,a_1,q_1) and (p_2,a_2,q_2) are edges in E. It is evident that for any path in G^2 there are two corresponding paths in G of the same length and, conversely, for any two paths in G of the same length there is a corresponding path in G^2. For a path P in G^2, the first (resp. second) corresponding path is simply the sequence of edges formed by the first (resp. second) components in the sequence of edges in P. The required decision algorithm is as follows.

```
AED(A)
 1  Make the NFA A trim;
 2  Compute the (strongly connected) components of A;
 3  FOUND = false;
 4  for each component G and while not FOUND
 5     do
 6        Compute G^2;
 7        Compute the set Q_1 of vertices (p_1,p_2) in G^2 such that
 8        – there is an edge ((p_1,p_2),(a_1,a_2),(q_1,q_2)) with a_1 ≠ a_2;
 9        Compute the set Q_2 of vertices in G^2 of the form (t,t);
10        if (there is a path from Q_2 to Q_1) then FOUND = true;
11  if (FOUND) output YES else output NO;
```

For the correctness of the algorithm we note that, at the last step, FOUND is true if and only if condition SCC of Lemma 1 is true. Indeed, if there is a path in G^2 from some (t,t) to some $(p_1,p_2) \in Q_1$ then there is also

a path from (t,t) to some (q_1,q_2) where the last edge in the path is $((p_1,p_2),(a_1,a_2),(q_1,q_2))$ with $a_1 \neq a_2$; hence, there must be two equal length paths in G starting at t and having different labels. Conversely, if there are two paths in G of the same length, starting at some state t and having different labels, then there are also two such paths differing on their last symbols, which implies that the algorithm will set FOUND to true when it processes the component G.

For the time complexity of the algorithm, first we note that Step 1 can be performed in linear time and then Step 2 also in linear time [5]. Now let n be the size of A, let k be the number of strongly connected components in the trimmed A, and let n_i be the size of the component i. Then $n_1 + \cdots + n_k = O(n)$. The i-th iteration of the loop requires time $O(n_i^2)$ to construct the product of the i-th component, which is of size $O(n_i^2)$, and then the next two steps are linear with respect to n_i^2. Also linear is the last step in the loop via a breadth-first search algorithm. So in the worst case the algorithm requires time $O(n_1^2 + \cdots + n_k^2)$.

Now regarding the second part of the theorem, once we have decided that $L(A)$ is indeed an encoding language, we proceed as follows. Let

$$P = (t,t),(b_1,b_1'),(t_1,t_1'),\ldots,(b_m,b_m'),(t_m,t_m')$$

be the path found in the current component of the above loop. Then there is an edge $((t_m,t_m'),(a,a'),(q,q'))$ with $a \neq a'$ in that component of A. Compute shortest paths in A from t_m to t and from t_m' to t. These would have some labels w, w' respectively. Let $v = b_1 \cdots b_m a$ and $v' = b_1' \cdots b_m' a'$. Then, we have two equal length cycles from t to t with distinct labels $z = vwv'w'$ and $z' = v'w'vw$. Let $K = \{z,z'\}$ and $j = \lceil n/2 \rceil$. Obviously K^j contains at least n words and, therefore, any subset of K^j of n words is an encoding part of $L(A)$ as required. Finally, we note that the extra steps to compute the encoding part of $L(A)$ do not increase the asymptotic complexity of the algorithm. □

4 Characterizing maximal D's with $D^* \subseteq S^{\otimes}$

In this section we fix an arbitrary subword constraint S of some length k, that is, a nonempty set S consisting of words of fixed length $k > 0$. The first main problem is to characterize structurally any nonempty language D whose words are of length at least k and is maximal with the property "$D^* \subseteq S^{\otimes}$". It turns out (see Theorem 2) that, for such a D, there is a nontrivial strongly connected component $[Q]$ of $\mathrm{Trie}(S)^{\otimes}$, with $Q \subseteq S$, such that for all D-words, their first k symbols drive the automaton $\mathrm{Trie}(S)^{\otimes}$ to a certain set of states $[X] \subseteq [Q]$, the D-words get accepted at a set of states $[Y] \subseteq [Q]$ that depends on X and is denoted as $[Y] = [Q_X^{>}]$, and Y has the property that, from any state $[y] \in [Y]$ and on input x, the automaton $\mathrm{Trie}(S)^{\otimes}$ gets to the state $[x]$, for any $x \in X$ – see Definition 2 and Fig. 2.

The second main (and related) problem is to characterize structurally any nontrivial language D whose words are of length at least k and is maximal with the property "$D^* \subseteq S^{\otimes}$". The characterization is the same as the one for a nonempty D with the additional requirement that the strongly connected component $[Q]$ contains a fork state – see Theorem 3. As discussed earlier, the fact that D is nontrivial implies that D^* is of exponential density and allows one to encode in D^* – therefore also in $S^{\otimes}$ – arbitrary data.

Definition 2. *Let Q be a nonempty subset of S such that $[Q]$ is a nontrivial strongly connected component of* $\mathrm{Trie}(S)^{\otimes}$. *For any nonempty subset X of Q, we define*

$$Q_X^{>} \triangleq \{v \in Q \mid \forall x \in X : \delta^{\otimes}([v],x) = [x]\} = \{v \in Q \mid \forall x \in X : vx \in S^{\otimes}\}.$$

Example 1. In Fig. 1, let $Q = \{0111, 1111, 1110, 1101, 1011\}$. Then, $Q_{\{1011\}}^{>} = \{1011, 0111, 1111\}$ and, for $X = \{0111, 1011\}$, we have that

$$Q_X^{>} = \{0111, 1111\}.$$

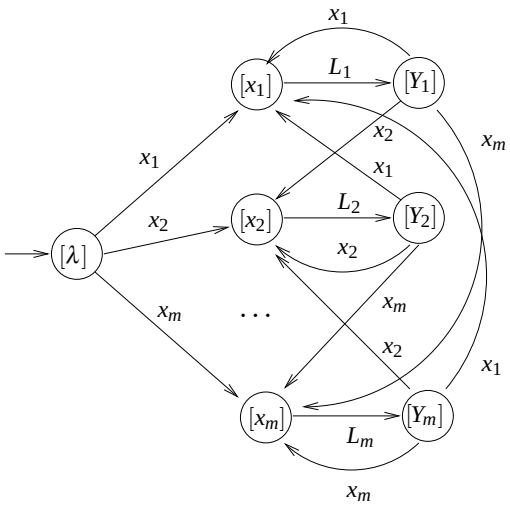

Fig. 2. The sets of states $[X] = \{[x_1], \ldots, [x_m]\}$ and $[Y] = [Q_X^{>}] = [Y_1] \cup \cdots \cup [Y_m]$ are subsets of some strongly connected component $[Q]$ of $\mathrm{Trie}(S)^{\otimes}$. Each node $[Y_i]$ in the diagram represents one or more states in $[Y]$, and each language L_i consists of the labels of all paths from $[x_i]$ to $[Y]$. Then, the language $\langle Q,X\rangle$ is equal to $\cup_{i=1}^{m} x_i L_i$.

The major structural observation for the desired languages D is that they can be expressed in terms of the sets

$$\langle Q,X\rangle_x \triangleq \{w \in \Sigma^* \mid \delta^{\otimes}([x],w) \in [Q_X^{>}]\}$$

for all $x \in X$, where $[Q]$ is a strongly connected component of $\mathrm{Trie}(S)^{\otimes}$ – thus, $Q \subseteq X$ – and $X \subseteq Q$. Let

$$\langle Q,X\rangle \triangleq \bigcup_{x\in X} x\langle Q,X\rangle_x.$$

Obviously, the words of this language are of length at least k. This notation is also important in the section on algorithmic considerations.

Example 2. In Fig. 1, for $Q = \{0111, 1111, 1110, 1101, 1011\}$ and $X = \{0111, 1011\}$, we have that

$$\langle Q,X\rangle_{1011} = 1(\lambda + 11^*)(0111(\lambda + 11^*))^* \text{ and } \langle Q,X\rangle_{0111} = (\lambda + 11^*)(0111(\lambda + 11^*))^*,$$

where we have used notation of regular expressions for denoting langauges.

Theorem 2. *Let S be any subword constraint of some length k, and let D be any nonempty language whose words are of length at least k. Then, D is maximal with $D^* \subseteq S^{\otimes}$ if and only if there are nonempty subsets X, Y, Q of S such that*

$$D = \langle Q,X\rangle = S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Y,$$

and $X, Y \subseteq Q$, $[Q]$ is a nontrivial strongly connected component of $\mathrm{Trie}(S)^{\otimes}$, $Y = Q_X^{>}$, and X is maximal with "$X \subseteq Q$ and $Q_X^{>} = Y$".

Theorem 3. *Let S be any subword constraint of some length k, and let D be any nontrivial language whose words are of length at least k. Then, D is maximal with $D^* \subseteq S^{\otimes}$ if and only if there are nonempty subsets X, Y, Q of S such that*

$$D = \langle Q,X\rangle = S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Y,$$

and $X, Y \subseteq Q$, $[Q]$ is a nontrivial strongly connected component of $\mathrm{Trie}(S)^{\otimes}$, containing a fork state, $Y = Q_X^{>}$, and X is maximal with "$X \subseteq Q$ and $Q_X^{>} = Y$".

The rest of the section deals with the proofs of the above results, which rely on a sequence of technical lemmata. The first observation gives a taste of what it means when $D^* \subseteq S^{\otimes}$, without necessarily requiring that D is maximal with this property.

Lemma 2. *Let D be a nonempty language whose words are of length at least k.*

1. *If $D^* \subseteq S^{\otimes}$ then $D = \bigcup_{x \in X} x(x^{-1}D)$ and $x(x^{-1}D)y \subseteq S^{\otimes}$, for all $x, y \in X$, where X is the set of all prefixes of D of length k.*
2. *If there is a subset X of S and languages D_x, for all $x \in X$, such that $D = \bigcup_{x \in X}(xD_x)$ and $xD_xy \subseteq S^{\otimes}$, for all $x, y \in X$, then $D^* \subseteq S^{\otimes}$.*

Proof. For the first statement, the part $D = \bigcup_{x \in X} x(x^{-1}D)$ can be shown without complications using standard set theoretic reasoning. Now take any $x, y \in X$ and $u \in x^{-1}D$. Then, $xu \in D$. Also, there exists a word of the form $yv \in D$. As $D^2 \subseteq S^{\otimes}$, it follows that $xuyv \in S^{\otimes}$. Hence, also $xuy \in S^{\otimes}$, as required.

For the second statement, we show $D^n \subseteq S^{\otimes}$ using induction on n. Obviously, $D^0 \subseteq S^{\otimes}$. For any $n \geq 1$ assume that $D^{n-1} \subseteq S^{\otimes}$ and consider any word $w \in D^n$. By the premises about D, w can be written as xuv, with $x \in X$, $u \in D_x$, and $v \in D^{n-1}$. Note here that, as $xux \in S^{\otimes}$, we have that $xu \in S^{\otimes}$. If $n = 1$ then v is empty and, in this case, $w = xu \in S^{\otimes}$, as required. If $n > 1$, then $v = yv_1$ for some $y \in X$ and $v_1 \in \Sigma^*$. In this case, as $xu, xuy, yv_1 \in S^{\otimes}$, it follows that $w = xuyv_1 \in S^{\otimes}$, as required. □

Lemma 3. *Let X, Q be nonempty subsets of S such that $X \subseteq Q$ and $[Q]$ is a nontrivial strongly connected component of* $\mathrm{Trie}(S)^{\otimes}$.

1. $\langle Q, X \rangle^* \subseteq S^{\otimes}$.
2. $\langle Q, X \rangle = S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Q_X^{>}$.
3. *If $Q_X^{>} \neq \emptyset$ then, for all $x \in X$, we have that $\langle Q, X \rangle_x \neq \emptyset$.*

Proof. For the first statement, it is sufficient to show that $x\langle Q, X \rangle_x y \subseteq S^{\otimes}$ – see Lemma 2. So let $w \in \langle Q, X \rangle_x$. Then, $\delta^{\otimes}([x], w) \in [Q_X^{>}]$ and, therefore, $\delta^{\otimes}([\lambda], xw) = [v]$ for some $v \in Q_X^{>}$. As $y \in X$, we also have $\delta^{\otimes}([v], y) \neq [\mathrm{sink}]$. Hence, $\delta^{\otimes}([\lambda], xwy) \neq [\mathrm{sink}]$ and, therefore, $xwy \in S^{\otimes}$, as required.

For the second statement, let $D = S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Q_X^{>}$. For the direction $D \subseteq \langle Q, X \rangle$, note that every word $w \in D$ is of the form $xw_1 = w_2 y$, with $x \in X$ and $y \in Q_X^{>}$. Then, as $w_2 y \in S^{\otimes}$, we have $\delta^{\otimes}([\lambda], w_2 y) = [y] \Rightarrow \delta^{\otimes}([\lambda], xw_1) \in [Q_X^{>}] \Rightarrow w_1 \in \langle Q, X \rangle_x \Rightarrow w \in \langle Q, X \rangle$. Conversely, consider any word $w \in \langle Q, X \rangle$. By the first statement of this lemma, $w \in S^{\otimes}$. Also, $w = xw_1$, for some $x \in X$ and $w_1 \in \langle Q, X \rangle_x$. Hence, $w \in X\Sigma^*$. Moreover, $\delta^{\otimes}([x], w_1) = [y]$, for some $y \in Q_X^{>}$. This implies that xw_1 must end with the word y. Hence, $w \in \Sigma^* Q_X^{>}$.

For the third statement, take any $x \in X$. As $Q_X^{>} \neq \emptyset$, we can pick any $v \in Q_X^{>}$. As both $x, v \in Q$, there is a path from $[x]$ to $[v]$ having some label $w \in \Sigma^*$. This implies that $w \in \langle Q, X \rangle_x$. □

Now we present the next major step to proving the desired theorem. We establish that any D with $D^* \subseteq S^{\otimes}$ must be a subset of some language of the form $\langle Q, X \rangle$. This result allows us to focus on a strongly connected component of $\mathrm{Trie}(S)^{\otimes}$ when we wish to characterize structurally those D's.

Lemma 4. *If D is a nonempty language whose words are of length at least k and $D^* \subseteq S^{\otimes}$, then*

$$D \subseteq \langle Q, X \rangle,$$

for some nonempty subsets Q, X of S with $X \subseteq Q$ and $[Q]$ a nontrivial strongly connected component of $\mathrm{Trie}(S)^{\otimes}$.

Proof. First note that, by Lemma 2, we can write

$$D = \bigcup_{x \in X} x(x^{-1}D),$$

such that X is the (nonempty) set of prefixes of length k of D and $x(x^{-1}D)y \subseteq S^{\otimes}$, for all $x,y \in X$. The rest of the proof consists of two parts, whose conjunction establishes the truth of the lemma. In the first part, we show that $X \subseteq Q$ such that $[Q]$ is a nontrivial strongly connected component of $\mathrm{Trie}(S)^{\otimes}$, and in the second part we show that $x^{-1}D \subseteq \langle Q,X\rangle_x$, for all $x \in X$.

Part 1. Consider any (possibly equal) $x,y \in X$. Then, there are words $u_1 \in x^{-1}D$ and $u_2 \in y^{-1}D$ such that $xu_1y \in S^{\otimes}$ and $yu_2x \in S^{\otimes}$. We need to show that there are (nonempty) paths from $[x]$ to $[y]$, and from $[y]$ to $[x]$. This, however, follows easily from the definition of $\mathrm{Trie}(S)^{\otimes}$ and the fact that there are paths in this automaton with nonempty labels xu_1y and yu_2x.

Part 2. Consider any word $w \in x^{-1}D$. Then, $xw \in D$. We need to show that $\delta^{\otimes}([x],w) \in [Q_X^{>}]$. For this, it is sufficient to show that $\delta^{\otimes}([x],w) = [v]$ for some $v \in Q$ such that $\delta^{\otimes}([v],y) = [y]$, for all $y \in X$. Indeed, as $xw \in S^{\otimes}$, $\delta^{\otimes}([\lambda],xw) = [v]$ for some $v \in S$, so $\delta^{\otimes}([x],w) = [v]$ and, therefore, there is a path from $[x]$ to $[v]$. Now, as $xwx \in S^{\otimes}$, we have that $\delta^{\otimes}([v],x) = [x]$. Thus, there is also a path from $[v]$ to $[x]$ and, therefore $v \in Q$. Now take any $y \in X$. As $xwy \in S^{\otimes}$, we have that $\delta^{\otimes}([\lambda],xwy) \neq [\mathrm{sink}]$, which implies that $\delta^{\otimes}([v],y) = [y]$, as required. □

The last technical lemma before the proof of Theorem 2 deals with containment relationships between sets of the form $Q_X^{>}$ and $P_Z^{>}$.

Lemma 5. *Let X,Z,Q,P be nonempty subsets of S such that $X \subseteq Q$, $Z \subseteq P$, $[Q]$ and $[P]$ are nontrivial strongly connected components of* $\mathrm{Trie}(S)^{\otimes}$*, and $Q_X^{>} \neq \emptyset$.*

1. *If $\langle Q,X\rangle \subseteq \langle P,Z\rangle$ then $X \subseteq Z$.*
2. *If $Z \subseteq Q$ and $\langle Q,X\rangle = \langle Q,Z\rangle$ then $X = Z$.*

Proof. For the first statement, take any $x \in X$. As $Q_X^{>} \neq \emptyset$, there is a word $w \in \langle Q,X\rangle_x$ – see Lemma 3. So $xw \in \langle Q,X\rangle$ and, therefore, $xw \in \langle P,Z\rangle$. Then, $xw \in z\langle P,Z\rangle_z$, for some $z \in Z$, which implies that $z = x$. Hence, $x \in Z$.

The second statement follows logically by applying the first statement twice. □

Proof of *Theorem 2*. First we do the 'only if' part. So suppose that D is maximal with $D^* \subseteq S^{\otimes}$. Then, $D \subseteq \langle Q,X\rangle$ according to Lemma 4. At the same time, Lemma 3 says that $\langle Q,X\rangle^* \subseteq S^{\otimes}$. As D is maximal we have that, in fact, $D = \langle Q,X\rangle$. Let $Y = Q_X^{>}$. By Lemma 3, we have

$$D = S^{\otimes} \cap X\Sigma^* \cap \Sigma^*Y.$$

As D is nonempty, we have that Y is nonempty as well.

It remains to show that X is maximal with "$X \subseteq Q$ and $Q_X^{>} = Y$". So suppose that $X \subseteq Z \subseteq Q$ and $Q_Z^{>} = Y$. Then, we need to show that $Z = X$. For this it suffices to show that $\langle Q,X\rangle = \langle Q,Z\rangle$. In turn, this would follow by the maximality of D if we show that $D \subseteq \langle Q,Z\rangle$. So take any $w \in D = \langle Q,X\rangle$. Then, $w = xw_1$ for some $x \in X$ and $w_1 \in \langle Q,X\rangle_x$, which implies $\delta^{\otimes}([x],w_1) \in [Q_X^{>}] = [Q_Z^{>}]$. Also, as $x \in Z$ we have that $w_1 \in \langle Q,Z\rangle_x$ and, therefore $xw_1 \in \langle Q,Z\rangle$, as required.

Now we do the 'if' part. By Lemma 3, we have that $D^* \subseteq S^{\otimes}$. To show that D is maximal, we assume that $D \subseteq B$ and $B^* \subseteq S^{\otimes}$, for some language B, and we deduce that $B = D$. By Lemma 4, we have that $B \subseteq \langle P,Z\rangle$, where Z,P are nonempty subsets of S, $Z \subseteq P$, and $[P]$ is a nontrivial strongly connected component of $\mathrm{Trie}(S)^{\otimes}$. This implies that $\langle Q,X\rangle \subseteq \langle P,Z\rangle$. It suffices to show that $P = Q$ and $X = Z$. By Lemma 5, we get $X \subseteq Z$, so there is a state belonging to both $[Q]$ and $[P]$. This implies $P = Q$. Also, obviously $Q_Z^{>} = P_Z^{>}$. As X is maximal with "$X \subseteq Q$ and $Q_X^{>} = Y$" and $X \subseteq Z \subseteq Q$, it suffices to show that $Q_Z^{>} = Q_X^{>}$.

First, by definition of $Q_X^>$, $X \subseteq Z$ implies $Q_Z^> \subseteq Q_X^>$. For the converse inclusion, take any $v \in Q_X^>$. Also, take any $x \in X$. As $x, v \in Q$, there is a path from $[x]$ to $[v]$ having some label w, which implies $w \in \langle Q, X \rangle_x$. As $\delta^\otimes([\lambda], xw) = [v]$, there is a word w' such that $xw = w'v$. So $w'v \in \langle Q, X \rangle$ and, therefore, $w'v \in \langle Q, Z \rangle$. Then, by Lemma 3, we have $w'v \in \Sigma^* Q_Z^>$ and, therefore, $v \in Q_Z^>$, as required. □

The next two lemmata are required for the proof of Theorem 3, which involves a fork state in the strongly connected component $[Q]$.

Lemma 6. *If $[Q]$ is a nontrivial strongly connected component of* $\mathrm{Trie}(S)^\otimes$ *then, for every $v \in Q$ and $n \geq 1$, there is $u \in Q$ and a path of length n from $[u]$ to $[v]$.*

Proof. As $[Q]$ is nontrivial, there is a path of some length $\ell \geq 1$ from $[v]$ to $[v]$. Obviously this path can be iterated arbitrarily many times to obtain a long path of length at least n from $[v]$ to $[v]$. Then, there has to be some state $[u]$ in that long path as required. □

Lemma 7. *Let $[Q]$ be a nontrivial strongly connected component of* $\mathrm{Trie}(S)^\otimes$.

1. *If $[Q]$ contains a fork state $[v]$ having transitions to some distinct states $[x_1], [x_2] \in [Q]$, then $\langle Q, X \rangle$ is a nontrivial language, where $X = \{x_1, x_2\}$.*
2. *There is a subset X of Q such that $\langle Q, X \rangle$ is nontrivial if and only if $[Q]$ contains a fork state.*

Proof. For the first statement, the premise implies that there are distinct symbols $\sigma_1, \sigma_2 \in \Sigma$ such that $\delta^\otimes([v], \sigma_1) = [x_1]$ and $\delta^\otimes([v], \sigma_2) = [x_2]$. Also, by Lemma 6, there is a state $[u]$ and a word x of length $k-1$ such that $\delta^\otimes([u], x) = [v]$. This implies that $x_1 = x\sigma_1$ and $x_2 = x\sigma_2$, and, therefore, $u \in Q_X^>$. Then, the sets $\langle Q, X \rangle_{x_1}$ and $\langle Q, X \rangle_{x_2}$ are nonempty, which implies that there are words w_1, w_2 such that $x_1 w_1, x_2, w_2 \in \langle Q, X \rangle$. Finally, as $x_1 w_1 x_2 w_2 \neq x_2 w_2 x_1 w_1$, the language $\langle Q, X \rangle$ is nontrivial.

For the second statement, the 'if' part is simply a simpler version of the first statement. So we consider the 'only if' part. As $\langle Q, X \rangle$ is nontrivial, the set $Q_X^>$ is nonempty. So let $v \in Q_X^>$. If X has at least two distinct elements x_1, x_2 then, as there are paths from $[v]$ to $[x_1]$ and $[x_2]$, it follows that a fork state must exist in these paths, as required. Now suppose $X = \{x\}$, but assume for the sake of contradiction that $[Q]$ contains no fork state. Then, obviously, the component $[Q]$ is a single cycle and, therefore, $Q_X^>$ consists of exactly one element, say v. There is also exactly one simple path from $[x]$ to $[v]$ having some label w. Then, it is easy to see that all paths from $[x]$ to $[v]$ have labels in $w(xw)^*$, which contradicts the premise that $\langle Q, X \rangle$ is nontrivial. □

Proof of *Theorem 3*. The 'if' part is simply a weaker form of the 'if' part in Theorem 2. For the 'only if' part, we first apply Theorem 2: there are nonempty subsets X, Y, Q of S such that

$$D = \langle Q, X \rangle = S^\otimes \cap X\Sigma^* \cap \Sigma^* Y,$$

and $X, Y \subseteq Q$, $[Q]$ is a nontrivial strongly connected component of $\mathrm{Trie}(S)^\otimes$, $Y = Q_X^>$, and X is maximal with "$X \subseteq Q$ and $Q_X^> = Y$". It remains to show that $[Q]$ contains a fork state. But this follows immediately from Lemma 7. □

5 Connection with a previous method

In [3] and [4], in the context of encoding data into $S^\otimes$, the authors consider the problem of constructing a nonempty set B such that $B^* \subseteq S^\otimes$, using the following method.

1. Pick any nonempty subset Y of S.
2. Let $S_Y = \{v \in S \mid \forall y \in Y : yv \in S^\otimes\}$.

3. Let $B_Y = S^{\otimes} \cap S_Y \Sigma^* \cap \Sigma^* Y$.

In that method, S_Y is the set of possible S-words that can be appended to any Y-word without violating the constraint S. As expected, it can be shown that $B_Y^* \subseteq S^{\otimes}$. However, if we use a bad choice for Y then S_Y could be empty. Here we can evaluate the above method in view of the tools developed in the previous section. It is clear that the set Y should be a subset of some Q such that $[Q]$ is a strongly connected component of $\mathrm{Trie}(S)^{\otimes}$. We define the following analogue of $Q_X^{>}$:

$$Q_Y^{<} \triangleq \{v \in Q \mid \forall y \in Y : \delta^{\otimes}([y], v) = [v]\} = \{v \in Q \mid \forall y \in Y : yv \in S^{\otimes}\}.$$

Then, the set B_Y constructed above can be written as

$$B_Y = S^{\otimes} \cap Q_Y^{<} \Sigma^* \cap \Sigma^* Y.$$

As expected there is a strong connection between B_Y's and sets of the form $\langle Q, X \rangle$, where $X = Q_Y^{<}$.

Lemma 8. *Let X, Y, Q be nonempty subsets of S such that $X, Y \subseteq Q$ and $[Q]$ is a nontrivial strongly connected component of* $\mathrm{Trie}(S)^{\otimes}$.

1. *If $Q_X^{>} = Y$ then $X \subseteq Q_Y^{<}$.*
2. *If $Q_Y^{<} = X$ then $Y \subseteq Q_X^{>}$.*
3. *If X is maximal with "$X \subseteq Q$ and $Q_X^{>} = Y$" then $X = Q_Y^{<}$.*
4. *If Y is maximal with "$Y \subseteq Q$ and $Q_Y^{<} = X$" then $Y = Q_X^{>}$.*
5. *X is maximal with "$X \subseteq Q$ and $Q_X^{>} = Y$" if and only if Y is maximal with "$Y \subseteq Q$ and $Q_Y^{<} = X$".*

Proof. For the first statement, take any $x \in X$. We want $yx \in S^{\otimes}$ for all $y \in Y$. So for any $y \in Y$ we have that also $y \in Q_X^{>}$, which implies $y \in Q$ and $yx \in S^{\otimes}$, as required. The second statement follows by symmetry.

For the third statement, we only need to show the direction $Q_Y^{<} \subseteq X$. So take any $x \in Q_Y^{<}$, and let $X' = X \cup \{x\}$. As $x \in Q$, we have that $X' \subseteq Q$. We shall show that $Q_X^{>} = Q_{X'}^{>}$ because then, by the maximality of X, we get that $X' = X$ and, therefore, $x \in X$, as required. As $X \subseteq X'$, we have $Q_{X'}^{>} \subseteq Q_X^{>}$. For the converse containment, take any $v \in Q_X^{>}$. Then, as $v \in Y$, $vx \in S^{\otimes}$. Also $vy \in S^{\otimes}$, for all $y \in X$. Hence, $vx' \in S^{\otimes}$ for all $x' \in X'$ and, therefore, $v \in Q_{X'}^{>}$, as required.

The fourth statement follows from the third one by symmetry. For the last statement, we only do the '<u>if</u>' part, as the converse would follow again by symmetry. So assume Y is maximal. Then we have $Y = Q_X^{>}$. We shall show that X is maximal by taking any $v \in Q$ such that the set $X' = X \cup \{v\}$ satisfies $Q_{X'}^{>} = Y$, and then proving that $X' = X$. Obviously it suffices to prove that $v \in X$, or that $v \in Q_Y^{<}$. So take any $y \in Y$. As $y \in Q_{X'}^{>}$, we have $yv \in S^{\otimes}$. Hence, $v \in Q_Y^{<}$, as required. □

Thus, in the method of [3] and [4] with the requirement that $Y \subseteq Q$, the constructed set $B_Y = S^{\otimes} \cap Q_Y^{<} \Sigma^* \cap \Sigma^* Y$ has the following properties.

- As $Y \subseteq Q_X^{>}$, we have $B_Y \subseteq S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Q_X^{>} = \langle Q, X \rangle$, where $X = Q_Y^{<}$.
- If Y is chosen to be maximal with "$Y \subseteq Q$ and $Q_Y^{<} = X$" then $Y = Q_X^{>}$,

$$B_Y = S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Q_X^{>}$$

and, X is maximal with "$X \subseteq Q$ and $Q_X^{>} = Y$". Therefore, B_Y is maximal with $B_Y^* \subseteq S^{\otimes}$.

6 Algorithmic Considerations for maximal D's with $D^* \subseteq S^{\otimes}$

In this section we consider algorithms for the following problems.

(P1) Given a subword constraint S, compute a nonempty language D that is maximal with $D^* \subseteq S^{\otimes}$.
(P2) Given a subword constraint S, compute a nontrivial language D that is maximal with $D^* \subseteq S^{\otimes}$.

We deal with the above problems by considering the following subproblems, which refer to a given $\mathrm{Trie}(S)^{\otimes}$ and a given nontrivial strongly connected component $[Q]$ of $\mathrm{Trie}(S)^{\otimes}$.

(SP1) Given a nonempty subset X of Q, compute the set $Q_X^{>}$.
(SP2) Given nonempty subsets Z,Y of Q such that $Q_Z^{>} = Y$, compute X such that $Z \subseteq X$ and X is maximal with "$X \subseteq Q$ and $Q_X^{>} = Y$".
(SP3) Given nonempty subsets X,Y of Q compute a deterministic automaton accepting $S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Y$.

We first consider algorithms for these subproblems. *We use the abbreviation T for* $\mathrm{Trie}(S)^{\otimes}$. We use a bijective encoding from the words in Q onto the set $\overline{Q} = \{0,1,\ldots,|Q|-1\}$, such that, for $v \in Q$, $\overline{v}$ is the code of v in $\overline{Q}$. Using hashing techniques the encoding and decoding functions can be done in time $O(1)$. As customary, we realize any subset $\overline{Z}$ of $\overline{Q}$ as a Boolean array of size $|Q|$ such that, for any $z \in Q$, we have that $z \in Z$ if and only if the entry $\overline{z}$ of the array is true. Thus, testing for membership in Z takes time $O(1)$.

6.1 Algorithm ASP1(T,Q,X) for (SP1), and the 2D array BQ

A simple algorithm is to take any pair $v \in Q$ and $x \in X$, and test whether $\delta^{\otimes}([v],x) \neq [\mathrm{sink}]$. If, for the current v, the test is true for all $x \in X$, then v is added in $Q_X^{>}$. It is not difficult to see that this algorithm performs in time $O(|Q||X|k)$ and space $O(|Q|)$.

It turns out, however, that subproblem (SP1) needs to be solved repeatedly when we are looking for maximal solutions in the original main problems. For this reason, we shall need as a preprocessing step to compute a $|Q| \times |Q|$ Boolean array BQ such that $\mathrm{BQ}[\overline{v},\overline{v'}]$ is true if and only if $\delta^{\otimes}([v],v') \neq [\mathrm{sink}]$. This array can be computed in time $O(|Q|^2 k)$ and space $O(|Q|^2)$, as it involves $|Q|^2$ steps and, in each step, we run the DFA T on an input word of length k. Then, algorithm ASP1(T,Q,X) works as described in the previous paragraph, but now the test $\delta^{\otimes}([v],x) \neq [\mathrm{sink}]$ is reduced to whether $\mathrm{BQ}[\overline{v},\overline{x}]$ is true. Hence, assuming that the array BQ is available, the algorithm runs in time $O(|Q||X|)$.

6.2 Algorithm ASP2(T,Q,Z,Y) for (SP2)

Here we assume that $Q_Z^{>} = Y$, and we compute X by initializing it to Z, and then by repeatedly adding into X a new element from $V = Q - X$, provided that condition $Q_X^{>} = Y$ remains true. In particular, the algorithm is as follows.

```
ASP2(T,Q,Z,Y)
1  X = Z; V = Q − X;
2  while (V ≠ ∅)
3      do
4          Pick v ∈ V;
5          Use ASP1(T,Q,X ∪ {v}) to compute Y' = Q^>_{X∪{v}};
6          if (Y' = Y) X = X ∪ {v};
7          V = V − {v};
8  return X;
```

Lemma 9. *The above algorithm computes in time $O(|Q|^2(|Q|-|Z|))$ a subset X of Q such that $Z \subseteq X$ and X is maximal with "$X \subseteq Q$ and $Q^{>}_{X} = Y$".*

Proof. First we establish some notation. Let $m = |Q-Z|$ and, for $i = 1,\ldots,m$, let X_i be the value of X after the i-th iteration of the loop. Let $X_0 = Z$. Obviously, for all i, $Q^{>}_{X_i} = Y$ and, for $i > 0$, $X_{i-1} \subseteq X_i$. In order to show that X_m is maximal, it suffices to show that $Q^{>}_{X_m \cup \{u\}} \neq Y$, for all $u \in Q - X_m$. So consider any such u. As the loop visits every element in $Q - X_0$, it will visit u after the end of some iteration r, say. If it were $Q^{>}_{X_r \cup \{u\}} = Y$ then u would have been added into X and would be in X_m, which is a contradiction. Hence, $Q^{>}_{X_r \cup \{u\}} \neq Y$. Now, as $X_0 \subseteq X_r \cup \{u\} \subseteq X_m \cup \{u\}$, we have that

$$Q^{>}_{X_m \cup \{u\}} \subseteq Q^{>}_{X_r \cup \{u\}} \subseteq Q^{>}_{X_0} = Y,$$

which implies that $Q^{>}_{X_m \cup \{u\}}$ cannot be equal to Y, as required.

For the time complexity, we have that the highest order of magnitude term results from executing m times line 5 of the algorithm. In iteration i, it takes time $O(|Q||X_i|)$ to execute that line. The final estimate follows by noting that $|X_i| = O(|Q|)$. □

In the subsection below on Problem (P1), we give an example of executing Algorithm ASP2 based on input from Fig. 1.

6.3 Algorithm ASP3(T,Q,X,Y) for (SP3)

We assume that $T = \text{Trie}(S)^{\otimes}$ is given, as well as, the sets Q, X, Y. The required deterministic automaton T' accepting

$$S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Y$$

can be constructed as follows by modifying T in *linear time* in terms of the sizes of the given structures.

- The states of T' are $[\text{sink}]$, all states in $[Q]$, and all $[z]$ with z a prefix of X.
- The start state is $[\lambda]$, and the set of final states is $[Y]$.
- The transitions of T' are all the transitions of T involving only the above states.

It is not difficult to see that, indeed, the automaton T' accepts exactly those words in $S^{\otimes}$ that end with a suffix in Y and begin with a prefix in X, as required.

6.4 Algorithm for Problem (P1)

We return now to (P1), our first original main problem. Here is the proposed algorithm.

A1(S)
1 Compute $T = \text{Trie}(S)^{\otimes}$;
2 Compute the strongly connected components of T;
3 Pick a nontrivial component $[Q]$ – exit if none exists;
4 Compute the Boolean array BQ;
5 Compute two nonempty subsets Z, Y of Q such that $Q^{>}_{Z} = Y$;
6 Use ASP2(T,Q,Z,Y) to compute a maximal X with $Q^{>}_{X} = Y$;
7 Use ASP3(T,Q,X,Y) to compute and return the automaton for $S^{\otimes} \cap X\Sigma^* \cap \Sigma^* Y$;

Step 2 can be computed in linear time in terms of the size of $\text{Trie}(S)^{\otimes}$ – see e.g., [5]. Steps 3 and 5 are nondeterministic and allow for various possibilities. Here we show how to perform Step 5 in time $O(|Q|)$.

We pick any $z \in Q$ and let $Z = \{z\}$. We need to show that $Y = Q_Z^>$ is not empty. So, by Lemma 6, there is some state $[u] \in [Q]$ with a path of length k to $[z]$. As $[z] \neq [\text{sink}]$, the label of that path must be equal to z. Hence, $u \in Q_Z^>$. By the results in the previous sections, it is easy to see that the above algorithm operates correctly as described in the following theorem. Also, as the set Z is of cardinality 1, Step 6 of the algorithm runs in time $O(|Q|^3)$.

Theorem 4. *The above algorithm computes, for any given subword constraint S of some length k, a deterministic automaton accepting a nonempty language D such that the words of D are of length at least k and D is maximal with $D^* \subseteq S^\otimes$; or the algorithm reports that no such language exits. The algorithm runs in time $O(|Q|^3 + |Q|^2 k)$ and space $O(|Q|^2 k)$, where $[Q]$ is any nontrivial strongly connected component of* $\text{Trie}(S)^\otimes$ *– if such exists.*

Example 3. Going back to Fig. 1 with $Q = \{0111, 1111, 1110, 1101, 1011\}$, Step 5 of Algorithm A1(S), as detailed above, can be performed by choosing, for instance, $Z = \{1011\}$ and computing $Y = Q_{\{1011\}}^> = \{1011, 0111, 1111\}$. Then, in Step 6 of A1($S$), the word 0111 will not be added to Z, as $\delta^\otimes([1011], 0111) = [\text{sink}]$. On the other hand, the words 1101, 1110, 1111 will be added to Z to obtain $X = \{1011, 1101, 1110, 1111\}$ with $Q_X^> = Y$. Thus, the language

$$D = S^\otimes \cap \{1011, 1101, 1110, 1111\}\Sigma^* \cap \Sigma^*\{1011, 0111, 1111\}$$

is maximal with $D^* \subseteq S^\otimes$.

6.5 Algorithm for Problem (P2)

The desired algorithm is very similar to the one used for Problem (P1). In particular Steps 3 and 5 are different here.

A2(S)
1 Compute $T = \text{Trie}(S)^\otimes$;
2 Compute the strongly connected components of T;
3 Pick a component $[Q]$ containing a fork state – exit if none exists;
4 Compute the Boolean array BQ;
5 Compute subsets Z, Y of Q such that $|Z| \geq 2$ and $Q_Z^> = Y$;
6 Use ASP2(T, Q, Z, Y) to compute a maximal X with $Q_X^> = Y$;
7 Use ASP3(T, Q, X, Y) to compute and return the automaton for $S^\otimes \cap X\Sigma^* \cap \Sigma^* Y$;

For the correctness of the algorithm (in particular Steps 3 and 5), we first note that there is a nontrivial language D with $D^* \subseteq S^\otimes$ if and only if there is a strongly connected component of $\text{Trie}(S)^\otimes$ containing a fork state – the 'only if' part follows from Theorem 3 and the 'if' part from Lemma 4 and Lemma 7. Thus, if there is no fork state in some component, Step 3 correctly decides to terminate the algorithm. On the other hand, if a fork state is found, then, according to Lemma 7, we can define effectively a two-element subset Z of Q such that $\langle Q, Z\rangle$ is nontrivial. Then, in Step 6, the algorithm attempts to add elements to Z in order to obtain a subset X of Q that is maximal with $Q_X^> = Y$. As before, the resulting set $\langle Q, X\rangle = S^\otimes \cap X\Sigma^* \cap \Sigma^* Y$ is maximal with $\langle Q, X\rangle^* \subseteq S^\otimes$ and, of course, the set is also nontrivial as it contains the nontrivial set $\langle Q, Z\rangle$. The time complexity of the above algorithm is the same as that of A1(S), as the most expensive steps are again Steps 4 and 6. So we have established the following result.

Theorem 5. *The above algorithm computes, for any given subword constraint S of some length k, a deterministic automaton accepting a nontrivial language D such that the words of D are of length at least k and D*

is maximal with $D^* \subseteq S^{\otimes}$; or the algorithm reports that no such language exits. The algorithm runs in time $O(|Q|^3+|Q|^2k)$ and space $O(|Q|^2k)$, where $[Q]$ is any strongly connected component of $\mathrm{Trie}(S)^{\otimes}$ *containing a fork state – if such exists.*

Example 4. Going back again to Fig. 1 with $Q = \{0111, 1111, 1110, 1101, 1011\}$, we see that $[0111]$ is a fork state with transitions going to states $[1110], [1111]$. Let $Z = \{1110, 1111\}$. Then, Step 5 of the above algorithm will compute that $Y = Q_Z^{>} = Q$ and, then, Step 6 will find that no other words will be added to Z, that is, $X = Z$. Thus, the language

$$D = S^{\otimes} \cap \{1110, 1111\}\Sigma^* \cap \Sigma^*\{0111, 1111, 1110, 1101, 1011\}$$

is nontrivial and maximal with $D^* \subseteq S^{\otimes}$.

7 Concluding Remarks

We have proposed a quadratic-time test for deciding whether a given NFA language L has exponential density, and we observed that this question has consequences in the encoding capability of L. Is it possible to obtain a sub-quadratic algorithm when L is given via a nondeterministic automaton? What can we say about the efficiency (e.g., in terms of information rate) of the encoding parts of an encoding language? It seems that the results of [16] could help in this direction.

We have also considered the problem of characterizing nontrivial languages D that are maximal with the property $D^* \subseteq S^{\otimes}$. Our characterization is structural and leads to algorithmically polynomial solutions. The recent work of [12] solves the more general problem of computing all maximal solutions of $D^* \subseteq R$, for any given regular language R using a brute force method of exponential time complexity. Is it possible to dig deeper and combine the two approaches with the aim of computing efficiently the more general problem? Is it possible to compute efficiently nontrivial D's that are maximal with the conjunctive property "$D^* \subseteq S^{\otimes}$ and D is a code (or a prefix code)"?

References

1. C. Campeanu, S. Konstantinidis: State complexity of the subword closure operation with applications to DNA coding. International Journal of Foundations of Computer Science 19:5 (2008), 1099–1112.
2. Crochemore, M., Hancart, C.: Automata for matching patterns. In [13], 399–462.
3. Cui, B.: Encoding methods for DNA languages defined via the subword closure operation. MSc Thesis, Department of Mathematics and Computing Science, Saint Mary's University, Halifax, Canada 2007.
4. Cui, B., Konstantinidis, S.: DNA coding using the subword closure operation. In [7], pp 284–289.
5. Dasgupta, S., Papadimitriou, C.H., Vazirani, U.V.: *Algorithms*. McGraw-Hill, 2006.
6. Ferretti, C., Mauri, G., Zandron, C. (eds): DNA Computing, 10th International Workshop on DNA Computing, DNA 10, Milan, Italy, June 7-10, 2004, Revised Selected Papers. Lecture Notes in Computer Science, Vol. 3384 (2005), Springer, Berlin.
7. Garzon, M.H., Yan, H.(eds): DNA Computing, 13th International Meeting on DNA Computing, DNA13, Memphis, TN, USA, June 4-8, 2007, Revised Selected Papers. Lecture Notes in Computer Science, Vol. 4848 (2008), Springer, Berlin.
8. Havel, I. M., Koubek, V. (eds): Proceedings. 7th International Symposium on Mathematical Foundations of Computer Science, MFCS 92, Prague, Czechoslovakia, August 1992. Lecture Notes in Computer Science, Vol. 629 Springer-Verlag, London, UK (1992)
9. Hirsch, E.A., Razborov, A. , Semenov, A., Slissenko, A. (eds): Proceedings. 3rd International Computer Science Symposium in Russia, CSR 08, Moscow, Russia, June 2008. Lecture Notes in Computer Science, Vol. 5010 Springer-Verlag, London, UK (2008)
10. Ibarra, O.H., Dang, Z. (eds): 10th International Conference on Developments in Language Theory, Santa barbara, CA, USA, June 26–29, 2006. Lecture Notes in Computer Science, Vol. 4036 (2006). Springer-Verlag, Berlin.
11. Kari, L., Konstantinidis, S., Sosík, P.: Bond-free languages: formalizations, maximality and construction methods. In [6], 169–181.
12. Kari, L., Seki, S.: Schema for parallel insertion and deletion. *Developments in Language Theory 2010*, *Lecture Notes in Computer Science*, Vol. 6224. Springer-Verlag, Berlin Heidelberg (2010) 376–393.

13. Rozenberg, G., Salomaa, A.: (eds). *Handbook of Formal Languages, Vol. I.* Springer-Verlag, Berlin, 1997.
14. Shur, A. M.: Combinatorial Complexity of Rational Languages. Discr. Anal. and Oper. Research, Ser. 1, 12:2 (2005), 78–99, in Russian.
15. Shur, A. M.: Factorial Languages of Low Combinatorial Complexity. In [10], 397–407.
16. Shur, A. M.: Combinatorial Complexity of Regular Languages. In [9], 289–301.
17. Szilard, A., Yu, S., Zhang, K., Shallit, J.: Characterizing Regular Languages with Polynomial Densities. In [8], 494–503.
18. Yu, S.: Regular Languages. In [13], 41–110.