Descriptional Complexity of Error Detection

Timothy Ng, David Rappaport and Kai Salomaa

Abstract The neighbourhood of a language L consists of all strings that are within a given distance from a string of L. For example, additive distances or the prefix-distance are regularity preserving in the sense that the neighbourhood of a regular language is always regular. For error detection and error correction applications an important question is to determine the size of the minimal deterministic finite automaton (DFA) needed to recognize the neighbourhood of a language recognized by an n state DFA. This paper surveys recent work on the state complexity of neighbourhoods of regularity preserving distances.

1 Introduction

Distance is a fundamental concept in mathematics which gives a numerical value to express the "closeness" of two objects. How we define "closeness" depends on what the objects we want to compare are and why we want to compare them. Here the objects we are interested in are strings, or sequences of symbols. Strings are particularly important in computer science, where many different kinds of objects are often represented as sequences of symbols.

A distance between strings can be extended into a distance between sets of strings, or languages. There are various ways to extend distances from strings to languages which are motivated by a number of applications, such as specification repair [1], computational biology [24], and error detection in communication channels [16, 19]. Information on the theory and applications of error correcting codes can be found in [7, 25, 36].

T. Ng · D. Rappaport · K. Salomaa (✉)
School of Computing, Queen's University, Kingston, ON K7L 2N8, Canada
e-mail: ksalomaa@cs.queensu.ca

T. Ng
e-mail: ng@cs.queensu.ca

D. Rappaport
e-mail: daver@cs.queensu.ca

A. Adamatzky (ed.), *Emergent Computation*, Emergence, Complexity
and Computation 24, DOI 10.1007/978-3-319-46376-6_6

The *Encyclopedia of Distances* by Deza and Deza [8] contains an extensive list of distances that are used across a large number of different fields, including geometry, biology, coding theory, image processing, and physics, among others. For each of these definitions, we can ask questions about the behaviour of these distances and their properties. The computational question typically considered is how hard it is to compute the distance between given languages [12, 14, 17, 29].

Suppose we are given a distance d between strings. A mathematically elegant extension of d to languages L_1 and L_2 is the *Hausdorff distance* [6, 8], which gives a good overall measure of the similarity of L_1 and L_2. On the other hand, for error detection and error correction applications when the distance function is used to measure the number of errors in strings, the natural way to define the distance between languages L_1 and L_2 is to take simply the distance between two closest strings in L_1 and L_2, respectively. If we assume that errors have unit weight, then L_1 and L_2 having distance r means that we can distinguish strings of L_1 and L_2, respectively, on a channel that introduces at most $r - 1$ errors [13, 18]. A related notion is the *inner distance* (or *self-distance*) of a language: if the distance of any two distinct strings of a language L is at least r, the language L corrects $r - 1$ errors [14, 16, 19].

The *neighbourhood* of radius r of a language L consists of all strings that are within distance at most r from a string of L. We say that a distance d is *regularity preserving* if the neighbourhood of a regular language with respect to d is always regular. This gives rise to the question how large is the deterministic finite automaton (DFA) needed to recognize the neighbourhood of a regular language. Roughly speaking, 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 is equal to the state complexity of the set of strings that have distance at least $r + 1$ from any string in L.[1] Over the last 20 years there has been much work on the state complexity of various regularity preserving operations and the reader can find more references in the survey [11].

It is known that the neighbourhood of a regular language with respect to an *additive distance* or *additive quasi-distance* [4] or with respect to the *prefix distance* and its variants [6] is always regular. The state complexity of neighbourhoods with respect to the Hamming distance was first considered by Povarov [30]. A tight lower bound for general additive distances was given by the current authors, however, a limitation is that the alphabet size depends on the size of the original DFA.

This paper surveys algorithmic properties and descriptional complexity of commonly used distance measures between sets of strings and, in particular, recent work on the state complexity of regularity preserving distances and related questions, such as approximate pattern matching. The contents of the paper is as follows. In the next section we recall some basic definitions on finite automata and Sect. 3 discusses distances between languages and regularity preserving distances. Descriptional com-

[1] Strictly speaking, for incomplete DFAs the state complexity of L and the complement of L, respectively, may differ by one.

plexity of neighbourhoods of regular languages is discussed in Sect. 4 and the last section highlights some open problems and further research topics on the descriptional complexity of error channels.

2 Definitions

We assume that the reader is familiar with the basics of finite automata and regular languages and below we just fix some notations. All unexplained notions can be found e.g. in the texts by Shallit [35] or Yu [37].

In the following Σ stands always for a finite alphabet and Σ^* is the set of strings over Σ. The length of a string $w \in \Sigma^*$ is $|w|$ and ε is the empty string. For $1 \leq i \leq |w|$, w_i stands for the ith symbol of w. The reversal of a string w is $w^R = w_{|w|}w_{|w|-1} \ldots w_1$. If $w = xyz$ we say that x is a prefix, z is a suffix and y is a substring of w. Here any of the strings x, y, z may be ε.

A nondeterministic finite automaton (NFA) is a tuple $A = (\Sigma, Q, \delta, Q_0, F)$ where Σ is the input alphabet, Q is the finite set of states, $\delta\colon Q \times \Sigma \to 2^Q$ is the multivalued transition function, $Q_0 \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states. In the usual way δ is extended as a function $Q \times \Sigma^* \to 2^Q$ and the language accepted by A is $L(A) = \{w \in \Sigma^* \mid \delta(Q_0, w) \cap F \neq \emptyset\}$. The automaton A is a deterministic finite automaton (DFA) if $|Q_0| = 1$ and δ is a single valued partial function. If δ is a total function, the DFA A is complete. Note that our definition allows DFAs to be incomplete, i.e., some transitions may be undefined. In cases where the transitions function is required to be always defined we use the term "complete DFA". It is well known that the deterministic and nondeterministic finite automata recognize the class of *regular languages.*

The (right) Kleene congruence of a language $L \subseteq \Sigma^*$ is the relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ defined by setting $x \equiv_L y$ 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 complete DFA for L [35]. For a given regular language L, the number of states of the minimal incomplete and minimal complete DFA recognizing L differ by at most one.

By the *state complexity* of a regular language L, $\mathrm{sc}(L)$, we mean the number of states of the minimal incomplete DFA recognizing L. The *nondeterministic state complexity* of L, $\mathrm{nsc}(L)$, is the number of states of a minimal NFA recognizing L. Note that a state minimal NFA for a regular language need not be unique.

3 Distance Measures on Strings and Sets of Strings

A *distance* on strings over Σ is a function $d : \Sigma^* \times \Sigma^* \to \mathbb{Q}$ which for all strings $x, y, z \in \Sigma^*$ satisfies the following

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

From the above conditions it follows easily that $d(x, y)$ must be non-negative for all $x, y \in \Sigma^*$.

A *quasi-distance* is a function for which the first condition is weakened from "iff" to "if", that is, a quasi-distance of two distinct strings may be zero. If d is a quasi-distance on Σ, we can define an equivalence relation $\sim_d$ on Σ by setting $x \sim_d y$ if and only if $d(x, y) = 0$. Then the mapping $d'([x]_{\sim_d}, [y]_{\sim_d}) = d(x, y)$ is a distance over $\Sigma^*/\sim_d$ [4].

A quasi-distance d is *integral* if for all strings x and y, $d(x, y) \in \mathbb{N}$. Note that a distance is a special case of a quasi-distance and all properties that hold for quasi-distances apply also to distances.

We now recall the definition of some commonly used distance measures on strings. The *Hamming distance* of two equal length strings x and y counts the number of positions in which x differs from y. Formally, the Hamming distance between strings $x, y \in \Sigma^*$ is defined as

$$d_H(x, y) = \begin{cases} |\{1 \leq i \leq |w| \mid x_i \neq y_i\}| & \text{if } |x| = |y|, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The distance is defined only when x is the same length as y.

For equal length strings Hamming distance counts the number of substitution operations needed to transform x into y. A natural extension for all pairs of strings is the *Levenshtein distance* [22], also called the *edit distance*, which counts the number of atomic substitution, insertion and deletion operations required to transform x into y. Formally the Levenshtein distance can be defined in terms of error systems considered by Kari and Konstantinidis [13] as a formalization of error in terms of formal languages, see also [15]. An error system is a formal language over the alphabet of edit operations. For an alphabet Σ, let $\mathcal{E}_\Sigma$ be the alphabet of edit operations over Σ defined by

$$\mathcal{E}_\Sigma = \{ (a/b) \mid a, b \in \Sigma \cup \{\varepsilon\}, ab \neq \epsilon \}.$$

An *error* is an edit operation (a/b) where $a \neq b$. An *edit string* is a string over $\mathcal{E}_\Sigma$. The weight $|e|_{\neq}$ of an edit string e is the number of errors in e. For an edit string $e = (a_1/b_1)(a_2/b_2)\dots(a_n/b_n)$, we call $x = a_1a_2\dots a_n$ the input part and $y = b_1b_2\dots b_n$ the output part of e $(a_i, b_i \in \Sigma \cup \{\varepsilon\},\ i = 1, \dots, n)$.

Now the edit distance between strings x and y, $d_e(x, y)$, is defined as the minimum weight edit string e having x (respectively, y) as the input (respectively, the output) part. The above definition has assigned weight one to all errors. It is possible to consider also definitions where the weights need not be equal [12, 13]. General edit distances are examples of additive quasi-distances considered in Sect. 3.2.

Example 1 An edit string to transform the string *hiphop* into *lollipop* is

$$e_1 = \frac{\varepsilon}{l}\frac{h}{o}\frac{i}{l}\frac{p}{l}\frac{h}{i}\frac{\varepsilon}{p}\frac{o}{o}\frac{p}{p}.$$

The length of this edit string is 8 and its weight is 6.

The edit string e_1 is not a minimum weight edit string between the given strings. The edit distance of the string *hiphop* and *lollipop* is 5, via the edit string

$$\frac{h}{l}\frac{\varepsilon}{o}\frac{\epsilon}{l}\frac{\varepsilon}{l}\frac{i}{i}\frac{p}{p}\frac{h}{\varepsilon}\frac{o}{o}\frac{p}{p}.$$

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 [6]. A parameterized prefix distance between regular languages has been considered by Kutrib et al. [20] for estimating the fault tolerance of information transmission applications. 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. Formally it is defined by

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

The definitions of the *suffix distance* and *substring distance* are analogous.[2]

Example 2 The strings *yorkdale* and *yorkville* have a prefix distance of 9 via their longest common prefix *york*. The strings *woodbine* and *guildwood* have a substring distance of 9 through their longest common substring *wood*. The strings *parkdale* and *riverdale* have a suffix distance of 9 through the longest common suffix *dale*.

3.1 Distance Between languages

If d is a distance on strings over Σ, the natural way to define the distance between a string $w \in \Sigma^*$ and a language $L \subseteq \Sigma^*$ is

$$d(w, L) = \inf\{d(w, w') \mid w' \in L\}.$$

We want to further extend the definition to measure the distance between two languages, or sets of strings. The *relative distance* [6] of the language L_1 to the language L_2 is defined as

$$d(L_1|L_2) = \sup\{d(w_1, L_2) \mid w_1 \in L_1\}.$$

[2] The latter is called in [6] subword distance but this term has been used also for a distance defined in terms of the longest noncontinuous subword [23].

The *bounded repair problem* [1] consists of deciding whether or not the relative edit distance of the *restriction language* L_1 to the *target language* L_2 is finite. The bounded repair problem (in the general non-streaming case) is PSPACE-complete when the restriction and target language are specified by NFAs and it is coNP-complete when the restriction language is specified by an NFA and the target language by a DFA [1]. The coNP-hardness result holds also when the restriction language is specified by a DFA. A variant of the bounded repair problem asks, roughly speaking, what is the number of edits per symbol of a string w in the restriction language that is needed to transform w to a string of the target language [2]. Chatterjee et al. [5] have studied systematically the complexity of the closely related *threshold edit distance problem* for pushdown automata and finite automata.

Note that the relative distance is not, in general, symmetric. In order to satisfy symmetry, Choffrut and Pighizzini [6] use the *Hausdorff distance* to define the distance between L_1 and L_2 by taking the maximum value

$$d_{\text{Hdorff}}(L_1, L_2) = \max\{\ d(L_1|L_2),\ d(L_2|L_1)\ \}.$$

The relative edit distance between regular languages was shown to be computable by reducing it to the limitedness problem for distance automata [6], and Leung and Podolskiy [21] have given an exponential time algorithm as well as a PSPACE-hardness lower bound for the limitedness problem. As mentioned above, a PSPACE algorithm for the relative distance between regular languages is known from the more recent work on the bounded repair problem [1].

The Hausdorff distance having a small value means, intuitively, that every string of L_1 is close to some strings of L_2 and vice versa, that is, the binary relation $L_1 \times L_2$ is "almost reflexive" with respect to the distance under consideration [6].

In the above sense the Hausdorff distance gives a good measure of similarity between languages. However, in error detection applications we want to ensure that every string of L_1 is at some minimum distance from every string of L_2 and vice versa. Thus, in the following we extend a distance from strings to languages simply by taking the smallest distance of two strings in the respective languages:

$$d(L_1, L_2) = \inf\{\ d(w_1, w_2)\ \mid\ w_1 \in L_1, w_2 \in L_2\ \}.$$

Unless otherwise mentioned, in the following when speaking about the edit distance (or some other distance measure) for languages, we mean the above definition. The *inner distance* of a language L [17] is

$$d(L) = \inf\{\ d(w, z)\ \mid\ w, z \in L,\ w \neq z\ \}.$$

The maximal error-detecting capability of a language L, in the sense defined by Konstantinidis and Silva [19], is one less than the inner distance of L.

The edit distance between two regular languages can be computed in polynomial time and the corresponding question for context-free languages is unsolvable [24].

Also the edit distance between a regular language and a context-free language can be computed in polynomial time [12].

3.2 Regularity Preserving Distances

We can consider the topological notion of neighbourhoods, or balls, of radius r with respect to a given distance d on strings. Informally, a neighbourhood of a language L is the set of strings which are at most a distance r away from some string in L according to the distance measure under consideration.

Formally, the neighbourhood of radius $r \geq 0$ of a language L under quasi-distance d is defined as

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

Suppose that the distance d measures the number of errors introduced in an information transmission channel [13, 15, 19]. Then $E(L, d, r)$ consists of all strings that a channel that introduces at most r errors can output when the input is a string of L. In other words, the complement of $E(L, d, r)$ is the unique maximal language L' such that $d(L, L') > r$.

Using the notion of neighbourhood we define the following notions on quasi-distances:

Definition 1 Let d be a quasi-distance on Σ^*. We say that d is *finite* if, for all $w \in \Sigma^*$ and $r \geq 0$ the neighbourhood $E(\{w\}, d, r)$ is finite.

We say that d is *regularity preserving* if for all regular languages L and $r \geq 0$, the neighbourhood $E(L, d, r)$ is regular.

For example, the edit, prefix, suffix and substring-distances are clearly all finite. On the other hand, it is known that finiteness of a distance d does not guarantee that d is regularity preserving [4]. Calude et al. [4] introduced a notion of additivity that is sufficient to guarantee that a quasi-distance preserves regularity. We say that a quasi-distance is *additive* if it respects composition of strings in the following sense.

Definition 2 A quasi-distance d on Σ^* is additive if for all $w_1, w_2 \in \Sigma^*$ and $r \geq 0$,

$$E(\{w_1 w_2\}, d, r) = \bigcup_{r_1 + r_2 = r} E(\{w_1\}, d, r_1) \cdot E(\{w_2\}, d, r_2), \tag{1}$$

Informally the additivity of d means that any neighbourhood of radius r of a concatenation of two strings $w_1 w_2$ consists of exactly all the language concatenations of neighbourhoods of w_1 and w_2 whose radii sum up to r. An additive distance is always finite but additive quasi-distances need not be finite [4].

It is easy to verify that the edit distance d_e is additive. For the edit distance the inclusion from left to right in (1) holds directly by definition and verifying the converse inclusion needs a short proof [4]. Also, the Hamming distance can be viewed as an additive distance if we define that the cost of deletions and insertions is infinite, that is, the distance between symbols of Σ and ε is defined to be infinite.

The additivity property is sufficient to guarantee that any neighbourhood of a regular language is regular.

Theorem 1 *([4]) An additive quasi-distance preserves regularity.*

For a given DFA A and radius $r \geq 0$, the original proof of Theorem 1 in [4] first verifies that the neighbourhoods of individual alphabet symbols are regular and then using this property and additivity of the quasi-distance d constructs an NFA for the neighbourhood $E(L(A), d, r)$. The construction is far from optimal from the state complexity point of view (since it results only in an NFA that, in general, needs to be determinized) and in the next section we will discuss constructions with better state complexity. Schulz and Mihov [34] have given a time efficient DFA construction for the neighbourhood of a single string w: if the radius is viewed as a constant, the DFA can be constructed in time linear in the length of w.

On the other hand, additivity is not necessary for a distance to preserve recognizability. It is clear that the prefix-, suffix- or substring distances are not additive. However, given a DFA A it is easy to construct an NFA B that recognizes a neighbourhood of $L(A)$ with respect to the prefix-distance by, roughly speaking, guessing the longest common prefix w_p of the input and a string in $L(A)$, and then counting the length of the remaining suffix of the input. Naturally the number of states of B has to depend on the radius, and depending on the state the simulated computation of A ends in, the NFA B can "know" the length of the shortest suffix that completes w_p to a string of $L(A)$. An analogous construction works for the suffix- and the substring distance.

Theorem 2 *([6, 28]) The prefix-, suffix- and substring distances preserve regularity.*

4 State Complexity of Neighbourhoods

A (combinatorial) channel is, in general, a binary relation on strings describing all input-output situations permitted by the channel. For more information on error channels and error detection we refer the reader e.g. to [9, 13, 16, 19]. If all substitution, insertion and deletion errors have a unit cost, the number of errors introduced by the channel is upper bounded by the edit distance of an input–output pair, and when using errors with general weights (including possibly zero weight) we can bound the number of errors by an additive quasi-distance.

Now assume that our channel $\mathcal{C}$ introduces at most r errors and consider a regular language L that is recognized by a DFA with n states. The the complement of the neighbourhood $E(L, d_e, 2r + 1)$ is the unique maximal language L' such that we can distinguish outputs produced by $\mathcal{C}$ on inputs from L and L', respectively. Complementation changes the size of a minimal incomplete DFA by at most one. This means that determining the state complexity of the neighbourhood $E(L, d_e, 2r+1)$ as a function of n can be viewed as the state complexity of error detection on a channel with at most r errors.

Povarov [30] was the first to systematically investigate the state complexity of Hamming-neighbourhoods. Note that a Hamming-neighbourhood of radius r can be viewed as a radius r neighbourhood where the underlying edit distance assigns value $r+1$ to all insertion and deletion operations, and in this way the Hamming distance can be interpreted as a special case of an additive distance.

We begin by recalling the NFA construction for Hamming neighbourhoods due to Povarov [30] since it is used for the first upper bound for deterministic state complexity, as well as in later constructions from [27]. An alternative proof for the upper bound of Theorem 3 based on finite transducers can be found in [31]. Recall that d_H denotes the Hamming distance.

Theorem 3 *([30, 31]) If $L \subseteq \Sigma^*$ has an NFA with n states and $r \in \mathbb{N}$, then*

$$\mathrm{nsc}(E(L, d_H, r)) \leq n \cdot (r+1).$$

For every $r \in \mathbb{N}$ and $n > r$ there exists an n-state NFA A over a two letter alphabet such that

$$\mathrm{nsc}(E(L(A), d_H, r) = n \cdot (r+1).$$

Proof sketch for the upper bound. Suppose L is recognized by an NFA $A = (\Sigma, Q, \delta, q_0, F_A)$. The neighbourhood $E(L(A), d_H, r)$ is recognized by an NFA

$$B = (\Sigma, Q \times \{0, 1, \ldots, r\}, \gamma, (q_0, 0), F_A \times \{0, 1, \ldots, r\}),$$

where the transitions of γ are defined by setting for $q \in Q$, $0 \leq i \leq r$ and $b \in \Sigma$:

$$\gamma((q,i), b) = \begin{cases} \{(p,i) \mid p \in \delta(q,b)\} \cup \{(p, i+1) \mid (\exists c \in \Sigma)\ p \in \delta(q,c)\} & \text{if } i < r, \\ \{(p,i) \mid p \in \delta(q,b)\} & \text{if } i = r. \end{cases}$$

The first component of the state of B simulates a computation of A on some input (possibly containing errors), and the second component keeps track of the cumulative error. Note that the definition of γ allows the possibility of increasing the value of the second component also on a transition with the correct input symbol (the case when $c = b$). These transitions, although redundant, clearly do not change the language of B. □

Theorem 3 is stated in [30] using the Hamming distance and the same upper bound straightforwardly translates for any additive integral quasi-distance. In the NFA construction the set of states remains $Q \times \{0, 1, \ldots, r\}$ and, when the input symbol is b, an error transition on $c \in \Sigma \cup \{\varepsilon\}$ takes state (q, i), $(q \in Q, 0 \leq i \leq r)$ to all possible states $(p, i + d(b, c))$, where $i + d(b, c) \leq r$ and p is reached from state q on input c in the original NFA. Additionally the construction adds ε-transitions that simulate an insertion operation.

Corollary 1 *If L is recognized by an NFA with n states, d is an additive integral quasi-distance and $r \in \mathbb{N}_0$, then*

$$\operatorname{nsc}(E(L, d, r)) \leq n \cdot (r + 1).$$

The derivation of an upper bound for the deterministic state complexity of Hamming neighbourhoods uses the NFA construction of the proof of Theorem 3 and the upper bound for the number of reachable states for the corresponding DFA makes use of the redundant transitions $(p, i + 1) \in \gamma((q, i), b)$ where $p \in \delta(q, b)$ in the definition of the NFA transition relation. Since [30] uses complete DFAs, below we translate the upper bound construction also for incomplete DFAs. The lower bound of Theorem 4 for neighbourhoods of radius one uses a construction where the minimal DFA for the Hamming neighbourhood does not have a dead state which means that the same lower bound holds when state complexity is based on incomplete DFAs.

Theorem 4 *([30])*

(i) *If A is a complete DFA with n states and $r \in \mathbb{N}_0$, then $E(L(A), d_H, r))$ has a complete DFA with at most $\frac{1}{2} \cdot n \cdot 2^{nr} + 1$ states.*
(ii) *If A is an incomplete DFA with n states and $r \in \mathbb{N}_0$, then $E(L(A), d_H, r))$ has an incomplete DFA with at most $\frac{1}{2} \cdot (n + 2) \cdot 2^{nr}$ states.*
(iii) *For all $n \geq 4$, there exists a complete DFA A with n states defined over a binary alphabet such that*

$$\operatorname{sc}(E(L(A), d_H, 1) = \frac{3}{8} n \cdot 2^n - 2^{n-4} + n.$$

Proof The proofs of (i) and (iii) can be found in [30]. Here we just translate the former proof for incomplete DFAs to give the estimation (ii).

Suppose L is recognized by an incomplete DFA $A = (\Sigma, Q, \delta, q_0, F_A)$ where $|Q| = n$. Let $B = (\Sigma, P, \gamma, p_0, F_B)$ be the NFA constructed for the neighbourhood $E(L(A), d_H, r)$ as in the proof of Theorem 3. In particular, the set of states P is $Q \times \{0, 1, \ldots, r\}$.

Let B' be the DFA obtained from B using the standard subset construction. Since A is deterministic, from the construction of B it follows that any subset of P that is reachable as a state of B' can have at most one element of $Q \times \{0\}$ and, furthermore, the reachable subsets $X \subseteq P$ have the following property. If $X \neq \{(q_0, 0)\}$ and X contains an element $(q, 0)$, $q \in Q$, then from the definition of the transitions of the NFA B it follows that also $(q, 1) \in X$.

This means that the number of non-empty subsets of P that are reachable as states of B' is upper bounded by

$$1 + (n \cdot 2^{n-1} + 2^n) \cdot 2^{n(r-1)} - 1 = \frac{1}{2} \cdot (n + 2) \cdot 2^{nr}.$$

□

The lower bound for radius one neighbourhoods is roughly within a factor of $\frac{3}{4}$ of the upper bound of Theorem 4. Significantly, the lower bound is over a binary alphabet—up to date this is the only good lower bound for the deterministic state complexity of additive neighbourhoods where the alphabet size does not depend on the size of the DFA and, furthermore, the underlying distance is just the Hamming distance.

Shamkin [33] has constructed finite languages L_n, $n \geq 4$, over a ternary alphabet such that L_n has an incomplete DFA of size n and for all $r \leq \frac{n}{2} - 1$ the state complexity of the radius r Hamming neighbourhood of L_n is at least $2^{\lfloor \frac{n}{2} - r \rfloor}$. The lower bound for Hamming neighbourhoods of radius $r \geq 1$ is proportional to 2^{-r}, that is, with a fixed number of states the lower bound decreases with increasing radius. This seems to be quite far from the upper bound.

The upper bounds of Theorem 4 could be improved by adding further redundant transitions to the original NFA construction (from Theorem 3) and then using a more detailed analysis of the number reachable states of the corresponding DFA. However, in the next subsection we get a better upper bound for the deterministic state complexity of neighbourhoods using a different approach. Instead of constructing an NFA and then determinizing it, we construct a DFA for the neighbourhood directly based on the finite automaton recognizing the original language.

4.1 Neighbourhoods of General Additive Distances

Here we consider the state complexity of neighbourhoods with respect to a general additive distance, and in the next subsection the prefix-distance and other related distance functions. If d is a regularity preserving distance, in the informal discussion we use the term *state complexity of d* to mean the state complexity of neighbourhoods with respect to d (given as a function of the state complexity of the original regular language).

In the rest of this section, without separate mention we assume that all distances and quasi-distances are integral, i.e., the range of values consists of the non-negative integers. When considering the state complexity of neighbourhoods, this is not more restrictive than using rational values. Note that an additive distance d is completely determined by the distances between elements of $\Sigma \cup \{\varepsilon\}$. Thus, if d has rational values we can find a constant k such that there is an integral distance d' that satisfies, for all strings $x, y \in \Sigma^*$, $d'(x, y) = k \cdot d(x, y)$. Consequently for any language L and radius $r \geq 0$, $E(L, d, r) = E(L, d', k \cdot r)$.

Theorem 5 *([26, 32]) Let d be an additive quasi-distance. If L has an NFA with n states and $r \geq 0$,*

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

The statement of Theorem 5 in [26, 32] does not have the term "-1" because there DFAs are required to be complete. The proof (for distances and quasi-distances in [26, 32], respectively) uses a construction based on additive weighted finite automata. Below we outline a direct DFA construction for the neighbourhood of an additive distance.

Proof sketch for Theorem 5. For simplicity we assume that d is a distance. This implies that for any $w \in \Sigma^*$ and $r' \geq 0$, $E(\{w\}, d, r')$ is finite [4].

Let $A = (\Sigma, Q, \delta, q_0, F_A)$ and denote $Q = \{q_0, q_1, \dots, q_{n-1}\}$. We construct for the neighbourhood $E(L(A), d, r)$ a DFA $B = (\Sigma, P, \gamma, p_0, F_B)$ where the set of states is

$$P = \{0, 1, \dots, r+1\}^n - \{(r+1, \dots, r+1)\},$$

$$F_B = \{(x_0, x_1, \dots, x_{n-1}) \in P \mid (\exists 0 \leq j \leq n-1)\ x_j \leq r \text{ and } q_j \in F_A\},$$

and $p_0 = (0, i_1, i_2, \dots, i_{n-1})$, where i_j, $1 \leq j \leq n-1$, is the minimum of the set

$$S_{\text{dist}-\varepsilon} = (\{d(\varepsilon, w) \mid q_j \in \delta(q_0, w)\} \cup \{r+1\}).$$

Finally, the transitions of γ are defined as follows. For $b \in \Sigma$ and $(x_0, x_1, \dots, x_{n-1}) \in P$, we define

$$\gamma((x_0, x_1, \dots, x_{n-1}), b) = (z_0, z_1, \dots, z_{n-1}),$$

where z_j, $0 \leq j \leq n-1$, is the minimum of the set

$$S_{\text{dist}-j} = \{x_i + d(b, w) \mid q_j \in \delta(q_i, w)\} \cup \{r+1\}.$$

Since d is a distance, the neighbourhood $E(\{b\}, d, r+1)$ is finite and the set $S_{\text{dist}-j}$ can be effectively constructed.

Intuitively, the DFA B operates as follows. Assuming that B has processed an input $u \in \Sigma^*$ and the current state is $(x_0, x_1, \dots, x_{n-1})$, the component x_j, $0 \leq j \leq n-1$, is the smallest distance between u and a string $w \in \Sigma^*$ that in the original NFA A takes the state q_0 to q_j. If $x_j = r+1$, then there is no string w with $d(u, w) \leq r$ that takes q_0 to q_j. The state $(r+1, \dots, r+1)$ would correspond to the dead state of the computation and is omitted from the set of states.

The initial state p_0 is chosen to satisfy the above property. Using the additivity of d it can be verified inductively that if the state $(x_0, x_1, \dots, x_{n-1})$ satisfies the above described property, so does $\gamma((x_0, x_1, \dots, x_{n-1}), b)$, $b \in \Sigma$. (The formal argument for correctness is analogous to the one used in [26, 32] for constructing a weighted finite automaton to recognize the neighbourhood.)

The choice of the set of final states guarantees that B accepts a string $u \in \Sigma^*$ iff $u \in E(L(A), d, r)$. □

When r and n are at least two, the upper bound of Theorem 5 is better than the upper bound of Theorem 4. The natural question is then whether the upper bound can be reached. Below we give a positive answer to this question. However, a limitation is that the size of the alphabet depends on the size of the DFA and the used (quasi-)distance needs to be defined based on the chosen radius.

Proposition 1 *(*[27]*)*

(i) *For all* $n, r \geq 1$*, there exists an additive distance* d_r *and an NFA* A_n *with* n *states over an alphabet of size* $2n - 1$ *such that* $\mathrm{sc}(E(L(A_n), d, r) = (r + 2)^n - 1$.

(ii) *For all* $n, r \geq 1$*, there exists an additive quasi-distance* d'_r *and a DFA* A'_n *with* n *states over an alphabet of size* $3n - 2$ *such that* $\mathrm{sc}(E(L(A'_n), d', r) = (r + 2)^n - 1$.

Proof The constructions for (i) and (ii) are variants of each other and (ii) is presented in detail in [27]. Here we outline the construction for (i).

Choose $\Sigma_n = \{a_1, \ldots, a_{n-1}, b_1, \ldots, b_n\}$. For $r \in \mathbb{N}$, we define a 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 $i \neq j$,
- $d_r(a_i, b_j) = r + 1$ for all $1 \leq i, j \leq n$,
- $d_r(\sigma, \epsilon) = r + 1$ for all $\sigma \in \Sigma$.

The above conditions specify a unique additive distance on Σ_n. Note that when we set the deletion cost to be $r + 1$, due to symmetry of d_r also the insertion cost will be $r + 1$.

We define the following family of n-state NFAs $A_n = (Q_n, \Sigma_n, \delta, 1, \{n\})$ where $Q_n = \{1, \ldots, n\}$ and the transition function δ is defined by setting

- $\delta(i, a_i) = \{i, i + 1\}$ 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 $i + 1 \leq j \leq n$.

All transitions not listed above are undefined. The NFA A_n is depicted in Fig. 1.

As in Corollary 1 we construct an NFA $B_{n,r} = (Q'_n, \Sigma_n, \delta', q'_0, F')$, for the neighbourhood $E(L(A_n), d_r, r)$, where $Q'_n = Q_n \times \{0, 1, \ldots, r\}$, $q'_0 = (q_0, 0)$, $F' = \{n\} \times \{0, 1, \ldots, r\}$ and the transition function δ' is defined by

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

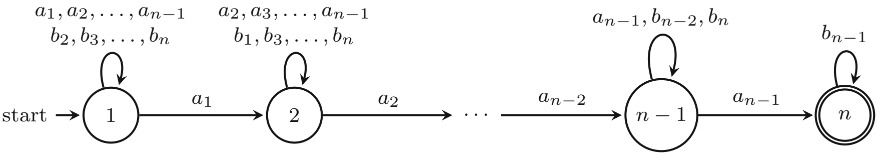

Fig. 1 The NFA A_n

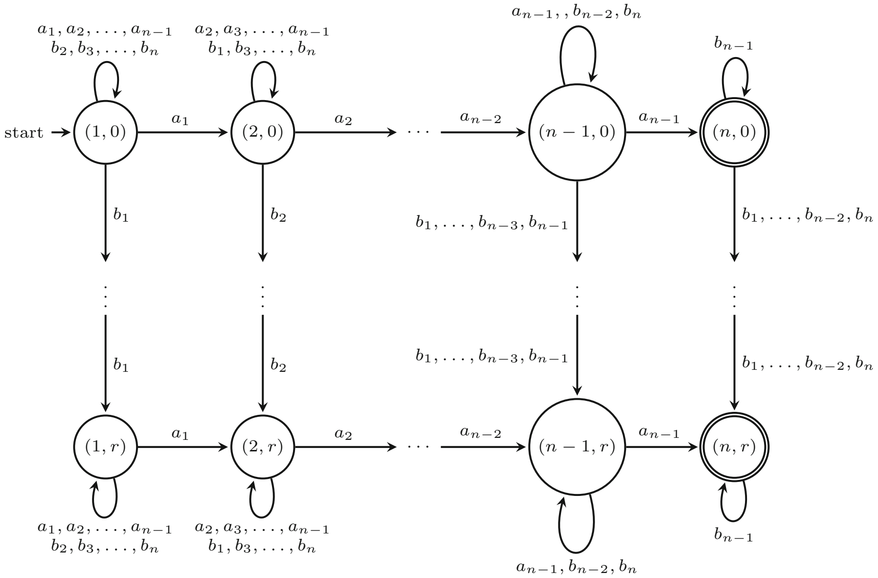

Fig. 2 The NFA $B_{n,r}$ for the neighbourhood $E(L(A_n), d_r, r)$

All transitions not listed above are undefined. The NFA $B_{n,r}$ is depicted in Fig. 2.

Note that the distance d_r associates cost one to substituting b_i with b_j, $i \neq j$, and cost $r + 1$ to all other substitutions, insertions and deletions. The "error transitions" are depicted as non-horizontal transitions in Fig. 2 and, due to the above observation, the only error transitions take a state (i, k) to $(i, k + 1)$ on symbol b_z, $1 \leq i \leq n$, $0 \leq k \leq r - 1$, where in the NFA A_n, $\delta(i, b_z)$ is undefined.

For $0 \leq k_i \leq r + 1$, $1 \leq i \leq n$, we define a string

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

Claim 1. 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 $w(k_1, \dots, k_n)$, $1 \leq i \leq n$. There is no computation of $B_{n,r}$ on $w(k_1, \dots, k_n)$ 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 $w(k_1, \dots, k_n)$ a state where the first component is i.

The first part of the claim is easy to verify by direct inspection: C_i reaches state $(i, 0)$ by reading the prefix $a_1 b_1^{k_1} \dots a_{i-1} b_{i-1}^{k_{i-1}}$. In state $(i, 0)$ the computation C_i reads a_i with the selfloop and the vertical error transitions on $b_i^{k_i}$ take the computation to state (i, k_i) where the remaining suffix can be processed using selfloops. For the second part of the claim we note that all horizontal transitions in $B_{n,r}$ are labeled by a_j's, and all horizontal transitions increment the first component of the state. Thus, the only way to reach a state (i, k_i'), would be to read each of the symbols a_1, ...,

a_{i-1} using a transition that increments the first component and then read a_i with a selfloop. Now the following k_i symbols b_i must be processed using error transitions which means that k_i' cannot be smaller than k_i.

Using Claim 1 we can now verify any two distinct strings $w(k_1, \ldots, k_n)$ and $w(k_1', \ldots, k_n')$, $0 \leq k_i, k_i' \leq r + 1$, are inequivalent with respect to the Kleene congruence of $E(L(A_n), d_n, r)$. Choose $1 \leq j \leq n$ such that $k_j < k_j'$ and define $z = b_j^{r-k_j} a_{j+1} \ldots a_{n-1}$.

By Claim 1, $w(k_1, \ldots, k_n) \cdot z \in L(B_n, r)$ because after reading $w(k_1, \ldots, k_n)$ the NFA $B_{n,r}$ can reach the state (j, k_j) and continuing the computation on z can make $r - k_j$ further error transitions. Similarly using Claim 1 we see that no computation of $B_{n,r}$ on $w(k_1', \ldots, k_n') \cdot z$ cannot reach an accepting state where the first component is n, because to do so it would need to reach on the prefix $w(k_1', \ldots, k_n')$ a state (j, ℓ) where $\ell \leq k_j$. Since $k_j < k_j'$ this is impossible by Claim 1.

Since $L(B_{n,r}) = E(L(A_n), d_n, r)$ it follows that the minimal complete DFA for $E(L(A_n), d_n, r)$ has at least $(r + 2)^n$ states. The congruence class of $w(r + 1, \ldots, r + 1)$ corresponds to the dead state of the DFA and can be omitted.

The proof for the part (ii) in [27] introduces additional alphabet symbols $c_1, \ldots, c_{n-1}$ and in each nondeterministic transition of A_n on a_i, in the DFA A_n' the selfloop is labeled instead by c_i, $1 \leq i \leq n - 1$. The quasi-distance d_r' assigns distance zero to a_i and c_i, $1 \leq i \leq n - 1$. With these definitions a lower bound argument for the size of a DFA for $E(L(A_n'), d_r', r)$ similar to the one used above for (i) goes through. □

To conclude this section we mention that the construction of Proposition 1 can be modified to yield a tight lower bound for the state complexity of approximate pattern matching. The descriptional complexity of pattern matching with mismatches was first considered by El-Mabrouk [10]. Given a pattern P of length m and a text T, the problem is to determine whether or not 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 $\binom{m+1}{r+1}$ [10].

Extending the problem for a general additive quasi-distance d and a set of patterns given as a regular language $L \subseteq \Sigma^*$, we want to determine the state complexity of the set $\Sigma^* \cdot E(L, d, r) \cdot \Sigma^*$, that is, the set of strings that contain a substring within distance at most r from a string of L. The following lower bound is based on a modification of the construction used in Proposition 1.

Proposition 2 *([27]) For $n, r \in \mathbb{N}$, there exist an additive distance d and an NFA A with n states defined over an alphabet 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.*

The authors [27] give an upper bound matching the bound of Proposition 2 which means that the state complexity of approximate pattern matching with r errors is exactly $(r + 2)^{n-2} + 1$. 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 $2^{n-2} + 1$

which corresponds to having error radius zero in approximate pattern matching. The lower bound for the error free case is obtained with a three letter alphabet [3] whereas Proposition 2 needs a variable size alphabet.

4.2 State Complexity of Prefix Distance

Additivity is not a necessary condition for a distance to be regularity preserving. For example, by Theorem 2 the prefix, suffix, and substring distances preserve regularity while these distances clearly are not additive.

The neighbourhood $E(L, d_p, r)$ (where d_p is the prefix distance and $r \geq 0$) consists of strings w that share a "long" prefix with a string $u \in L$, more precisely, it is required that the combined length of the parts of w and u outside their longest common prefix is at most the constant r. In view of this, it seems reasonable to expect that the state complexity prefix distance neighbourhoods does not incur a similar exponential size blow-up as, for example, the edit distance.

Theorem 6 *([28]) For $n > r \geq 0$ and a DFA A with n states, the neighbourhood $E(L(A), d_p, r)$ can be recognized by a DFA with $n \cdot (r+1) - \frac{r(r+1)}{2}$ states.*

For $n > r \geq 0$ there exists a regular language L over an alphabet of size $n+1$ with $\mathrm{sc}(L) = n$ *such that*

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

Proof sketch. We outline the general idea only for the upper bound [28]. Suppose $A = (\Sigma, Q, \delta, q_0, F_A)$. We can construct for the neighbourhood $E(L(A), d_p, r)$ a DFA B with state set

$$P = (Q - F_A) \times \{1, \ldots, r+1\} \cup F_A \cup \{p_1, \ldots, p_r\}.$$

Intuitively, B operates as follows. The computation of B simulates the computation of A and, in states $(q, j) \in (Q - F_A) \times \{1, \ldots, r+1\}$ the second component j keeps track of the minimum of the following two values: (i) the number of steps A needs q to reach a final state, and, (ii) the minimum path length in A from q to a final state that first goes one or more steps back in the current computation and then any number of steps forward (on an arbitrary input). Elements of F_A have always counter value zero and, hence, are not associated with the second component representing a counter. The details of the definition of the transitions of B that correctly update the counter value in the second component of the states can be found in [28].

If the simulated computation of A encounters an undefined transition, B performs at most r further transitions using a sequence of "error-transitions" using the states $p_1, \ldots, p_r$. The number of allowable error transitions depends on the value of the counter when the undefined transition of A was encountered.

All states of B except the states $(q, r+1)$, $q \in Q - F$, with counter value $r+1$, are final. Note that the states of the form $(q, r+1)$, $q \in Q - F$, are needed because when simulating the transitions of A in the first component the counter value may also decrease if the "forward" distance in A to a final state becomes smaller.

The state set P of B has in total $(n - |F_A|) \cdot (r+1) + r + |F_A|$ elements. The size of P is maximized by choosing $|F_A| = 1$ and, furthermore, it can be verified that at least $\frac{r(r+1)}{2}$ elements of P must, independently of the alphabet size, be unreachable as states of B. □

The lower bound construction for Theorem 6 uses an alphabet of size $n+1$ where n is the number of states of the original DFA. It is known that the general upper bound cannot be reached using an alphabet of size $n-2$.

Proposition 3 *([28]) Let A be a DFA with n states. If the state complexity of $E(L(A), d_p, r)$ equals $n \cdot (r+1) - \frac{r(r+1)}{2}$, then the alphabet of A needs at least $n-1$ letters.*

The paper [28] gives tight bounds also for the nondeterministic state complexity of neighbourhoods defined by the prefix, suffix, and substring distances. The bounds for the nondeterministic state complexity of the prefix distance and suffix distance, respectively, coincide due to the observation that $d_s(x, y) = d_p(x^R, y^R)$ for all strings x, y, and the fact that the transitions of an NFA can be reversed without changing the size of the NFA which means that, for any regular language L, $\text{nsc}(L) = \text{nsc}(L^R)$.

On the other hand, the deterministic state complexity of L^R is usually significantly different from the state complexity of L [11, 37]. It seems likely that constructing a DFA for the neighbourhood of an n-state DFA with respect to the suffix distance causes a much larger worst-case size blow-up than the bound for prefix distance in Theorem 6. The precise deterministic state complexity of the suffix distance remains open.

5 Conclusion and Open Problems

The precise worst-case state complexity of the radius r neighbourhood of an n state DFA language with respect to an additive quasi-distance is $(r+2)^n - 1$. However, the lower bound construction of Proposition 1 has the following limitations:

- The construction uses an alphabet that depends linearly on the number of states of the original DFA.
- The underlying distance is defined based on the radius of the neighbourhood.
- We don't have a tight bound for state complexity defined in terms of complete DFAs. A complete DFA with $(r+2)^n$ states can recognize a radius r neighbourhood of an n state DFA. It is not known how to construct a complete n state DFA matching this upper bound.

The main open problem consists of proving lower bounds for additive distances (or quasi-distances) using languages over a binary, or constant size, alphabet. The known good lower bound construction based on binary alphabets, due to Povarov [30], deals only with the restricted case of radius one Hamming neighbourhoods (Theorem 4). The other important improvement to the lower bound result of Proposition 1 would be to find a construction where the same distance (or quasi-distance) definition works for neighbourhoods of arbitrary radius.

Descriptional complexity questions are relevant also for the more general error channels considered by Kari and Konstantinidis [13] and Konstantinidis and Silva [19]. As briefly discussed in Sect. 3, the edit distance of two strings being at most a constant r can be defined in terms of an error system that allows at most r substitution, insertion and deletion errors. With respect to this channel, the set of possible outputs for an input belonging to a regular language L consists of the edit distance neighbourhood of L having radius r.

General error channels (or error systems) realized by rational channels [9, 13, 19] can formalize many further types of errors, such as transposition errors, or so called scattered or burst errors that are relevant for data communication applications. The set of possible outputs $\mathcal{C}(L)$ produced by a rational error channel $\mathcal{C}$ corresponding to inputs belonging to a regular language L is always regular. However, the set $\mathcal{C}(L)$ need not be a neighbourhood of L defined by a distance metric and future work can consist to determine the state complexity of $\mathcal{C}(L)$ as a function of the state complexity of L and the size of a finite transducer realizing the error channel $\mathcal{C}$. The descriptional complexity of error systems has been considered from a different point of view by Kari and Konstantinidis [13] who establish upper and lower bounds for the sizes of DFAs that recognize a given error system,

References

1. Benedikt, M., Puppis, G., Riveros, C.: Bounded repairability of word languages. J. Comput. Syst. Sci. **79**, 1302–1321 (2013)
2. Benedikt, M., Puppis, G., Riveros, C.: The per-character cost of repairing word languages. Theoret. Comput. Sci. **539**, 38–67 (2014)
3. Brzozowski, J., Jirásková, G., Li, B.: Quotient complexity of ideal languages. In: Latin American Theoretical Informatics Symposium, pp. 208–221 (2010)
4. Calude, C.S., Salomaa, K., Yu, S.: Additive distances and quasi-distances between words. J. Univers. Comput. Sci. **8**, 141–152 (2002)
5. Chatterjee, K., Henzinger, T.A., Ibsen-Jensen, R., Otop, J.: Edit distance for pushdown automata. In: 42nd ICALP, Proceedings. Part II. Lecture Notes in Computer Science, vol. 9135, pp. 121–133 (2015)
6. Choffrut, C., Pighizzini, G.: Distances between languages and reflexivity of relations. Theoret. Comput. Sci. **286**, 117–138 (2002)
7. Cohen, G., Honkala, I., Litsyn, S., Lobstein, A.: Covering Codes. Elsevier North-Holland Mathematical Library, vol. 54 (1997)
8. Deza, M.M., Deza, E.: Encyclopedia of Distances. Springer, Berlin Heidelberg (2009)
9. Dudzinski, K., Konstantinidis, S.: Formal descriptions of code properties: decidability, complexity, implementation. Int. J. Found. Comput. Sci. **23**, 67–85 (2012)
10. 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)

11. Gao, Y., Moreira, N., Reis, R., Yu, S.: A survey on operational state complexity. arXiv:1509.03254v1 [cs.FL], Sept 2015. (To appear in *Computer Science Review*.)
12. Han, Y.-S., Ko, S.-K., Salomaa, K.: The edit-distance between a regular language and a context-free language. Int. J. Found. Comput. Sci. **24**, 1067–1082 (2013)
13. Kari, L., Konstantinidis, S.: Descriptional complexity of error/edit systems. J. Autom. Lang. Comb. **9**(2/3), 293–309 (2004)
14. Kari, L., Konstantinidis, S., Kopecki, S., Yang, M.: An efficient algorithm for computing the edit distance of a regular language via input-altering transducers (2014)
15. Konstantinidis, S.: An algebra of discrete channels that involve combinations of three basic error types. Inform. Comput. **167**, 120–131 (2001)
16. Konstantinidis, S.: Transducers and the properties of error-detection, error-correction, and finite-delay decodability. J. Univers. Comput. Sci. **8**, 278–291 (2002)
17. Konstantinidis, S.: Computing the edit distance of a regular language. Inform. Comput. **205**, 1307–1316 (2007)
18. Konstantinidis, S., Silva, P.: Maximal error-detecting capabilities of formal languages. J. Autom. Lang. Comb. **13**(1), 55–71 (2008)
19. Konstantinidis, S., Silva, P.V.: Computing maximal error-detecting capabilities and distances of regular languages. Fund. Inform. **101**, 257–270 (2010)
20. Kutrib, M., Meckel, K., Wendlandt, M.: Parameterized prefix distance between regular languages. In: SOFSEM 2014: Theory and Practice of Computer Science, pp. 419–430 (2014)
21. Leung, H., Podolskiy, V.: The limitedness problem on distance automata: Hashiguchi's method revisited. Theoret. Comput. Sci. **310**, 147–158 (2004)
22. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. **10**(8), 707–710 (1966)
23. Lothaire, M.: Applied combinatorics on words, Chapter 1 algorithms on words. In: Encyclopedia of Mathematics and It's Applications, vol. 105. Cambridge University Press, New York (2005)
24. Mohri, M.: Edit-distance of weighted automata: general definitions and algorithms. Int. J. Found. Comput. Sci. **14**(6), 957–982 (2003)
25. Morelos-Zaragoza, R.H.: The Art of Error Correcting Coding. John Wiley & Sons, Chichester, England (2006)
26. Ng, T., Rappaport, D., Salomaa, K.: Quasi-distances and weighted finite automata. In: Proceedings of DCFS 2015, Lecture Notes Computer Science, vol. 9118, pp. 209–219 (2015)
27. Ng, T., Rappaport, D., Salomaa, K.: State complexity of neighbourhoods and approximate pattern matching. In: Potapov I., (ed.) Proceedings of DLT 2015. Lecture Notes Computer Science, vol. 9168, pp. 389–400 (2015)
28. Ng, T., Rappaport, D., Salomaa, K.: State complexity of prefix distance. In: Proceedings of CIAA 2015. Lecture Notes Computer Science, vol. 9223, pp. 238–249 (2015)
29. Pighizzini, G.: How hard is computing the edit distance? Inform. Comput. **165**(1), 1–13 (2001)
30. Povarov, G.: Descriptive complexity of the Hamming neighborhood of a regular language. In: Language and Automata Theory and Applications, pp. 509–520 (2007). (An updated version available at: http://gp-sci.googlecode.com/svn/trunk/phd/.../HammingNeighborhood.pdf)
31. Povarov, G.: Finite transducers and nondeterministic state complexity of regular languages. Russ. Math. (Iz. VUZ) **54**(6), 19–25 (2010)
32. Salomaa, K., Schofield, P.: State complexity of additive weighted finite automata. Int. J. Found. Comput. Sci. **18**(6), 1407–1416 (2007)
33. Shamkin, S.: Descriptional complexity of Hamming neighbourhoods of finite languages (in Russian). M.Sc. Thesis, Ural Federal University, Ekaterinburg, Russia (2011)
34. Schulz, K.U., Mihov, S.: Fast string correction with Levenshtein automata. Int. J. Doc. Anal. Recogn. **5**, 67–85 (2002)
35. Shallit, J.: A Second Course in Formal Languages and Automata Theory, Cambridge University Press (2009)
36. van Lint, J.H.: Introduction to Coding Theory. Springer, Graduate Texts in Mathematics (1999)
37. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A., (eds.) Handbook of Formal Languages, pp. 41–110. Springer, Berlin, Heidelberg (1997)