State Complexity of Prefix Distance

Timothy Ng, David Rappaport, and Kai Salomaa

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

Abstract. The prefix distance between strings x and y is the number of symbol occurrences in the strings that do not belong to the longest common prefix of x and y. The suffix and the substring distance are defined analogously in terms of the longest common suffix and longest common substring, respectively, of two strings. We show that the set of strings within prefix distance k from an n state DFA (deterministic finite automaton) language can be recognized by a DFA with $(k+1)\cdot n - \frac{k(k+1)}{2}$ states and this number of states is needed in the worst case. Also we give tight bounds for the nondeterministic state complexity of the set of strings within prefix, suffix or substring distance k from a regular language.

1 Introduction

Various similarity measures between strings and languages have been considered for information transmission applications. The edit distance counts the number of substitution, insertion and deletion operations that are needed to transform one string to another. The Hamming distance counts the number of positions in which two equal length strings differ. A distance measure between words can be extended in various ways as a distance between sets of strings (or languages) [3, 4] and algorithms for computing the distance between languages are important for error-detection and error-correction applications [4, 9, 10]. The descriptional complexity of error/edit systems has been considered by Kari and Konstantinidis [8]. Other types of sequence similarity measures have been considered e.g. by Apostolico [1].

Instead of counting the number of edit operations, the similarity of strings can be defined by way of their longest common prefix, suffix, or substring, respectively [4]. For example, the prefix distance of strings x and y is the sum of the length of the suffix of x and the suffix of y that occurs after their longest common prefix. A parameterized prefix distance between regular languages has been considered by Kutrib et al. [11] for estimating the fault tolerance of information transmission applications.

The neighbourhood of radius k of a language L consists of all strings that are within distance k from some string in L. Calude et al. [3] have shown that the neighbourhood of a regular language with respect to an additive distance is regular. A distance is said to be additive if it, in a certain sense, respects string concatenation. This gives rise to the question how large is the (non)deterministic

finite automaton (DFA, respectively, NFA) needed to recognize the neighbourhood of a regular language, that is, what is the state complexity of neighbourhoods of regular languages.

Povarov [15] has given an improved upper bound and a closely matching lower bound for the state complexity of Hamming neighbourhoods of radius one. Upper bounds for the state complexity of neighbourhoods with respect to an additive distance or quasi-distance have been obtained by the authors [14, 16] using a construction based on weighted finite automata.

It follows from Choffrut and Pighizzini [4] that the prefix, suffix and substring distance preserve regularity, that is, the neighbourhood of a regular language of finite radius remains regular. Here we study the state complexity of these neighbourhoods. We show that if L is recognized by a deterministic finite automaton (DFA) of size n, the prefix neighbourhood of L of radius $k < n$ has a DFA of size $(k+1) \cdot n - \frac{k(k+1)}{2}$ and that this bound cannot be improved in the worst case. Our lower bound construction uses an alphabet of size $n+1$ and we show that the general upper bound cannot be reached using languages defined over a fixed alphabet.

We consider also the nondeterministic state complexity of prefix, suffix and substring neighbourhoods. If L has a nondeterministic finite automaton (NFA) of size n, the neighbourhood of L of radius k can be recognized by an NFA of size $n+k$. The upper bound for the substring neighbourhood of L of radius k is $(k+1) \cdot n + 2k$. In all cases we give matching lower bounds for nondeterministic state complexity, and in the lower bound constructions L has, in fact, a DFA of size n.

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 [17] or the survey by Yu [18]. A survey of distances is given by Deza and Deza [5]. Recent surveys on descriptional complexity of regular languages include [6, 7, 12].

In the following Σ is always a finite alphabet, the set of strings of Σ is Σ^* and ε is the empty string. The reversal of a string $x \in \Sigma^*$ is x^R. The set of nonnegative integers is $\mathbb{N}_0$. The cardinality of a finite set S is denoted $|S|$ and the powerset of S is 2^S. A string $w \in \Sigma^*$ is a *substring* or *factor* of x if there exist strings $u, v \in \Sigma^*$ such that $x = uwv$. If $u = \varepsilon$, then w is a *prefix* of x. If $v = \varepsilon$, then w is a *suffix* of x.

A *nondeterministic finite automaton* (NFA) is a 5-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 $\delta : Q \times \Sigma \to 2^Q$, $Q_0 \subseteq Q$ is a set of initial states, 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 string $w \in \Sigma^*$ is *accepted* by A if, for some $q_0 \in Q_0$, $\delta(q_0, w) \cap F \neq \emptyset$ and the language recognized by A consists of all strings accepted by A. an ε-NFA is an extension of an NFA where transitions can be labeled by the empty string

ε [17, 18], i.e., δ is a function $Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$. It is known that every ε-NFA has an equivalent NFA without ε-transitions and with the same number of states. An NFA $A = (Q, \Sigma, \delta, Q_0, F)$ is a *deterministic finite automaton* (DFA) if $|Q_0| = 1$ and, for all $q \in Q$ and $a \in \Sigma$, $\delta(q, a)$ either consists of one state or is undefined. 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 DFA A is *minimal* if each state $q \in Q$ is reachable from the initial state and no two states are equivalent.

Note that our definition of a DFA allows some transitions to be undefined, that is, by a DFA we mean an incomplete DFA. It is well known that, for a regular language L, the sizes of the minimal incomplete and complete DFAs differ by at most one. The constructions in Section 3 are more convenient to formulate using incomplete DFAs but our results would not change in any significant way if we were to require that all DFAs are complete.

The (incomplete deterministic) *state complexity* of a regular language L, $\mathrm{sc}(L)$, is the size of the minimal DFA recognizing L. The *nondeterministic state complexity* of L, $\mathrm{nsc}(L)$, is the size of the minimal NFA recognizing L. The minimal NFA recognizing a regular language need not be unique. A common way of establishing lower bounds for nondeterministic state complexity relies on fooling sets.

Definition 1. *A set of pair of strings $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, $x_i, y_i \in \Sigma^*$, $i = 1, \ldots, m$, is a* fooling set *for a language L if $x_i y_i \in L$, $i = 1, \ldots, m$ and, for all $1 \leq i < j \leq m$, $x_i y_j \notin L$ or $x_j y_i \notin L$.*

Proposition 1 ([2, 7]). *If L has a fooling set S then* $\mathrm{nsc}(L) \geq |S|$.

To conclude this section, we recall definitions of the distance measures used in the following. Generally, 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 *neighbourhood* of a language L of radius k with respect to a distance d is the set

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

Let $x, y \in \Sigma^*$. The *prefix distance* of x and y counts the number of symbols which do not belong to the longest common prefix of x and y [4]. It is defined by

$$d_p(x, y) = |x| + |y| - 2 \cdot \max_{z \in \Sigma^*} \{|z| \mid x, y \in z\Sigma^*\}.$$

Similarly, the *suffix distance* of x and y counts the number of symbols which do not belong to the longest common suffix of x and y and is defined

$$d_s(x, y) = |x| + |y| - 2 \cdot \max_{z \in \Sigma^*} \{|z| \mid x, y \in \Sigma^* z\}.$$

The *substring distance* measures the similarity of x and y based on their longest common continuous substring (or factor) and is defined

$$d_f(x, y) = |x| + |y| - 2 \cdot \max_{z \in \Sigma^*} \{|z| \mid x, y \in \Sigma^* z \Sigma^*\}.$$

The paper [4] refers to d_f as the *subword distance*. The term "subword distance" has been used also for a distance defined in terms of the longest common noncontinuous subword [13].

3 State Complexity of Prefix Neighbourhoods

In this section we consider the deterministic state complexity of prefix neighbourhoods. We construct a DFA for the neighbourhood of radius k with respect to the prefix distance d_p. After that we show that the construction is optimal by giving a matching lower bound. The lower bound construction uses an alphabet of size $n+1$ where n is the number of states of the DFA. We show that the upper bound cannot be reached by languages defined over a constant size alphabet.

Proposition 2. *Let $n > k \geq 0$ and L be a regular language recognized by a DFA with n states. Then there is a DFA recognizing $E(L, d_p, k)$ with at most $n \cdot (k+1) - \frac{k(k+1)}{2}$ states.*

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be the DFA that recognizes L. We define the function $\varphi : Q \to \mathbb{N}_0$ by

$$\varphi(q) = \min_{w \in \Sigma^*} \{|w| \mid \delta(q, w) \in F\}.$$

The function $\varphi(q)$ gives the length of the shortest path from a state q to the closest, or next, reachable final state. Note that under this definition, if $q \in F$, then $\varphi(q) = 0$.

We construct a DFA $A' = (Q', \Sigma, \delta', q_0', F')$ that recognizes the neighbourhood $E(L, d_p, k)$. We define the state set

$$Q' = ((Q - F) \times \{1, \ldots, k+1\}) \cup F \cup \{p_1, \ldots, p_k\}.$$

Note that some states of Q' are always unreachable and at the end of the proof we calculate an upper bound for the number of reachable states. The initial state q_0' is defined

$$q_0' = \begin{cases} q_0, & \text{if } q_0 \in F; \\ (q_0, \varphi(q_0)) & \text{if } q_0 \notin F \text{ and } \varphi(q_0) \leq k; \\ (q_0, k+1) & \text{if } q_0 \notin F \text{ and } \varphi(q_0) > k. \end{cases}$$

The set of final states is given by

$$F' = ((Q - F) \times \{1, \ldots, k\}) \cup F \cup \{p_1, \ldots, p_k\}.$$

Let $q_{i,a} = \delta(i, a)$ for $i \in Q$ and $a \in \Sigma$, if $\delta(i, a)$ is defined. Then for all $a \in \Sigma$, the transition function δ' is defined for states $i \in F$ by

$$\delta'(i, a) = \begin{cases} (q_{i,a}, 1), & \text{if } q_{i,a} \in Q - F; \\ q_{i,a}, & \text{if } q_{i,a} \in F; \\ p_1, & \text{if } \delta(i, a) \text{ is undefined.} \end{cases}$$

For states $(i,j) \in Q - F \times \{1,\ldots,k+1\}$, δ' is defined

$$\delta'((i,j),a) = \begin{cases} q_{i,a}, & \text{if } q_{i,a} \in F; \\ (q_{i,a}, \min\{j+1, \varphi(q_{i,a})\}), & \text{if } \varphi(q_{i,a}) \text{ or } j+1 \leq k; \\ (q_{i,a}, k+1), & \text{if } \varphi(q_{i,a}) \text{ and } j+1 > k; \\ p_{j+1}, & \text{if } \delta(i,a) \text{ is undefined.} \end{cases}$$

Finally, we define δ' for states p_ℓ for $\ell = 1,\ldots,k-1$ by $\delta'(p_\ell, a) = p_{\ell+1}$. The machine A' has three types of states. The first type consists of final states of A. The second type are new states p_ℓ, which form a chain of error states. When a transition that was undefined in A is encountered during some computation, A' is taken to the chain of error states p_i. The third type of states consists of states of A which are not final states and are paired with a counter. For a state (i,j), the counter component j keeps track of the distance of the current computation to the closest final state of A.

On input $w \in \Sigma^*$, there are three cases to consider. Let $x \in L$ be a closest string to w according to the prefix distance d_p.

1. First, suppose that $x = wx'$ for some $x' \in \Sigma^*$. Then $w \in E(L, d_p, k)$ if and only if $|x'| \leq k$. Consider the computation on w, which must end in some state (i,j). Otherwise, the computation either ends in a final state, in which case $x = w$, or it ends in some state p_ℓ, which cannot be the case as w is a proper prefix of a word in L. Since x is the closest word in L to w, there must be a shortest path of length $|x'|$ in the original DFA A from state i to a final state of A. By definition, (i,j) is a final state if $j = \varphi(i) \leq k$. Thus, $j = \varphi(i) = |x'|$ and (i,j) is a final state if and only if $j = |x'| \leq k$.
2. Next, suppose that $w = xw'$ for some $w' \in \Sigma^*$. In this case, $w \in E(L, d_p, k)$ if $|w'| \leq k$. The machine reaches some final state f of A once it reads all of x. Then the machine continues reading w' until it reaches some state $q \in Q'$. The state q is either a state (i,j) or a state p_ℓ, since otherwise, $q \in F$ and $w' = \varepsilon$.
   (a) Consider $q = (i,j)$. By definition, (i,j) is a final state if $j \leq k$. Since x is a closest word in L to w, $j = |w'|$ must be the distance of the current computation from the closest final state f unless $|w'| > k$, in which case $j = k+1$. Otherwise, there was some state (i',j') that was encountered during the computation of w' with a final state f' that was closer than f. Thus, if $|w'| > k$, then $j = k+1$ and (i,j) is not a final state. Otherwise, $j = |w'| \leq k$ and (i,j) is a final state.
   (b) Now consider when $q \neq (i,j)$ and let $w' = w'_1 w'_2$. The computation from f on w'_1 reaches some state $q' = (i',j')$ for which there is no transition in A defined for the first symbol of w'_2. By the same reasoning as above, $j' = |w'_1| < k$. Since an undefined transition was encountered on the first symbol of w'_2, the machine goes to state $p_{|w'_1|+1}$. From state $p_{|w'_1|+1}$, the machine reads the rest of w'_2. Now, if $|w'| > k$, then $|w'_2| > k - |w'_1|$ and the computation on the rest of w'_2 fails when it reaches p_k and there are

no further transitions. Otherwise, $|w'| \leq k$ and the computation of w'_2 ends in a state $p_{|k'_1|+|k'_2|}$, which is a final state since $|w'| = |w'_1|+|w'_2| \leq k$.

3. Finally, suppose that $w = pw'$ and $x = px'$ with $p, w', x' \in \Sigma^*$ such that p is the longest common prefix of w and x. Note that if $w' = \varepsilon$, then it becomes Case 1, and if $x' = \varepsilon$, then Case 2 applies. Thus $w \in E(L, d_p, k)$ if and only if $|w'| + |x'| \leq k$. In this case, A' reads w until it reaches a state (i_p, j_p) on the prefix p. At this point, reading x' from (i_p, j_p) will take the machine to some final state f, while reading w' from (i_p, j_p) takes the machine to some other state $q \in Q'$. Note that $|x'| \leq k$, since otherwise $|x'| + |w'| > k$, and $j_p = \varphi(i_p) = |x'|$, since otherwise x would not be a closest word to w. Now, q is either of the form (i, j) or a state p_ℓ.
   (a) Suppose q is of the form (i, j). Then j is either $|w'| + |x'|$ or $k + 1$. If $|w'| > k - |x'|$, then $j = k + 1$ and (i, j) is not a final state. If $j \leq |w'| + |x'|$, then there must be some final state f' closer to a state on the computation path of w' from (i_p, j_p) which cannot be the case if x is a closest word to w. Thus, $j = |w'| + |x'| \leq k$ and (i, j) is a final state.
   (b) Now, suppose $q \neq (i, j)$ and let $w' = w'_1 w'_2$. The computation from (i_p, j_p) on w'_1 reaches some state $q' = (i', j')$ for which there is no transition in A on the first symbol of w'_2. By the same reasoning as above, $j' = |w'_1| < k - |x'|$. Since an undefined transition was encountered on the first symbol of w'_2, the machine goes to state $p_{|w'_1|+|x'|+1}$. From state $p_{|w'_1|+|x'|+1}$, the rest of w'_2 is read. If $|w'_2| > k - (|w'_1| + |x'|)$, then the computation of w'_2 falls off at p_k. Otherwise, the computation ends in state $p_{|w'_2|+|w'_1|+|x'|}$. We have

$$|w'_2| + |w'_1| + |x'| = |w'| + |x'| \leq k$$

   and thus, $p_{|w'_2|+|w'_1|+|x'|}$ is a final state.

The set of states Q' has $(n - f) \cdot (k + 1) + k + f$ elements but they cannot all be reachable. Based on the definition of the transitions of δ' we observe that if there is a transition entering a state (q, j), $q \in Q - F$, $1 \leq j \leq k + 1$, then $\varphi(q)$ must be at least j. Thus, all elements of the set

$$S_{ur} = \{(q, j) \mid q \in Q - F, 1 \leq j \leq k + 1,\ j > \varphi(q)\}$$

are unreachable as states of A'. Since increasing the number of final states of A by one decreases the cardinality of Q' by k and decreases the cardinality of S_{ur} by at most k, it is clear that an upper bound for the cardinality of the set of potentially reachable states $Q' - S_{ur}$ is obtained by choosing $f = 1$. Using the observation that all useful states must reach a final state, in the case when $F = \{q_f\}$ is a singleton set, the cardinality of S_{ur} is minimized when, in the DFA A for each $1 \leq i \leq k$, exactly one non-final state q_i has a shortest path of length i that reaches q_f. In this case $S_{ur} = \{(q_i, j) \mid i < j \leq k + 1,\ i = 1, \ldots, k\}$ and $|S_{ur}| = \frac{k(k+1)}{2}$.

We have verified that at most $n \cdot (k+1) - \frac{k(k+1)}{2}$ states of A' can be reachable. □

The lower bound construction that we present uses an alphabet with variable size. We will show later that it is impossible to reach the upper bound (for all n) with an alphabet of fixed size.

Lemma 1. *For $n > k \in \mathbb{N}$, there exists a DFA A_n with n states over an alphabet of size $n+1$ such that*

$$\mathrm{sc}(E(L(A_n), d_p, k) \geq n \cdot (k+1) - \frac{k(k+1)}{2}.$$

Proof. We define a DFA $A_n = (Q_n, \Sigma_n, \delta_n, q_0, F)$ (Figure 1) by choosing

$$Q_n = \{0, \ldots, n-1\}, \quad \Sigma_n = \{a, b, c_1, \ldots, c_{n-1}\},$$

$q_0 = 0$, $F = \{0\}$, and the transition function is given by

- $\delta_n(q, a) = q$ for all $q \in Q_n$,
- $\delta_n(q, b) = q + 1 \bmod n$ for $q = 1, \ldots, n-1$,
- $\delta_n(0, c_i) = i$ for $i = 1, \ldots, n-1$,

Note that for every state $q \in Q_n$, we have $\varphi(q) = n - q$.

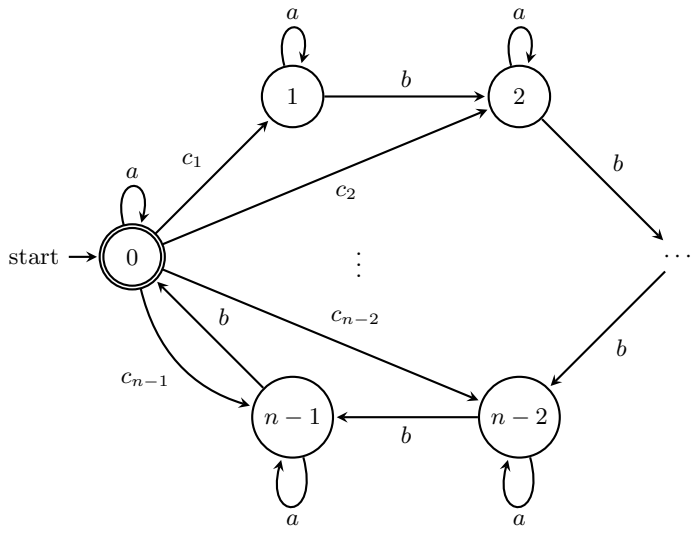

Fig. 1. The DFA A_n.

We transform A_n into the DFA $A'_n = (Q'_n, \Sigma_n, \delta'_n, q'_0, F')$ by following the construction from Proposition 2. To determine the reachable states of Q'_n, we first consider states of the form $(i, j) \in (Q_n - \{q_0\}) \times \{1, \ldots, k+1\}$. For states $i \in Q_n - \{q_0\}$ with $\varphi(i) > k$, we can reach state (i, j) via the word $c_i a^j$ for

$j = 1, \ldots, k+1$. For states $i \in Q_n - \{q_0\}$ with $\varphi(i) \leq k$, we can reach state (i, j) via the word $c_i a^j$ for $j = 1, \ldots, \varphi(i)$. However, states (i, j) with $j > \varphi(i)$ are unreachable by definition of A'_n. Thus the number of unreachable states in $(Q_n - \{q_0\}) \times \{1, \ldots, k+1\}$ is

$$\sum_{i=n-k}^{n-1} |\{i\} \times \{\varphi(i)+1, \ldots, k+1\}| = \sum_{i=1}^{k} |\{i+1, \ldots, k+1\}| = \sum_{i=1}^{k} i = \frac{k(k+1)}{2}.$$

Now consider states $p_1, \ldots, p_k$. The state p_ℓ is reachable on the word b^ℓ. Finally, 0 is reachable since it is the initial state. Thus, the number of reachable states is

$$(n-1) \cdot (k+1) - \frac{k(k+1)}{2} + k + 1 = n \cdot (k+1) - \frac{k(k+1)}{2}.$$

Now, we show that all reachable states are pairwise inequivalent. First, note that 0 can be distinguished from any other state by the word ε. Next, we distinguish states of the form (i, j) from states of the form p_ℓ via the word $a^k b^{n-i}$. From state (i, j), reading a^k takes the machine to state $(i, \min\{\varphi(i), k+1\})$. Subsequently reading b^{n-i} takes the machine to the final state 0. However, for every state p_ℓ, reading a^k forces the machine beyond state p_k, after which there are no transitions defined.

Next, without loss of generality, we let $\ell < \ell'$ and consider states p_ℓ and $p_{\ell'}$. From above, the state p_ℓ can be reached by a word b^ℓ and $p_{\ell'}$ is reached by a word $b^{\ell'}$. Choose $z = a^{k-\ell}$. The string z takes state p_ℓ to the state p_k, where it is accepted. However, the computation on string z from state $p_{\ell'}$ is undefined since $\ell' + k - \ell > k$.

Finally, we consider states of the form (i, j). Let $i < i'$ and consider states (i, j) and (i', j'). Recall that (i, j) can be reached by a word $c_i a^j$ and (i', j') is reached by a word $c_{i'} a^{j'}$. Let $z = b^{n-i+k}$. From state (i, j), the word z goes to state 0 on b^{n-i}. Then by reading b^k from state 0, we reach state p_k and thus, $c_i a^j \cdot z \in E(L(A_n), d_p, k)$. However, when reading z from state (i', j'), we reach state 0 on $b^{n-i'}$, since $i' > i$. We are then left with $b^{i'-i+k}$. Reading b^k takes us to state p_k, where we still have $b^{i'-i}$ and no further defined transitions. Thus, $c_{i'} a^{j'} \cdot z \notin E(L(A_n), d_p, k)$.

Next, we fix i and let $j < j'$. The state (i, j) is reachable by the word $c_i a^j$ and (i, j') is reachable by $c_i a^{j'}$. First, consider the case when $\varphi(i) > k$. Then let $z = a^k - j$. Reading z from (i, j) takes us to state (i, k), which is a final state, so we have $c_i a^j \cdot z \in E(L(A_n), d_p, k)$. However, from (i, j'), reading z brings us to state $(i, k+1)$ and we have $c_i a^{j'} \cdot z \notin E(L(A_n), d_p, k)$.

Now, consider the case when $\varphi(i) \leq k$. Let $z = c_i a^{k-j-1}$. From state (i, j), reading c_i takes the machine to state p_{j+1} and reading a^{k-j-1} puts the machine in state p_k. Thus, $c_i a^j \cdot z \in E(L(A_n), d_p, k)$. From (i, j'), reading z takes us to state p_k with $a^{j'-j}$ still unread since $j' + k - j - 1 > k$ and thus with no further transitions available, we have $c_i a^{j'} \cdot z \notin E(L(A_n), d_p, k)$.

Thus, we have shown that there are $n \cdot (k+1) - \frac{k(k+1)}{2}$ reachable states and that all reachable states are pairwise inequivalent. □

Taking Proposition 2 together with Lemma 1, we get the following theorem.

Theorem 1. *For $n > k \geq 0$, if $\mathrm{sc}(L) = n$ then*

$$\mathrm{sc}(E(L, d_p, k)) \leq n \cdot (k+1) - \frac{k(k+1)}{2}$$

and this bound can be reached in the worst case.

The proof of Lemma 1 uses an alphabet of size $n+1$. To conclude this section we observe that the general upper bound cannot be reached by languages defined over a fixed alphabet.

Proposition 3. *Let A be a DFA with n states. If the state complexity of $E(L(A), d_p, n)$ equals $n \cdot (k+1) - \frac{k(k+1)}{2}$, then the alphabet of A needs at least $n-1$ letters.*

4 Nondeterministic State Complexity

We consider the nondeterministic state complexity of neighbourhoods of a regular language with respect to the prefix-, the suffix- and the substring distance, respectively.

4.1 Prefix and Suffix Distance

We consider first neighbourhoods with respect to the prefix distance, and the results for the suffix distance are obtained as a consequence of the fact that the nondeterministic state complexity of a regular language L is the same as the nondeterministic state complexity of the reversal of L and using the observation $d_s(x, y) = d_p(x^R, y^R)$ for all strings x and y.

We give an upper bound for the nondeterministic state complexity of the neighbourhood of radius k with respect to the prefix distance d_p and give a matching lower bound construction.

Proposition 4. *Let $k \geq 0$ and L be a regular language recognized by an NFA with n states. Then there is an NFA recognizing $E(L, d_p, k)$ with at most $n + k$ states.*

Proof. Let $A = (Q, \Sigma, \delta, Q_0, F)$ be the NFA recognizing L. We define an NFA $A' = (Q', \Sigma, \delta', I, F)$ for the language $E(L, d_p, k)$ by

- $Q' = Q \cup \{p_1, \ldots, p_k\}$, $I = Q_0$,
- $F' = F \cup \{p_1, \ldots, p_k\} \cup \{q \in Q \mid \varphi(q) \leq k\}$.

Recall that for $q \in Q$, $\varphi(q)$ denotes the length of the shortest string that takes q to a final state. The transition function is defined for all $a \in \Sigma$ by

- $\delta'(q, a) = \delta(q, a) \cup \{p_1\}$ for all $q \in F$,
- $\delta'(q, a) = \delta(q, a) \cup \{p_{\varphi(q)+1}\}$ for all $q \in Q$ with $\varphi(q) < k$,

– $\delta'(p_i, a) = p_{i+1}$ for $i = 1, \ldots, k-1$.

□

Using the fooling sets of Proposition 1 we get a matching lower bound.

Lemma 2. *For $n, k \in \mathbb{N}$, there exists a DFA A with n states over $\Sigma = \{a, b\}$ such that any NFA for $E(L(A), d_p, k)$ requires $n + k$ states.*

Theorem 2. *For a regular language $L \subseteq \Sigma^*$ recognized by an NFA with n states and an integer $k \geq 0$,*

$$\mathrm{nsc}(E(L, d_p, k)) \leq n + k.$$

There exists a DFA A with n states such that for all $k \geq 0$,

$$\mathrm{nsc}(E(L(A), d_p, k)) = n + k.$$

We get the results for the suffix distance neighbourhoods as a corollary of Theorem 2 and the observation that, for all strings x and y, $d_s(x, y) = d_p(x^R, y^R)$.

Corollary 1. *Let $k \geq 0$ and L be a regular language recognized by a DFA with n states. Then there is an NFA recognizing $E(L, d_s, k)$ with at most $n + k$ states.*

The following lemma is a symmetric variant of the lower bound construction for prefix distance neighbourhoods. As a consequence of Corollary 1 and Lemma 3 we then get a tight bound for the nondeterministic state complexity of suffix neighbourhoods.

Lemma 3. *For $n, k \in \mathbb{N}$, there exists a DFA A with n states over $\Sigma = \{a, b\}$ such that any NFA for $E(L(A), d_s, k)$ requires $n + k$ states.*

Theorem 3. *For a regular language $L \subseteq \Sigma^*$ recognized by an NFA with n states and an integer $k \geq 0$,*

$$\mathrm{nsc}(E(L, d_s, k)) \leq n + k.$$

There exists a DFA A with n states such that for all $k \geq 0$,

$$\mathrm{nsc}(E(L(A), d_s, k)) = n + k.$$

4.2 Substring Distance Neighbourhoods

A neighbourhood with respect to the substring distance can be recognized by an NFA that, roughly speaking, makes $k + 1$ copies of the NFA A recognizing the original language. Later we will show that the construction is optimal.

Lemma 4. *If A is an n-state NFA and $k \in \mathbb{N}_0$, the neighbourhood $E(L(A), d_f, k)$ can be recognized by an NFA with $(k + 1) \cdot n + 2k$ states.*

Proof sketch. Combining the constructions used for Proposition 4 and Corollary 1, an NFA B for the language $E(L(A), d_f, k)$ uses a chain of k states both at the beginning and at the end of the computation to keep track of the length of the nonmatching prefixes (respectively, suffixes) of the input and a word of $L(A)$. After processing the prefixes, the NFA B has to "remember" the sum of the lengths of the nonmatching prefixes (which can be up to k), and for this reason B is equipped with $k+1$ copies of the original NFA A. □

Although both the upper bound and the construction used in the proof of Lemma 4 differ significantly from the corresponding bound and construction for prefix distance (or suffix distance) neighbourhoods, it turns out that for the lower bound, we can use the same cyclic languages.

Lemma 5. *There exists a DFA A with n states such that, for all $k \geq 0$,*

$$\mathrm{nsc}(E(L(A), d_f, k)) \geq (k+1) \cdot n + 2k.$$

Proof sketch. By choosing $\Sigma = \{a, b\}$ and $L = (a^n)^*$, the minimal incomplete DFA for L has n states. Define

$$S_1 = \{(b^\ell a^i,\ a^{n-i} b^{k-\ell}) \mid 0 \leq i \leq n-1,\ 0 \leq \ell \leq k\},$$

$$S_2 = \{(a^n b^j,\ b^{k-j}) \mid 1 \leq j \leq k\},\quad S_3 = \{(b^j,\ b^{k-j} a^n) \mid 1 \leq j \leq k\}.$$

When $k \leq n$, it can be verified that $S_1 \cup S_2 \cup S_3$ is a fooling set for $E(L, d_f, k)$. When $n < k$, we can modify the definition of S_2 and S_3 to construct a fooling set of cardinality $(k+1) \cdot n + 2k$ for $E(L, d_f, k)$. □

As a consequence of Lemmas 4 and 5 we have an exact bound for the nondeterministic state complexity of neighbourhoods with respect to the substring distance:

Theorem 4. *If L has an NFA with n states and $k \in \mathbb{N}_0$,*

$$\mathrm{nsc}(E(L, d_f, k)) \leq (k+1) \cdot n + 2k.$$

For every $n \in \mathbb{N}$ there exists a DFA A with n states such that for all $k \in \mathbb{N}_0$, $\mathrm{nsc}(E(L(A), d_f, k)) = (k+1) \cdot n + 2k$.

5 Conclusion

We have given a tight bound for the deterministic state complexity of neighbourhoods with respect to the prefix distance and tight bounds for the nondeterministic state complexity of the prefix, suffix and substring distance neighbourhoods.

Due to the fact that the reversal of a regular language L can be recognized by an NFA having the same size as an NFA for L, the bounds for the nondeterministic state complexity of suffix neighbourhoods were obtained as a corollary of the corresponding bounds for prefix neighbourhoods. The situation is essentially different for DFAs since, for a DFA A with n states, the incomplete DFA recognizing $L(A)^R$ needs in the worst case $2^n - 1$ states. Obtaining tight bounds for the deterministic state complexity of neighbourhoods with respect to the suffix distance, or the substring distance, remains an open problem.

References

1. Apostolico, A.: Maximal Words in Sequence Comparisons Based on Subword Composition. In: Elomaa, T. et al. (Eds.), Ukkonen Festschrift, Lect. Notes Comput. Sci. 6060 (2010) 34–44
2. Birget, J.C.: Intersection and union of regular languages and state complexity. Information Processing Letters **43** (1992) 185–190
3. Calude, C.S., Salomaa, K., Yu, S.: Additive Distances and Quasi-Distances Between Words. Journal of Universal Computer Science **8**(2) (2002) 141–152
4. Choffrut, C., Pighizzini, G.: Distances between languages and reflexivity of relations. Theoretical Computer Science **286**(1) (2002) 117–138
5. Deza, M.M., Deza, E.: Encyclopedia of Distances. Springer Berlin Heidelberg (2009)
6. Gao, Y., Moreira, N., Reis, R., Yu, S.: A review on state complexity of individual operations. Faculdade de Ciencias, Universidade do Porto, Technical Report DCC-2011-8 `www.dcc.fc.up.pt/dcc/Pubs/TReports/TR11/dcc-2011-08.pdf` To appear in *Computer Science Review.*
7. Holzer, M. Kutrib, M.: Descriptional and computational complexity of finite automata — A survey. *Inform. Comput.* **209** (2011) 456–470.
8. Kari, L., Konstantinidis, S.: Descriptional complexity of error/edit systems. Journal of Automata, Languages, and Combinatorics **9** (2004) 293–309
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.: Computing the edit distance of a regular language. Information and Computation **205** (2007) 1307–1316
11. Kutrib, M., Meckel, K., Wendlandt, M.: Parameterized Prefix Distance between Regular Languages. In: SOFSEM 2014: Theory and Practice of Computer Science. Lect. Notes Comput. Sci. **8327** (2014), Springer, 419–430
12. Kutrib, M., Pighizzini, G.: Recent trends in descriptional complexity of formal languages. *Bulletin of the EATCS* 111 (2013) 70–86.
13. Lothaire, M.: Applied Combinatorics on Words, Ch. 1 Algorithms on Words. Encyclopedia of Mathematics and It's Applications 105, Cambridge University Press, New York, 2005
14. Ng, T., Rappaport, D., Salomaa, K.: Quasi-Distances and Weighted Finite Automata. In: Descriptional Complexity of Formal Systems, DCFS'15, Waterloo, Ontario, June 25–27, 2015, Lect. Notes Comput. Sci. **9118** (2015) (to appear)
15. Povarov, G.: Descriptive Complexity of the Hamming Neighborhood of a Regular Language. In: Language and Automata Theory and Applications. (2007) 509–520
16. Salomaa, K., Schofield, P.: State Complexity of Additive Weighted Finite Automata. International Journal of Foundations of Computer Science **18**(06) (December 2007) 1407–1416
17. Shallit, J.: A second course in formal languages and automata theory. Cambridge University Press, Cambridge, MA (2009)
18. Yu, S.: Regular languages. In Rozenberg, G., Salomaa, A., eds.: Handbook of Formal Languages. Springer-Verlag, Berlin, Heidelberg (1997) 41–110