Quasi-Distances and Weighted Finite Automata

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. We show that the neighbourhood of a regular language L with respect to an additive quasi-distance can be recognized by an additive weighted finite automaton (WFA). The size of the WFA is the same as the size of an NFA (nondeterministic finite automaton) for L and the construction gives an upper bound for the state complexity of a neighbourhood of a regular language with respect to a quasi-distance. We give a tight lower bound construction for the determinization of an additive WFA using an alphabet of size five. The previously known lower bound construction needed an alphabet that is linear in the number of states of the WFA.

Keywords: regular languages, weighted finite automata, state complexity, distance measures

1 Introduction

In many applications it is crucial to measure the similarity between data. How we define the distance between objects depends on what the objects we want to compare are and why we want to compare them [5]. One of the most commonly used similarity measures for words is the Levenshtein distance [13], also called the edit distance [4, 11, 12, 15]. By the edit distance between languages L_1 and L_2 we mean the smallest distance between a word of L_1 and of L_2, respectively. This definition is natural for error correction applications; however, other definitions such as the relative distance or Hausdorff distance have also been considered [3, 5].

The edit distance is additive with respect to concatenation of words in the sense defined by Calude et al. [2]. Pighizzini [15] has shown that the edit distance between a word and a language recognized by a one-way nondeterministic auxiliary pushdown automaton is computable in polynomial time. Konstantinidis [12] showed that the edit distance of a regular language, that is, the smallest edit distance between two distinct words in the language can be computed in polynomial time. Han et al. [8] gave a polynomial time algorithm to compute the edit distance between a regular language and a context-free language. Error/edit systems for error correction have been studied by Kari and Konstantinidis [10], and the error correction capabilities of regular languages with respect to edit operations were recently investigated by Benedikt et al. [1].

A quasi-distance is a generalization of the notion of distance in that it allows the possibility of distinct elements having distance zero. Calude et al. [2] showed

that the neighbourhood of a regular language with respect to an additive distance or quasi-distance is regular. The neighbourhood of radius r of a language L consists of all words that have distance at most r from some word of L.

In an additive weighted finite automaton (WFA) [17] the weight of a path is the sum of the weights of the individual transitions that make up the path and the weight of an accepted word w is the minimum weight of a path from the start state to a final state that spells out w. Note that this differs significantly from weighted automata used, for example, in image processing applications [6, 7].

For a given nondeterministic finite automaton (NFA) A, an additive distance d and radius r, Salomaa and Schofield [17] gave a construction for an additive weighted finite automaton (WFA) which recognizes the neighbourhood of radius r of the language recognized by A. The construction relies on the fact that additive distances are finite, that is, the neighbourhood of any word is always finite. This makes the construction not suitable for quasi-distances, since neighbourhoods of additive quasi-distances are not guaranteed to be finite [2].

Here we show that neighbourhoods of a regular language with respect to an additive quasi-distance can be recognized by a WFA. Given an NFA A, the WFA recognizing a constant radius neighbourhood of $L(A)$ can be constructed in polynomial time. The construction relies on the property that the neighbourhoods with respect to a quasi-distance are regular and a finite automaton for the neighbourhood can be constructed effectively. The construction yields also an upper bound for the size of a deterministic finite automaton (DFA) needed to recognize the neighbourhood of radius r of a regular language (given by an NFA) with respect to a quasi-distance. The upper bound is significantly better than the bound obtained by constructing an NFA for the neighbourhood [2] and then determinizing the NFA.

We study also the state complexity of additive WFAs. A WFA A within a given weight bound R recognizes a regular language, and Salomaa and Schofield [17] gave an upper bound for the size of a DFA for this language. They also gave a matching lower bound construction; however, the WFAs used for the lower bound construction needed an alphabet of size linear in the number of states of the WFA. Here we give a tight lower bound construction for the "determinization of WFAs" using a five-letter alphabet.

The paper concludes with a discussion of open problems on the state complexity of neighbourhoods of a regular language with respect to an additive distance or quasi-distance.

2 Preliminaries

We assume that the reader is familiar with the basics of finite automata and regular languages [9, 19, 20]. A general reference for weighted finite automata is [6].

In the following Σ is always a finite alphabet, Σ^* is the set of words over Σ and ε is the empty word. The length of a word w is $|w|$. When there is no danger

of confusion, a singleton set $\{w\}$ is denoted simply as w. The set of non-negative integers (respectively, rationals) is $\mathbb{N}_0$ (respectively, $\mathbb{Q}_0$).

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. A DFA A is *complete* if δ 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 DFA A is *minimal* if each state of 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, for $x, y \in \Sigma^*$,

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

A 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 complete DFA for L [19, 20]. 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.

Definition 1 ([17]). *An* additive weighted finite automaton *(WFA) is a 6-tuple $A = (Q, \Sigma, \gamma, \omega, q_0, F)$ where Q is a finite set of states, Σ is an alphabet, $\gamma : Q \times \Sigma \to 2^Q$ is the transition function, $\omega : Q \times \Sigma \times Q \to \mathbb{Q}_0$ is a partial weight function where $\omega(q_1, a, q_2)$ is defined if and only if $q_2 \in \gamma(q_1, a)$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states.*

Strictly speaking, the transitions of γ are also determined by the domain of the partial function β. In the following by a WFA we always mean an additive weighted finite automaton as in Definition 1. By a transition of A on symbol $a \in \Sigma$ we mean a triple (q_1, a, q_2) such that $q_2 \in \gamma(q_1, a)$, $q_1, q_2 \in Q$. A computation path α of a WFA A along a word $w = a_1 a_2 \cdots a_m$, $a_i \in \Sigma$, $i = 1, \ldots, m$, from state p_1 to p_2 is a sequence of transitions that spell out the word w,

$$\alpha = (q_0, a_1, q_1)(q_1, a_2, q_2) \cdots (q_{m-1}, a_m, q_m),$$

where $p_1 = q_0$, $p_2 = q_m$, and $q_i \in \gamma(q_{i-1}, a_i)$, $1 \leq i \leq m$. The weight of a computation path is

$$\omega(\alpha) = \sum_{i=1}^{m} \omega(q_{i-1}, a_i, q_i).$$

We let $\Theta(p_1, w, p_2)$ denote the set of all computation paths along a word w from p_1 to p_2. The *language recognized by A within the weight bound $r \geq 0$* is the set

of words for which there exists a computation path that is accepted by A and has weight at most r, defined as

$$L(A, r) = \{w \in \Sigma^* : (\exists f \in F)(\exists \alpha \in \Theta(q_0, w, f))\ \ \omega(\alpha) \leq r\}.$$

Proposition 1 ([17]). *If A is a WFA with n states where all transition weights are integers and $r \in \mathbb{N}_0$, then $L(A, r)$ can be recognized by a DFA with at most $(r+2)^n$ states.*

3 WFA Construction for a Quasi-Distance Neighbourhood

We construct a WFA to recognize the neighbourhood of a regular language with respect to a quasi-distance. First we recall some definitions concerning additive distances and quasi-distances between words [2].

A function $d : \Sigma^* \times \Sigma^* \to \mathbb{Q}_0$ is a *distance* if it satisfies, for all $x, y, z \in \Sigma^*$,

1. $d(x, y) = 0$ if and only if $x = y$,
2. $d(x, y) = d(y, x)$,
3. $d(x, z) \leq d(x, y) + d(y, z)$.

The function d is a *quasi-distance* if it satisfies conditions 2 and 3 and $d(x, y) = 0$ always when $x = y$, that is, a quasi-distance allows the possibility that distinct word may have distance zero. The *neighbourhood* of radius r of a language L is the set

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

A distance d is said to be *finite* if the neighbourhood of any given radius of an individual word with respect to d is finite. A (quasi-)distance d is *additive* if for every factorization $w = w_1 w_2$ and radius $r \geq 0$,

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

It is known that the neighbourhood of a regular language with respect to a quasi-distance is regular [2]. The next lemma constructs a WFA for this language. The construction is inspired by related constructions in [2, 18].

An additive (quasi-)distance d is determined by the finite number of values $d(a, b)$, $d(a, \varepsilon)$, where $a, b \in \Sigma$. For the complexity estimate of the lemma we assume that d is a fixed additive quasi-distance that is given by listing the values $d(a, b)$, $d(a, \varepsilon)$, $a, b \in \Sigma$.

Lemma 1. *Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA with n states, d an additive quasi-distance, and $R \geq 0$ is a constant. There exists an additive WFA A with n states such that for any $0 \leq r \leq R$,*

$$L(A, r) = E(L(N), d, r)$$

Furthermore, the WFA A can be constructed in time $O(n^3)$.

Proof. We define an additive WFA $A = (Q, \Sigma, \gamma, \omega, q_0, F)$ as follows. The transition function γ is defined by setting, for $p \in Q$, $a \in \Sigma$,

$$\gamma(p, a) = \{q : (\exists x \in \Sigma^*)\ q \in \delta(p, x) \text{ and } d(a, x) \leq R\}.$$

That is, for each pair of states p, q, we add a transition from p to q on a in the WFA A if there is a word $x \in \Sigma^*$ with $d(a, x) \leq R$ that takes p to q in the NFA N. The transition (p, a, q) in A has weight

$$\omega((p, a, q)) = \min_{x \in \Sigma^*} \{d(a, x) : q \in \delta(p, x)\}. \tag{1}$$

We claim that a word w spells out a path in A with weight r ($\leq R$) from the start state q_0 to a state q_1 if and only if some word u with $d(w, u) \leq r$ takes the state q_0 to q_1 in the NFA B.

We prove the "only if" direction of the claim using induction on the length of w. If $w = \varepsilon$, then $q_1 = q_0$ and there is nothing to prove. For the inductive step consider $w = ub$, $u \in \Sigma^*$, $b \in \Sigma$, where the claim holds for u. Since w takes state q_0 to q_1 by a path with weight r in the WFA A, the word u takes q_0 to a state p by a path of weight r_1 where $r_1 + \omega(p, b, q_1) = r$.

By the inductive assumption, there exists $u_p \in \Sigma^*$, $d(u, u_p) \leq r_1$ such that u_p in the NFA N takes q_0 to the state p. By the definition of the transition weights of A in (1), there exists a word $v_{p,b}$, with $d(b, v_{p,b}) = \omega(p, b, q_1)$ such that in the NFA N the word $v_{p,b}$ takes state p to state q_1.

Since d is additive and $r_1 + \omega(p, b, q_1) = r$, we have

$$E(u, d, r_1) \cdot E(b, d, \omega(p, b, q_1)) \subseteq E(w, d, r).$$

Thus, $d(w, u_p v_{p,b}) \leq r$ and in the NFA N the word $u_p v_{p,b}$ takes the start state q_0 to q_1. This concludes the proof of the "only if" direction of the claim.

An analogous argument establishes the "if" direction of the claim. Since the start states of A and N coincide and A and N have the same set of final states, the claim implies that, for any $r \leq R$, $L(A, r) = E(L(N), d, r)$.

It remains to give an upper bound for the time complexity of finding the weights (1) in order to verify the claim concerning the time bound for constructing A. Since d is additive, for given $p, q \in Q$ and $a \in \Sigma$, the set of words x such that $d(a, x) \leq R$ and x takes p to q in the NFA N is regular. This means that, for $p \in Q$ and $a \in \Sigma$, the set $\gamma(p, a)$ can be efficiently constructed and the weights of the transitions of N are computed as follows.

A word $x = b_1 b_2 \cdots b_m, b_i \in \Sigma$ is in the neighbourhood of a of radius R if and only if there exists an index $i \in \{1, \ldots, m\}$ such that

$$d(a, b_i) + \sum_{j \in \{1, \ldots, m\}, j \neq i} d(\epsilon, b_j) \leq R.$$

For the radius R neighbourhood of a, $a \in \Sigma$, we define the two-state WFA $B_a = (\{I_0, I_1\}, \Sigma, \eta, \rho, I_0, \{I_1\})$, shown in Figure 1. The states of B_a are $\{I_0, I_1\}$. For each symbol $\sigma \in \Sigma$, we define self-loop transitions $\eta(q, \sigma) = q$ with weight

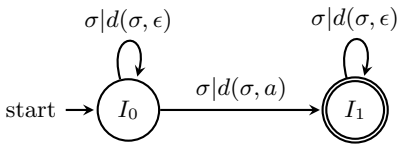

Fig. 1. The WFA B_a recognizing the language $\{x \in \Sigma : d(a,x) \leq R\}$

.

$d(\sigma,\epsilon)$ for both states and the transition $\eta(I_0,\sigma) = I_1$ with weight $d(\sigma,a)$ for the transition which consumes the symbol a.

Let $M_a = (\{I_0,I_1\} \times Q, \Sigma, \delta_a, \omega_a, (I_0,q_0), I_1 \times F)$ be the WFA obtained as a cross product of the WFA B_a and the NFA N. The states of M_a are of the form (P,q), where $P \in \{I_0,I_1\}$ and $q \in Q$. The transitions of M_a are defined by setting, for $q \in Q$, $\sigma \in \Sigma$,

$$\begin{aligned} \delta_a((I_0,q),\sigma) &= \{(I_0,\delta(q,\sigma)),(I_1,\delta(q,\sigma))\}, \\ \delta_a((I_1,q),\sigma) &= \{(I_1,\delta(q,\sigma))\}. \end{aligned}$$

The weights of transitions $((P_1,q_1),\sigma,(P_2,q_2))$ defined in δ_{M_a} are defined

$$\omega_a((P_1,q_1),\sigma,(P_2,q_2)) = \begin{cases} d(\sigma,\epsilon), & \text{if } P_1 = P_2; \\ d(\sigma,a), & \text{if } P_1 \neq P_2. \end{cases}$$

For states $p,q \in Q$, paths from states (I_0,p) to (I_1,q) are labelled by words x with weight $d(a,x)$.

We compute the paths with the least weight for every pair of states of M_a. There are $2n$ states in the product machine and minimal weight paths for every pair of states can be computed in time $O(n^3)$ via the Floyd-Warshall algorithm [4]. A transition from p to q on a is added if there is a path from (I_0,p) to (I_1,q) with weight at most R. □

Lemma 1 gives the following result.

Theorem 1. *Suppose that L has an NFA with n states and d is a quasi-distance. The neighbourhood of L of radius R can be recognized by an additive WFA having n states within weight bound R.*

As a consequence of Theorem 1 and Proposition 1 we get in Corollary 1 an upper bound for the state complexity of the neighbourhood of a regular language with respect to an additive quasi-distance d where all values $d(u,v)$, $u,v \in \Sigma^*$ are integers.

We note that if a quasi-distance d associates a non-negative integer value withany pair of words, then the weights of the WFA A constructed in the proof of Lemma 1 are integral. Furthermore, a neighbourhood with respect to a quasi-distance d with rational values can be converted to a neighbourhood with respect

to a quasi-distance with integral values by multiplying the radius and the values of d by a suitably chosen constant. This can be done since the distance between any two words is determined by distances between two alphabet symbols and alphabet symbols and the empty word.

Corollary 1. *Let N be an NFA with n states, $R \in \mathbb{N}_0$, and d a quasi-distance $\Sigma^* \times \Sigma^* \to \mathbb{N}_0$. Then the neighbourhood $E(L(N), d, R)$ can be recognized by a DFA with $(R+2)^n$ states.*

The upper bound $(R+2)^n$ is significantly better than what is obtained by first constructing an NFA for $E(L(N), d, R)$ as in [2] and then determinizing the NFA. If the set of states of N is Q, Theorem 8 of [2] [1] constructs an NFA for $E(L(N), d, R)$ with set of states $Q \times D$ where $D \subseteq \mathbb{N}$, roughly speaking, consists of all integers at most R that can be represented as a sum of distances between an element of Σ and an element of Σ^*.

We do not have a lower bound corresponding to the upper bound of Corollary 1, and the state complexity of neighbourhoods of regular languages with respect to an additive distance or quasi-distance remains an open question. Povarov [16] has given a lower bound for the radius-one Hamming neighbourhood of a regular language that is tight within an order of magnitude.

In the next section we will give a lower bound construction for the size of a DFA needed to simulate an additive WFA that matches the upper bound of Proposition 1. However, this does not necessarily shed light on the state complexity of neighbourhoods of regular languages because an arbitrary additive WFA need not recognize a neighbourhood of a (regular) language.

4 State Complexity of Weighted Finite Automata

Salomaa and Schofield [17] have given a matching lower bound construction for Proposition 1 using a family of WFAs over an alphabet of size $2n-1$ where n is the number of states of the WFA. Here, we define a family of WFAs over a five-letter alphabet which reaches the upper bound $(r+2)^n$.

Let $A_n = (Q_n, \Sigma, \gamma, \omega, 1, n)$ be an additive WFA with $Q_n = \{1, 2, \ldots, n\}$ and $\Sigma = \{a, b, c, d, e\}$. The transition function γ with $q \in Q$ and $\sigma \in \Sigma$ is defined

$$\gamma(q,\sigma) = \begin{cases} \{1,2\}, & \text{if } q = 1, \sigma = a \text{ or } q = 2, \sigma = b; \\ \{3\}, & \text{if } q = 1, \sigma = b \text{ or } q = 2, \sigma = a; \\ \{q+1\}, & \text{if } q = 3, \ldots, n-1 \text{ and } \sigma = a, b; \\ \{q\}, & \text{if } q = 1, \ldots, n \text{ and } \sigma = c, d, e. \end{cases}$$

[1] Theorem 8 of [2] assumes that N is deterministic. However, the construction used in the proof works also for an NFA.

The weight function ω for a transition $\alpha \in Q_n \times \Sigma \times Q_n$ is defined

$$\omega(\alpha) = \begin{cases} 1, & \text{if } \alpha = (1, c, 1); \\ 1, & \text{if } \alpha = (2, d, 2); \\ 1, & \text{if } \alpha = (q, e, q) \text{ for all } q \in Q; \\ 0, & \text{for all other transitions defined by } \gamma. \end{cases}$$

The transition diagram for A_n is shown in Figure 2 with the non-zero weights of each transition marked after the alphabet symbols labeling the transition. For example, state 1 has self-loops on a and d with weight zero and self-loops on c and e with weight one.

We will use the WFAs A_n to give a lower bound for the size of DFAs for a language recognized by a WFA within a given weight bound. First in Lemma 2 we establish a technical property of the weights of computations of A_n reaching a particular state and for this purpose we introduce the following notation.

For $0 \leq k_i \leq r + 1$ and $1 \leq i \leq n$, we define the words

$$w(k_1, \ldots, k_n) = \begin{cases} ac^{k_n} bd^{k_{n-1}} ac^{k_{n-2}} \cdots ac^{k_3} bd^{k_2} c^{k_1}, & \text{if } n \text{ is odd;} \\ abd^{k_n} ac^{k_{n-1}} bd^{k_{n-2}} \cdots ac^{k_3} bd^{k_2} c^{k_1}, & \text{if } n \text{ is even.} \end{cases}$$

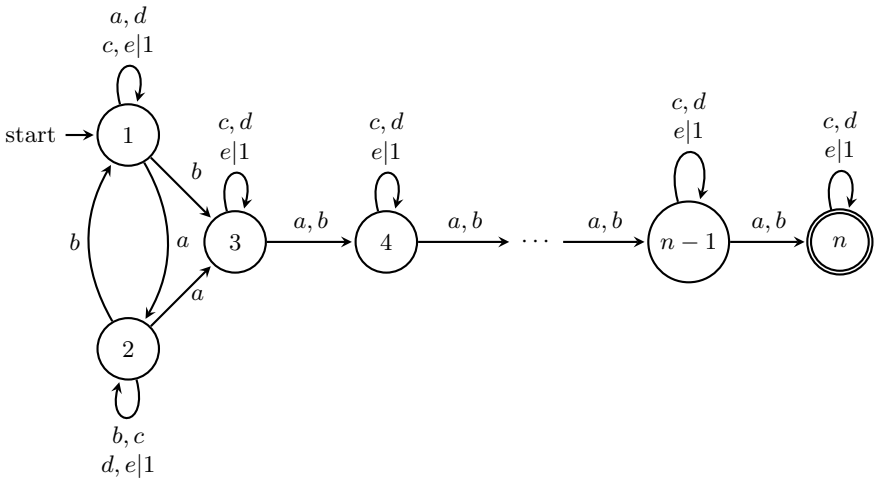

Fig. 2. The weighted finite automaton A_n used in the proof of Lemma 2.

Lemma 2. *Let $n \in \mathbb{N}$. The WFA A_n after processing the input $w(k_1, \ldots, k_n)$ can reach the state s, $1 \leq s \leq n$, on a path with weight k_s. Furthermore, any computation of A_n on input $w(k_1, \ldots, k_n)$ that reaches state s, $1 \leq s \leq n$, has weight k_s.*

Proof. In the string $w(k_1, \ldots, k_n)$ occurrences of symbols a and b alternate. Thus the computation of A can exit states 1 and 2 after making a self-loop on a in state 1 or a self-loop on b in state 2 and, furthermore, this is the only way for the computation to get out of the "binary cycle" of states 1 and 2.

Below using a case analysis we verify that, for $1 \leq s \leq n$, A_n has a computation with weight k_s that ends in state s and, furthermore, any computation ending in s has weight k_s.

(i) First consider the case where n is even. Consider a computation of A_n that reaches a state s where $s \geq 2$ is even. Note that after exiting the cycle of states 1 and 2, only the symbols a or b move the computation to the next state. Thus, the only way to reach s is that the computation must make a self-loop on b in state 2 directly before reading the substring d^{k_s}. After that the following k_s symbols d are read via the weight one transitions. This also applies for the case $s = 2$.
If $s \geq 3$ is odd, in order to reach state s, directly before reading the substring c^{k_s} the computation must on input a make a self-loop in state 1 and then the following k_s symbols c are read with transitions of weight one in state 1. Finally consider the case $s = 1$. In order to end in state 1, the computation must not have made any self-loops on a in state 1 or b in state 2. If this is done the computation ends in a state z with $z \geq 2$. Thus, reading the final b takes the computation from state 2 to state 1, where the transition on d is taken k_2 times. The computation remains in state 1 and reads the rest of the word c^{k_1} on the transition of weight 1 exactly k_1 times.

(ii) Next consider the case where n is odd. The above argument remains the same, almost word for word. The only minor difference is in the case $s = n$. In order to reach state n, the computation must read the first symbol a using a self-loop and then the following k_n symbols c using transitions of weight 1. (Note that when n is odd, in $w(k_1, \ldots, k_n)$ the first symbol a is followed by k_n symbols c.)

□

Lemma 3. *Let A_n be the WFA defined above and $r \in \mathbb{N}$. Then the minimal DFA for $L(A_n, r)$ needs $(r+2)^n$ states.*

Proof. It is sufficient to show that all words $w(k_1, \ldots, k_n)$, $0 \leq k_i \leq r+1$, $i = 1, \ldots, n$, belong to distinct classes of $\equiv_{L(A_n,r)}$.

Consider two distinct words $w(k_1, \ldots, k_n)$ and $w(k'_1, \ldots, k'_n)$ with $0 \leq k_i, k'_i \leq r+1, i = 1, \ldots, n$. There exists an index j such that $k_j \neq k'_j$. Without loss of generality, we assume that $k_j < k'_j$. Choose

$$z = e^{r-k_j} a^{n-j}.$$

Since $k_j < k'_j \leq r+1$, it follows that $r - k_j \geq 0$ and z is a well-defined word. We claim that

$$w(k_1, \ldots, k_n) \cdot z \in L(A, r), \quad w(k'_1, \ldots, k'_n) \cdot z \notin L(A, r).$$

By Lemma 2, A has a computation on input $w(k_1, \ldots, k_n)$ that ends in state j with weight k_j. In state j, A reads the first $r - k_j$ symbols e of z, after which the total weight is $k_j + (r - k_j) = r$. The zero weight transitions on the suffix a^{n-j} take the automaton from state j to the final state n.

Now consider from which states q the WFA A can reach the accepting state n on input z. On any state of A, the symbols c, d, e define self-loops. On states $3 \leq q \leq n-1$, transitions to state $q+1$ only occur on a, b. For states $q = 1, 2$, a transition to state $q+1$ occurs only on a. Thus, A can reach the accepting state n from a state q on input z only if $q = j$.

Thus, the only possibility for A to accept $w(k'_1, \ldots, k'_n) \cdot z$ would be that the computation has to reach state j on the prefix $w(k'_1, \ldots, k'_n)$. By Lemma 2, the weight of this computation can only be k'_j. But when continuing the computation on z from state j, A has to read the first $r - k_j$ symbols e, each with a self-loop transition having weight one. After this, the weight of the computation will be $k'_j + r - k_j > r$. Thus, $w(k'_1, \ldots, k'_n) \cdot z \notin L(A, r)$.

Thus, the equivalence relation $\equiv_{L(A,r)}$ has index at least $(r+2)^n$. □

As a consequence of Lemma 3 and Proposition 1 we have:

Theorem 2. *If A is an n state WFA with integer weights for transitions and $r \in \mathbb{N}$, then*

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

For $n, r \in \mathbb{N}$, there exists an n state WFA A with integral weights defined over a five-letter alphabet such that $\mathrm{sc}(L(A, r)) = (r+2)^n$.

5 Conclusion

For the state complexity of a language recognized by an additive WFA with a given weight we have established a tight lower bound using a constant size alphabet. The earlier known lower bound construction [17] used a variable alphabet that has size linear in the number of states of the WFA.

We have also constructed a WFA recognizing the neighbourhood of a regular language with respect to an additive quasi-distance. This yields an upper bound $(r+2)^n$ for the state complexity of a neighbourhood of radius r of an n state NFA language with respect to an additive quasi-distance. The upper bound is significantly better than a bound obtained by directly constructing an NFA for the neighbourhood [2] and then determinizing the NFA. The same upper bound $(r+2)^n$ has been known previously for neighbourhoods with respect to an additive distance.

The precise state complexity of neighbourhoods with respect to a distance or a quasi-distance remains open. Povarov [16] gives an upper bound $n \cdot 2^{n-1} + 1$ for the Hamming neighbourhood of radius one of an n-state regular language and an almost matching lower bound. For neighbourhoods of radius $r \geq 2$ no good lower bounds are known. Finding such lower bounds will be a topic of a forthcoming paper [14].

References

1. Benedikt, M., Puppis, G., Riveros, C.: Bounded repairability of word languages. Journal of Computer and System Science 79 (2013) 1302–1321
2. Calude, C.S., Salomaa, K., Yu, S.: Distances and quasi-distances between words. Journal of Universal Computer Science 8(2) (2002) 141–152
3. Choffrut, C., Pighizzini, G.: Distances between languages and reflexivity of relations. Theoretical Computer Science 286 (2002) 117–138
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd ed. MIT Press, Cambridge, Massachuchetts (2001)
5. Deza, M.M., Deza, E.: *Encyclopedia of Distances.* Springer-Verlag, Berlin-Heidelberg (2009)
6. Droste, M., Kuich W., Vogler, H. (Eds.): *Handbook of Weighted Automata.* EATCS Monographs in Theoretical Computer Science, Springer (2009)
7. Eramian, M.: Efficient simulation of nondeterministic weighted finite automata. Journal of Automata, Languages, and Combinatorics 9 (2004) 257–267
8. 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 (2013) 1067–1082
9. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata — A survey. Inf. Comput. 209 (2011) 456–470
10. Kari, L., Konstantinidis, S.: Descriptional complexity of error/edit systems. Journal of Automata, Languages, and Combinatorics 9 (2004) 293–309
11. Konstantinidis, S.: Transducers and the properties of error detection, error-correction, and finite-delay decodability. Journal of Universal Computer Science 8 (2002) 278–291
12. Konstantinidis, S.: Computing the edit distance of a regular language. Information and Computation 205 (2007) 1307–1316
13. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8) (1966) 707–710
14. Ng, T., Rappaport, D., Salomaa, K.: State complexity of neighbourhoods and approximate pattern matching. Submitted for publication (March 2015)
15. Pighizzini, G.: How hard is computing the edit distance? Information and Computation 165 (2001) 1–13
16. Povarov, G.: Descriptive complexity of the Hamming neighborhood of a regular language. Proceedings of the 1st International Conference Language and Automata Theory and Applications, LATA 2007, pp. 509–520
17. Salomaa, K., Schofield, P.: State complexity of additive weighted finite automata. Intenational Journal of Foundations of Computer Science 18(6) (2007) 1407–1416
18. Schofield, P.: Error Quantification and Recognition Using Weighted Finite Automata. MSc thesis, Queen's University, Kingston, Canada (2006)
19. Shallit, J.: *A Second Course in Formal Languages and Automata Theory,* Cambridge University Press (2009)
20. Yu, S.: Regular languages, in: *Handbook of Formal Languages,* Vol. I, (G. Rozenberg, A. Salomaa, Eds.), Springer, 1997, pp. 41–110