State Complexity of Neighbourhoods and Approximate Pattern Matching

Timothy Ng, David Rappaport, and Kai Salomaa(✉)

School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada
{ng,daver,ksalomaa}@cs.queensu.ca

Abstract. The neighbourhood of a language L with respect to an additive distance consists of all strings that have distance at most the given radius from some string of L. We show that the worst case (deterministic) state complexity of a radius r neighbourhood of a language recognized by an n state nondeterministic finite automaton A is $(r+2)^n$. The lower bound construction uses an alphabet of size linear in n. We show that the worst case state complexity of the set of strings that contain a substring within distance r from a string recognized by A is $(r+2)^{n-2}+1$.

Keywords: Regular languages · State complexity · Lower bounds · Additive distance

1 Introduction

The similarity of strings is often defined using the edit distance [11,16], also known as the Levenshtein distance [14]. The edit distance is particularly useful for error-correction and error-detection applications [7–10,12]. A useful property is that the edit distance is additive with respect to concatenation of strings in the sense defined by Calude et al. [4].

If the distance of any two distinct strings of a language L is greater than r, the language L can detect up to r errors [9,11,13] (assuming the errors have unit weight). Alternatively we can consider what the shortest distance is between strings in languages L_1 and L_2, that is, what is the smallest number errors that transform a string of L_1 into a string of L_2. Calude at al. [4] showed that the neighbourhood of a regular language with respect to an additive distance is always regular. Additive quasi-distances preserve regularity as well [4]. This gives rise to the question how large is the deterministic finite automaton (DFA) needed to recognize the neigbourhood of a regular language. Informally, determining the optimal size of the DFA for the neighbourhood gives the *state complexity of error detection.* Note that since complementation does not change the size of a DFA, the size of the minimal DFA for the neighbourhood of L of radius r equals to the state complexity of the set of strings that have distance at least $r+1$ from any string in L.

Povarov [17] showed that the Hamming neighbourhood of radius one of an n-state DFA language can be recognized by a DFA of size $n \cdot 2^{n-1}+1$ and also

I. Potapov (Ed.): DLT 2015, LNCS 9168, pp. 389–400, 2015.
DOI: 10.1007/978-3-319-21500-6_31

gave a lower bound $\frac{3}{8}n \cdot 2^n - 2^{n-4} + n$ for its state complexity. Using a weighted finite automaton construction the third author and Schofield [18] gave an upper bound of $(r+2)^n$ for the neighbourhood of radius r of an n-state DFA-language. No good lower bounds are known for neighbourhoods of radius at least two.

The string matching problem consists of finding occurrences of a particular string in a text [2]. El-Mabrouk [6] considers the problem of pattern matching with r mismatches from a descriptional complexity point of view. Given a pattern P of length m and a text T, the problem is to determine whether T contains substrings of length m having characters differing from P in at most r positions, that is, substrings having Hamming distance at most r from P. For a pattern $P = a^m$ consisting of occurrences of only one character, the state complexity was shown to be $\binom{m+1}{r+1}$ [6].

The state complexity of $\Sigma^* L \Sigma^*$ was considered by Brzozowski, Jirásková, and Li [3] and was shown to have a tight bound of $2^{n-2} + 1$. A DFA recognizing $\Sigma^* L \Sigma^*$ can be viewed to solve the exact string matching problem. In the terminology of Brzozowski et al. [3], $\Sigma^* L \Sigma^*$ is a two-sided ideal and the descriptional complexity of related subregular language families was studied recently by Bordihn et al. [1].

This paper studies the descriptional complexity of neighbourhoods and of approximate string matching. As our main result we give a lower bound $(r+2)^n$ for the size of a DFA recognizing the radius r neighbourhood of an n-state regular language. The lower bound matches the previously known upper bound [18]. The bound can be reached either using a neighbourhood of an n-state DFA language with respect to an additive quasi-distance or using a neighbourhood of an n state NFA (nondeterministic finite automaton) language using an additive distance.

The lower bound constructions use an alphabet of size linear in n. A further limitation is that the (quasi-)distance associates different values to different edit operations. The precise state complexity of the edit distance with unit error costs remains open.

We also show that if L is recognized by an n-state NFA the set of strings that contain a substring within distance r from a string in L with respect to an additive (quasi-)distance is recognized by a DFA of size $(r+2)^{n-2} + 1$ and that this bound cannot be improved in the worst case. When r is zero the result coincides with the state complexity of two-sided ideals [3].

2 Preliminaries

Here we briefly recall some definitions and notation used in the paper. For all unexplained notions on finite automata and regular languages the reader may consult the textbook by Shallit [19] or the survey by Yu [20]. A survey of distances is given by Deza and Deza [5] and the notion of quasi-distance is from Calude et al. [4].

We denote by Σ a finite alphabet, Σ^* the set of words over Σ, and ε the empty word. A *nondeterministic finite automaton* (NFA) is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is an alphabet, δ is a multi-valued transition function $\delta : Q \times \Sigma \to 2^Q$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final states. We extend the transition function δ to $Q \times \Sigma^* \to 2^Q$ in the usual way. A word $w \in \Sigma^*$ is *accepted* by A if $\delta(q_0, w) \cap F \neq \emptyset$ and the language recognized by A consists of all strings accepted by A. The automaton A is a *deterministic finite automaton* (DFA) if, for all $q \in Q$ and $a \in \Sigma$, $\delta(q, a)$ either consists of one state or is undefined. The DFA A is complete if $\delta(q, a)$ is defined for all $q \in Q$ and $a \in \Sigma$. Two states p and q of a DFA A are equivalent if $\delta(p, w) \in F$ if and only if $\delta(q, w) \in F$ for every string $w \in \Sigma^*$. A complete DFA A is *minimal* if each state $q \in Q$ is reachable from the initial state and no two states are equivalent. The (right) Kleene congruence of a language $L \subseteq \Sigma^*$ is the relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ defined by setting

$$x \equiv_L y \text{ iff } [(\forall z \in \Sigma^*) \;\; xz \in L \Leftrightarrow yz \in L].$$

The language L is regular if and only if the index of $\equiv_L$ is finite and, in this case, the index of $\equiv_L$ is equal to the size of the minimal DFA for L [19]. The minimal DFA for a regular language L is unique. The *state complexity* of L, $\mathrm{sc}(L)$, is the size of the minimal complete DFA recognizing L.

A function $d : \Sigma^* \times \Sigma^* \to [0, \infty)$ is a *distance* if it satisfies for all $x, y, z \in \Sigma^*$ the conditions $d(x, y) = 0$ if and only if $x = y$, $d(x, y) = d(y, x)$, and $d(x, z) \leq d(x, y) + d(y, z)$. The function d is a *quasi-distance* [4] if it satisfies conditions 2 and 3 and $d(x, y) = 0$ if $x = y$; that is, a quasi-distance between two distinct elements can be zero. In the following, unless otherwise mentioned, we consider only *integral* (quasi-)distances; that is, d is always a function $\Sigma^* \times \Sigma^* \to \mathbb{N}_0$.

The neighbourhood of a language L of radius r is the set

$$E(L, d, r) = \{x \in \Sigma^* \mid (\exists w \in L) d(x, w) \leq r\}.$$

A distance d is *finite* if for all nonnegative integers r the neighbourhood of radius r of any string with respect to d is finite. A distance d is *additive* [4] if for every factorization of a string $w = w_1 w_2$ and radius $r \geq 0$,

$$E(w, d, r) = \bigcup_{r_1 + r_2 = r} E(w_1, d, r_1) \cdot E(w_2, d, r_2).$$

A neighbourhood of a regular language with respect to an additive quasi-distance is regular [4].

The following upper bound for the state complexity of the neighbourhood of a regular language with respect to additive distances is known by [18] and by [15] for additive quasi-distances. The results are stated in terms of weighted finite automata.

Proposition 2.1 ([15,18]). *If A is an n-state NFA and d an additive quasi-distance, then for any $r \in \mathbb{N}$, $\mathrm{sc}(E(L(A), d, r)) \leq (r + 2)^n$.*

We will use also the NFA construction for a neighbourhood due to Povarov [17]. Informally, the construction makes $r + 1$ copies of an NFA A, with each copy corresponding to a cumulative error ranging from 0 to r. A transition from a level i to a level $i' > i$ occurs when there is a transition that does not exist in A. There are $r+1$ such copies of A to allow for at most r errors. Strictly speaking, [17] deals with additive distances but exactly the same construction works for quasi-distances.

Proposition 2.2 ([17]). *If A is an NFA with n states and d is an additive quasi-distance, then $E(L(A), d, r)$ has an NFA of size $n \cdot (r+1)$.*

3 State Complexity of Additive Neighbourhoods

As the main result of this section we give a tight lower bound for the state complexity of a neighbourhood of a regular language given by a DFA (respectively, by an NFA) with respect to an additive quasi-distance (respectively, an additive distance).

For $n \in \mathbb{N}$ we consider an alphabet

$$\Sigma_n = \{a_1, \ldots, a_{n-1}, b_1, \ldots, b_n, c_1, \ldots, c_{n-1}\}. \tag{1}$$

For $r \in \mathbb{N}$, we define a quasi-distance $d_r : \Sigma_n^* \times \Sigma_n^* \to \mathbb{N}_0$ by the conditions:

- $d_r(a_i, a_j) = r+1$ for $i \neq j$
- $d_r(b_i, b_j) = 1$ for $i \neq j$
- $d_r(a_i, b_j) = d_r(c_i, b_j) = r+1$ for all $1 \leq i, j \leq n$
- $d_r(a_i, c_i) = 0$ for $1 \leq i \leq n-1$
- $d_r(c_i, c_j) = r+1$ for all $1 \leq i, j \leq n$
- $d_r(a_i, c_j) = r+1$ for all $i \neq j$
- $d_r(\sigma, \varepsilon) = r+1$ for all $\sigma \in \Sigma$.

Note that the value $d_r(\sigma, \varepsilon)$ denotes the cost of the deletion and insertion operations and that the listed substitution, insertion, and deletion operations on elements of Σ_n define a unique additive quasi-distance of Σ_n^* [4].

Lemma 3.1. *The function d_r is an additive quasi-distance.*

We define the following family of incomplete DFAs. Let $A_n = (Q_n, \Sigma_n, \delta, 1, \{n\})$ be a DFA with n states where $Q_n = \{1, \ldots, n\}$ and Σ_n is as in (1). The transition function δ is defined by setting

- $\delta(i, a_i) = i+1$ for $1 \leq i \leq n-1$
- $\delta(i, a_j) = i$ for $1 \leq i \leq n-2$ and $i+1 \leq j \leq n-1$
- $\delta(i, b_j) = i$ for $1 \leq i \leq n-1$ and $j = i-1$ or $i+1 \leq j \leq n$
- $\delta(i, c_i) = i$ for $1 \leq i \leq n-1$

All transitions not listed are undefined. The DFA A_n is depicted in Figure 1.

The quasi-distance d_r identifies the symbols a_i and c_i, $1 \leq i \leq n$. By using two different symbols that have distance zero in our quasi-distance allows us to define A_n to be deterministic. By identifying a_i and c_i we can later modify

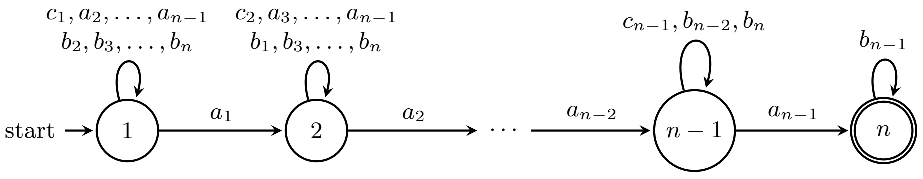

Fig. 1. The DFA A_n

the construction to give a lower bound for the neighbourhood of a language recognized by an NFA with respect to a distance (see Lemma 3.4).

To establish a lower bound for the state complexity of the neighbourhood $E(L(A_n), d_r, r)$ we define a set S of strings that are all pairwise inequivalent with respect to the Kleene congruence of the neighbourhood. First we construct an NFA $B_{n,r}$ for $E(L(A_n), d_r, r)$ and the inequivalence of the strings in S is verified using properties of $B_{n,r}$.

Suppose we have a DFA $A = (Q, \Sigma, \delta, q_0, F)$. Using Proposition 2.2 (due to [17]), an NFA $B = (Q', \Sigma, \delta', q_0', F')$ which recognizes the neighbourhood of radius r of $L(A)$ with respect to a quasi-distance d is defined by setting $Q' = Q \times \{0, \ldots, r\}$, $q_0' = (q_0, 0)$, $F' = F \times \{0, \ldots, r\}$ and the transitions of δ' for $q \in Q$, $0 \le k \le r$ and $a \in \Sigma$ are defined as

$$\delta'((q,k),a) = (\delta(q,a),k) \cup \bigcup_{b \in (\Sigma \cup \{\varepsilon\}) \setminus \{a\}} \{(\delta(q,b), k + d(a,b)) \mid k + d(a,b) \le r\}.$$

Now as described above we construct the NFA

$$B_{n,r} = (Q_n', \Sigma_n, \delta', q_0', F'), \tag{2}$$

shown in Figure 2, which recognizes the neighbourhood of $L(A_n)$ of radius r with respect to the quasi-distance d_r, where $Q_n' = Q_n \times \{0, 1, \ldots, r\}$, $q_0' = (q_0, 0)$, $F' = F \times \{0, 1, \ldots, r\}$ and the transition function δ' is defined by

- $\delta'((q,j), a_q) = \{(q,j), (q+1,j)\}$ for $1 \le q \le n-1$,
- $\delta'((q,j), a_{q'}) = \{(q,j)\}$ for all $1 \le q \le n-1$ and $q \le q' \le n-1$,
- $\delta'((q,j), b_i) = \{(q,j+1)\}$ for $1 \le q \le n$ and $i = 1, \ldots, q-2, q$,
- $\delta'((q,j), b_i) = \{(q,j)\}$ for $1 \le q \le n$ and $i = q-1, q+1, \ldots, n$,
- $\delta'((q,j), c_q) = \{(q,j), (q+1,j)\}$ for $1 \le q \le n-1$.

All transitions not listed above are undefined. Note that since in the distance d_r the cost of inserting/deleting a symbol is $r+1$ and $B_{n,r}$ recognizes a neighbourhood of radius r there are no error transitions corresponding to insertion/deletion. For the same reason the only error transitions for substitution correspond to substituting b_i with b_j, $i \neq j$. The distance between a_i and c_i is zero (no error), and all other substitutions have cost $r+1$.

For $0 \le k_i \le r+1$, $1 \le i \le n$, we define the string

$$w(k_1, \ldots, k_n) = a_1 b_1^{k_1} a_2 b_2^{k_2} \cdots a_{n-1} b_{n-1}^{k_{n-1}} b_n^{k_n}. \tag{3}$$

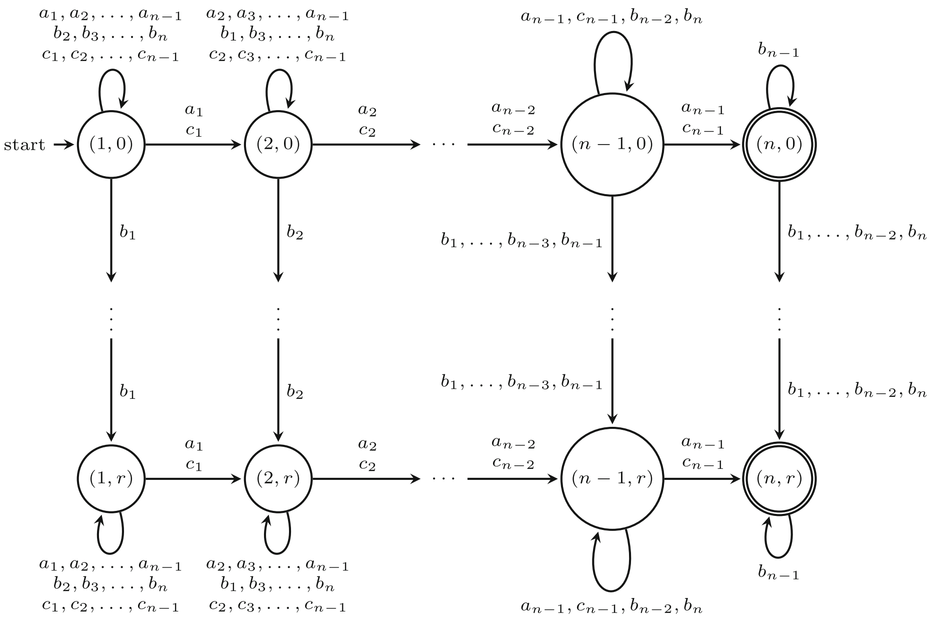
start
$(1,0)$
$(2,0)$
$(n-1,0)$
$(n,0)$
$(1,r)$
$(2,r)$
$(n-1,r)$
(n,r)
$a_1, a_2, \ldots, a_{n-1}$ / $b_2, b_3, \ldots, b_n$ / $c_1, c_2, \ldots, c_{n-1}$
$a_2, a_3, \ldots, a_{n-1}$ / $b_1, b_3, \ldots, b_n$ / $c_2, c_3, \ldots, c_{n-1}$
$a_{n-1}, c_{n-1}, b_{n-2}, b_n$
b_{n-1}
a_1 c_1
a_2 c_2
a_{n-2} c_{n-2}
a_{n-1} c_{n-1}
b_1
b_2
$b_1, \ldots, b_{n-3}, b_{n-1}$
$b_1, \ldots, b_{n-2}, b_n$

Fig. 2. The NFA $B_{n,r}$

The next lemma establishes a technical property of the computations of the NFA $B_{n,r}$ on the strings $w(k_1, \ldots, k_n)$. The property is then used to establish that the strings are pairwise inequivalent with respect to the language recognized by $B_{n,r}$.

Lemma 3.2. *If $k_i \leq r$, then there exists a computation C_i of the NFA $B_{n,r}$ which reaches the state (i, k_i) at the end of the input $w(k_1, \ldots, k_n)$, $1 \leq i \leq n$. There is no computation of $B_{n,r}$ on $w(k_1, \ldots, k_n)$ that reaches a state (i, k_i') with $k_i' < k_i$. Furthermore, if $k_i = r + 1$, no computation of $B_{n,r}$ reaches at the end of $w(k_1, \ldots, k_n)$ a state where the first component is i.*

Proof. We verify that a computation C_i can reach state (i, k_i), $k_i \leq r$. First consider the case $i < n$. For $j = 1, \ldots, i-1$, a_j takes state $(j, 0)$ to $(j+1, 0)$ and the next k_j symbols b_j are read using the self-loop in state $(j+1, 0)$. In this way the computation reaches state $(i, 0)$ where we read a_i using the self-loop and then reading the k_i symbols b_i the computation reaches (i, k_i). In state (i, k_i) the remaining suffix $a_{i+1}b_{i+1}^{k_{i+1}} \cdots a_{n-1}b_{n-1}^{k_{n-1}}b_n^{k_n}$ is consumed using the self-loops. Second, in the case $i = n$ similarly as above the computation after symbol a_{n-1} reaches state $(n, 0)$, the symbols b_{n-1} are read using self-loops and reading the k_n symbols b_n takes us to state (n, k_n).

To verify the second part of the lemma we first observe the following. The only transitions of $B_{n,r}$ which move from a state (i, j) to a state of the form $(i+1, j')$, $0 \leq j \leq r$, are on symbols a_i and c_i. Note that since the distance d_r

associates cost $r+1$ to insertions and deletions, as well as to replacing a_i or c_i by any other symbol, the NFA $B_{n,r}$ does not have error transitions that change the first component of a state. Since $d_r(a_i, c_i) = 0$, we can treat them as the same letter and for convenience, we refer only to a_i. Thus, the only way to reach a state (q, j) for any $j \leq r$, is by taking transitions $((i,j), a_i, (i+1, j))$ on each occurrence of a_i in $w(k_1, \ldots, k_n)$ for $i < q$. Otherwise, the computation remains in some state (i', j) for $i' < q$.

Now we show that there is no computation of $w(k_1, \ldots, k_n)$ that can reach a state (j, k'_j) with $k'_j < k_j$. As discussed above, the only way for the computation to end in a state (j, i), $0 \leq i \leq r$, is by reaching the state $(j, 0)$ when consuming the prefix $a_1 b_1^{k_1} \cdots a_{j-1} b_{j-1}^{k_{j-1}}$ and then reading a_j using a self-loop. There is no other way to reach a state (j, i) for any i, since exiting the states with second components zero (corresponding to the original DFA) requires reading some $a_{j'}$ with a self-loop, after which there is no transition which can be taken to move to a state $(j'+1, i)$. If in the state $(j, 0)$ the symbol a_j is not read with a self-loop then the first component becomes $j+1$ and we cannot reach a state (j, i) with the remaining suffix. Thus, from $(j, 0)$ the NFA is forced to read the following k_j symbols b_j with error transitions, ending in the k_j-th level in the state (j, k_j).

Exactly the same argument verifies that in the case $k_j = r+1$, no computation can end in a state where the first component is j. As above it is seen that to do this we must in the state $(j, 0)$ read the symbol a_j with a self-loop and after attempting to read the following $r+1$ symbols b_j with an error transition the computation becomes undefined. □

With the previous lemma we can now establish a lower bound for the state complexity of the neighbourhood of $L(A_n)$.

Lemma 3.3. *Let A_n be the DFA as in Figure 1. The strings $w(k_1, \ldots, k_n)$, $0 \leq k_i \leq r+1$, $1 \leq i \leq n$, are all pairwise inequivalent with respect to the Kleene congruence of $E(L(A_n), d_r, r)$.*

Proof. We consider two distinct strings $w(k_1, \ldots, k_n)$ and $w(k'_1, \ldots, k'_n)$ with $0 \leq k_i, k'_i \leq r+1$ for $i = 1, \ldots, n$. There exists an index j such that $k_j \neq k'_j$ and without loss of generality, we have $k_j < k'_j$. To distinguish the strings $w(k_1, \ldots, k_n)$ and $w(k'_1, \ldots, k'_n)$ consider the word $z = b_j^{r-k_j} a_{j+1} \cdots a_{n-1}$. The string z is well-defined since $k_j < k'_j \leq r+1$ and so $r - k_j \geq 0$.

Let $B_{n,r}$ be the NFA constructed for $E(L(A), d_r, r)$ as in (2). We claim that $w(k_1, \ldots, k_n) \cdot z \in L(B_{n,r})$ but $w(k'_1, \ldots, k'_n) \cdot z \notin L(B_{n,r})$. We note that by Lemma 3.2, $B_{n,r}$ has a computation on $w(k_1, \ldots, k_n)$ that ends in state (j, k_j). Note that $k_j \leq r$. When continuing the computation on the string z, by reading the $r - k_j$ symbols b_j's, the machine is taken to the state (j, r). Then, reading the suffix $a_{j+1} \cdots a_{n-1}$ takes the machine to the accepting state (n, r).

To show $w(k'_1, \ldots, k'_n) \cdot z \notin L(B_{n,r})$, we consider from which states of $B_{n,r}$ an accepting state, that is, a state with first component n is reachable on the string z. We recall that in $B_{n,r}$ the transitions on b_j cannot change the first component of the state. (According to the definition of $B_{n,r}$ the reason for this is that d_r associates cost $r+1$ to insertion/deletion or to subsitute a symbol a_i, c_i by b_j.)

Thus, for $B_{n,r}$ to reach an accepting state (with first component n) on the string $w(k'_1, \ldots, k'_n) \cdot z$, a computation must reach a state of the form (j, ℓ_j) on the prefix $w(k'_1, \ldots, k'_n)$. By Lemma 3.2, this is possible only if $\ell_j \geq k'_j$. From state (j, ℓ_j), $\ell_j \geq k'_j$, reading the substring $b_j^{r-k_j}$ takes the machine to an undefined state, as it is not possible to make $r - k_j$ error transitions on b_j in a state where the second component is $\ell_j > k_j$. This means that $B_{n,r}$ cannot accept the string $w(k'_1, \ldots, k'_n) \cdot z$.

Thus, each string $w(k_1, \ldots, k_n)$, $0 \leq i \leq r+1$, $1 \leq i \leq n$, defines a distinct equivalence class of $\equiv_{E(L(A),d_r,r)}$. □

As a corollary of the proof of the previous lemma we get also a lower bound for the state complexity of the neighbourhood of an NFA-language with respect to an additive distance.

Lemma 3.4. *For $n, r \in \mathbb{N}$ there exists an additive distance d'_r and an NFA A'_n over an alphabet Σ'_n of size $2n-1$ such that*

$$\mathrm{sc}(E(L(A'_n), d'_r, r)) \geq (r+2)^n.$$

Proof. Choose $\Sigma'_n = \{a_1, \ldots, a_{n-1}, b_1, \ldots, b_n\}$ and d'_r is the restriction of d_r to the alphabet Σ'_n (where d_r is the quasi-distance of Lemma 3.1). The function d'_r does not assign distance zero to any pair of distinct elements.

The NFA A'_n is obtained from the DFA A_n in Figure 1 by replacing all c_i-transitions by a_i-transitions, $1 \leq i \leq n-1$. Thus, A'_n is nondeterministic. An NFA $B'_{n,r}$ for the neighbourhood $E(L(A'_n), d'_r, r)$ is obtained from the NFA $B_{n,r}$ in (2) simply by omitting all transitions on c_i, $1 \leq i \leq n-1$. Note that in $B_{n,r}$ the transitions on c_i exactly coincide with the transitions on a_i, $1 \leq i \leq n-1$, reflecting the situation that $d_r(a_i, c_i) = 0$.

The strings $w(k_1, \ldots, k_n)$ (as in (3)) did not involve any symbols c_i, and the proof of Lemma 3.3 remains the same, word for word, just by replacing $B_{n,r}$ with $B'_{n,r}$. □

Now putting together Lemma 3.3, Lemma 3.4 and Proposition 2.1, we have:

Theorem 3.1. *If d is an additive quasi-distance, A is an NFA with n states and $r \in \mathbb{N}$,*

$$\mathrm{sc}(E(L(A), d, r) \leq (r+2)^n).$$

There exists an additive quasi-distance d_r and a DFA A with n states over an alphabet of size $3n-2$ such that $\mathrm{sc}(E(L(A), d_r, r) = (r+2)^n$.

There exists an additive distance d'_r and an NFA A' with n states over an alphabet of size $2n-1$ such that $\mathrm{sc}(E(L(A'), d'_r, r) = (r+2)^n$.

The lower bound construction has the trade-off of either using a DFA and a quasi-distance or an NFA and a distance, respectively. It would be interesting to know whether or not the general upper bound can be improved in cases where we are using a distance and the language is specified by a DFA.

4 State Complexity of Pattern Matching

We consider an extension of the pattern matching problem with mismatches in the sense of El-Mabrouk [6]. For a given finite automaton A and an additive quasi-distance d we construct a DFA for the language $\Sigma^* E(L(A), d, r)\Sigma^*$, that is, the set of strings that contain a substring within distance r from a string of $L(A)$. The construction gives an upper bound for the pattern matching problem and using a modification of the constructions in the previous section we show that the upper bound is optimal.

Lemma 4.1. *Let $A = (Q, \Sigma, \delta, q_0, F_A)$ be an n-state NFA with $k \geq 1$ final states and d is an additive quasi-distance. Then the language*

$$L_1 = \Sigma^* E(L(A), d, r)\Sigma^*$$

can be recognized by a DFA B with $(r + 2)^{n-1-k} + 1$ states.

Proof. Let $Q = \{q_0, q_1, \ldots, q_{n-1}\}$. If $q_0 \in F_A$, then $L_1 = \Sigma^*$ and there is nothing to prove. Thus, in the following we can assume that $F = \{q_{n-k}, q_{n-k+1}, \ldots, q_{n-1}\}$, $1 \leq k \leq n-1$. Furthermore, without loss of generality we assume that

$$(\forall w \in \Sigma^*)\ \delta(q_0, w) \cap F_A \neq \emptyset \text{ implies } d(\varepsilon, w) > r. \tag{4}$$

If the above condition does not hold, $\varepsilon \in E(L(A), d, r)$ and there is nothing to prove.

The DFA B recognizing L_1 operates as follows. Roughly speaking, B is looking for a substring of the input that belongs to $E(L(A), d, r)$. For this purpose, for all non-final states q_z of A, the deterministic computation of B keeps track of the smallest cumulative error between a string that takes q_0 to q_z and any suffix of the input processed thus far. Note that for the initial state q_0 this value is always zero and, hence, the states of P store the cumulative error only for the nonfinal states $q_1, \ldots, q_{n-k-1}$. When B has found a substring belonging to $E(L(A), d, r)$ the computation goes to the final state p_f and after that accepts an arbitrary suffix. Next we give the definition of B and after that include a brief correctness argument.

Define $B = (P, \Sigma, \gamma, p_0, F_B)$ with set of states

$$P = \{(i_1, \ldots, i_{n-k-1}) \mid 0 \leq i_j \leq r + 1,\ j = 1, \ldots, n - k - 1\} \cup \{p_f\},$$

the initial state is $p_0 = (h_1, \ldots, h_{n-k+1})$ where

$$h_z = \inf\{d(\varepsilon, w) \mid q_z \in \delta(q_0, w)\},\ \ 1 \leq z \leq n - k - 1,$$

and the set of final states is defined as $F_B = \{p_f\}$. Note that by (4) we know that $\varepsilon \notin L_1$. Next we define the transitions of B. First, $\gamma(p_f, b) = p_f$ for all $b \in \Sigma$. For $\mathbf{p} = (i_1, \ldots, i_{n-k-1}) \in P$, $0 \leq i_z \leq r + 1$, $z = 1, \ldots, n - k - 1$, and $b \in \Sigma$ we define

(i) $\gamma(\mathbf{p}, b) = p_f$ if $(\exists 1 \le z \le n-k-1)(\exists w \in \Sigma^*)\ \ \delta(q_z, w) \cap F_A \neq \emptyset$ and $i_z + d(b, w) \le r$;

(ii) and if the conditions in (i) do not hold, then $\gamma(\mathbf{p}, b) = (j_1, \ldots, j_{n-k-1})$, where, for $x = 1, \ldots, n-k-1$,

$$\begin{aligned} j_x &= \inf[\ \{i_z + d(b, w) \mid q_x \in \delta(q_z, w), 1 \le z \le n-k-1\} \\ &\quad \cup\ \{d(b, w) \mid q_x \in \delta(q_0, w)\}\ ]. \end{aligned}$$

In a state of the form $(i_1, \ldots, i_{n-k-1}) \in P$, the component i_z, $1 \le z \le n-k-1$, keeps track of the smallest distance $d(u_{\text{suf}}, w)$ where u_{suf} is a suffix of the input processed up to that point and w is a string that in A takes the initial state q_0 to state q_z. The smallest error between the suffix ε and a string that in A reaches q_0 is always zero and this value is not stored in the state of B. If the computation has found a substring in $E(L(A), d, r)$, the state of B will be p_f. □

By modifying the construction used in the proof of Lemma 3.4 (and Lemma 3.3) we give a lower bound that matches the upper bound from Lemma 4.1.

Lemma 4.2. *For $n, r \in \mathbb{N}$, there exist an additive distance d and an NFA A with n states defined over an alphabet Σ of size $2n-1$ such that the minimal DFA for $\Sigma^* E(L(A), d, r)\Sigma^*$ must have at least $(r+2)^{n-2} + 1$ states.*

Proof. Choose $\Sigma_n = \{a_1, \ldots, a_{n-1}, b_1, \ldots, b_n\}$ and let A'_n and d'_r be as in the proof of Lemma 3.4. Let $B'_{n,r}$ be the NFA constructed for $E(L(A'_n), d'_r, r)$ in the proof of Lemma 3.4.[1] For $0 \le k_i \le r+1$, $i = 1, 2, \ldots, n-2$, define

$$u(k_1, k_2, \ldots, k_{n-2}) = a_1 b_1^{k_1} a_2 b_2^{k_2} \cdots a_{n-2} b_{n-2}^{k_{n-2}}.$$

Using the notations of (3) we have $u(k_1, \ldots, k_{n-2}) \cdot a_{n-1} = w(k_1, k_2, \ldots, k_{n-2}, 0, 0)$.

We claim that the strings $u(k_1, \ldots, k_{n-2})$ are all pairwise inequivalent with respect to the Kleene congruence of $\Sigma_n^* E(L(A'_n), d'_r, r)\Sigma_n^*$. Consider two strings $u(k_1, \ldots, k_{n-2})$ and $u(k'_1, \ldots, k'_{n-2})$ where for some $1 \le j \le n-2$, $k_j < k'_j$.

Choose $z = b_j^{r-k_j} a_{j+1} \cdots a_{n-1}$. As in the proof of Lemma 3.2 it is observed that $B'_{n,r}$ has a computation on $u(k_1, \ldots, k_{n-2})$ that reaches state (j, k_j), and a computation started from state (j, k_j) on input z can reach the accepting state (n, r). Thus, $u(k_1, \ldots, k_{n-2}) \cdot z \in L(B'_{n,r}) = E(L(A'_n), d'_r, r)$. We claim that

$$u(k'_1, \ldots, k'_{n-2}) \cdot z \notin \Sigma_n^* E(L(A'_n), d'_r, r)\Sigma_n^*. \tag{5}$$

Note that the string $u(k'_1, \ldots, k'_{n-2}) \cdot z$ contains exactly one occurrence of both a_1 and a_{n-1} and these are, respectively, the first and the last symbol of the string. Since the distance d'_r associates cost $r+1$ to any operation that substitutes, deletes or inserts a symbol a_i, if the negation of (5) holds then the only possibility is that $u(k'_1, \ldots, k'_{n-2}) \cdot z$ must be in $E(L(A'_n), d'_r, r)$. This, in turn, is possible

[1] $B'_{n,r}$ is obtained from the NFA of Fig. 2 by omitting all the transitions on c_i's.

only if the computation of $B'_{n,r}$ on the prefix $u(k'_1, \ldots, k'_{n-2})$ ends in a state of the form (j, x), $0 \le x \le r$. Now Lemma 3.2 implies that the second component x must be at least k'_j and it follows that the computation on the suffix z cannot end in an accepting state. (Lemma 3.2 uses $B_{n,r}$ but the same argument applies here because $B_{n,r}$ equals $B'_{n,r}$ when we omit the c_i-transitions.)

Finally we note that none of the strings $u(k_1, \ldots, k_{n-2})$, $0 \le k_i \le r+1$, is in $\Sigma_n^* E(L(A'_n), d'_r, r)\Sigma_n^*$ and hence they are not equivalent with $a_1 a_2 \cdots a_{n-1}$ which then gives the one additional equivalence class. □

Combining the previous lemmas we can state the main result of this section.

Theorem 4.1. *Let d be an additive quasi-distance on Σ^*. For any n-state NFA A and $r \in \mathbb{N}$ we have*

$$\mathrm{sc}(\Sigma^* \cdot E(L(A), d, r) \cdot \Sigma^*) \le (r+2)^{n-2} + 1).$$

For given $n, r \in \mathbb{N}$ there exists an additive distance d_r and an n-state NFA A defined over an alphabet of size $2n-1$ such that $\mathrm{sc}(\Sigma^* E(L(A), d_r, r)\Sigma^* = (r+2)^{n-2} + 1$.

Proof. The upper bound of Lemma 4.1 is maximized by an NFA with one final state as $(r+2)^{n-2}+1$. The lower bound follows by Lemma 4.2. □

Recall that Brzozowski et al. [3] have shown that, for an n-state DFA language L, the worst case state complexity of the two-sided ideal $\Sigma^* L \Sigma^*$ is $2^{n-2}+1$. This corresponds to the case of having error radius zero ($r = 0$) in Theorem 4.1. Lemma 4.2 requires a linear size alphabet whereas the lower bound for the error free case is obtained with a three letter alphabet [3]. As in Theorem 3.1 in the lower bound result of Lemma 4.2 we can select A to be a DFA if we allow d to be a quasi-distance.

5 Conclusion

We have given a tight lower bound construction for the state complexity of a neighbourhood of a regular language. The construction uses a variable alphabet of size linear in the number of states of the NFA. The main open problem for further work is to develop lower bounds for neighbourhoods of languages over a fixed alphabet. For radius one Hamming neighbourhoods an improved upper bound and a good lower bound using a binary alphabet were given by Povarov [17].

Our lower bound for the approximate pattern matching problem was obtained by modifying the lower bound construction for neighbourhoods of a regular language. This was, roughly speaking, made possible by the choice of the distance function and the language definition where the strings must contain the symbols $a_1, \ldots, a_{n-1}$ in this particular order. Similar constructions will be more challenging if restricted to a fixed alphabet.

References

1. Bordihn, H., Holzer, M., Kutrib, M.: Determination of finite automata accepting subregular languages. Theoretical Computer Science **410**, 3209–3249 (2009)
2. Boyer, R.S., Moore, J.S.: A fast string searching algorithm. Communications of ACM **20**, 762–772 (1977)
3. Brzozowski, J., Jirásková, G., Li, B.: Quotient complexity of ideal languages. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 208–221. Springer, Heidelberg (2010)
4. Calude, C.S., Salomaa, K., Yu, S.: Additive Distances and Quasi-Distances Between Words. Journal of Universal Computer Science **8**(2), 141–152 (2002)
5. Deza, M.M., Deza, E.: Encyclopedia of Distances. Springer-Verlag, Heidelberg (2009)
6. El-Mabrouk, N.: On the size of minimal automata for approximate string matching. Technical report, Institut Gaspard Monge, Université de Marne la Vallée, Paris (1997)
7. Han, Y.-S., Ko, S.-K., Salomaa, K.: The edit distance between a regular language and a context-free language. International Journal of Foundations of Computer Science **24**, 1067–1082 (2013)
8. Kari, L., Konstantinidis, S.: Descriptional complexity of error/edit systems. Journal of Automata, Languages, and Combinatorics **9**, 293–309 (2004)
9. Kari, L., Konstantinidis, S., Kopecki, S., Yang, M.: An efficient algorithm for computing the edit distance of a regular language via input-altering transducers. CoRR abs/1406.1041 (2014)
10. Konstantinidis, S.: Transducers and the properties of error detection, error-correction, and finite-delay decodability. Journal of Universal Computer Science **8**, 278–291 (2002)
11. Konstantinidis, S.: Computing the edit distance of a regular language. Information and Computation **205**, 1307–1316 (2007)
12. Konstantinidis, S., Silva, P.: Maximal error-detecting capabilities of formal languages. J. Automata, Languages and Combinatorics **13**, 55–71 (2008)
13. Konstantinidis, S., Silva, P.: Computing maximal error-detecting capabilities and distances of regular languages. Fundamenta Informaticae **101**, 257–270 (2010)
14. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady **10**(8), 707–710 (1966)
15. Ng, T., Rappaport, D., Salomaa, K.: Quasi-distances and weighted finite automata. In: Shallit, J., Okhotin, A. (eds.) DCFS 2015. LNCS, vol. 9118, pp. 209–219. Springer, Heidelberg (2015)
16. Pighizzini, G.: How hard is computing the edit distance? Information and Computation **165**, 1–13 (2001)
17. Povarov, G.: Descriptive complexity of the hamming neighborhood of a regular language. In: Language and Automata Theory and Applications, pp. 509–520 (2007)
18. Salomaa, K., Schofield, P.: State Complexity of Additive Weighted Finite Automata. International Journal of Foundations of Computer Science **18**(06), 1407–1416 (2007)
19. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press (2009)
20. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (Eds.) Handbook of Formal Languages, vol. I, pp. 41–110. Springer (1997)