Computing Maximal Error-detecting Capabilities and Distances of Regular Languages[1]

Stavros Konstantinidis[†] and Pedro V. Silva[‡]

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

[‡]Centro de Matemática, Faculdade de Ciências
Universidade do Porto – R. Campo Alegre 687
4169-007 Porto, Portugal
pvsilva@fc.up.pt

Abstract. A (combinatorial) channel consists of pairs of words representing all possible input-output channel situations. In a past paper, we formalized the intuitive concept of "largest amount of errors" detectable by a given language L, by defining the maximal error-detecting capabilities of L with respect to a given class of channels, and we showed how to compute all maximal error-detecting capabilities (channels) of a given regular language with respect to the class of rational channels and a class of channels involving only the substitution-error type. In this paper we resolve the problem for channels involving errors of any combination of the basic types substitution, insertion, deletion. Moreover, we consider the problem of finding the inverses of these channels, in view of the fact that L is error-detecting for γ if and only if it is error-detecting for the inverse of γ. We also discuss a natural method of reducing the problem of computing (inner) distances of a given regular language L to the problem of computing maximal error-detecting capabilities of L.

Key words: algorithm, automaton, combinatorial channel, edit string, error detection, maximal, regular language, string distance.

1 Introduction

A (combinatorial) channel consists of pairs of words describing all input-output situations permitted by the channel. The fact that (w, z) is in the channel means that the word z can be received via the channel when the word w is used as input. Then, if $w \neq z$, we say that z contains errors, or that w was received with errors. A language L is *error-detecting for* some channel γ, [6], if there is no pair (w, z) in the channel such that $w \neq z$ and both w and z are in L_λ; that is, the channel cannot transform a word of L_λ into a different word of L_λ. This fact allows one to detect that a received word z contains errors *exactly* when z is not in L_λ. Here λ is the empty word and $L_\lambda = L \cup \{\lambda\}$.

An *error model* is a set $\mathbb{C}$ of channels. Intuitively, $\mathbb{C}$ contains the possible channels that appear to model the error situations arising in some application where information needs to be transmitted or stored. In [8] we introduced the concept of *maximal error-detecting capability* of a language L, with respect to a certain error model $\mathbb{C}$. This is a channel γ in $\mathbb{C}$ such that L is error-detecting for γ and L is not error-detecting for γ', for any channel γ' in $\mathbb{C}$ that properly contains γ. In [8] we posed the question of computing the maximal error-detecting capabilities of a given regular language for

[1]Research supported by (i) a Discovery Research Grant of NSERC, Canada, and (ii) Project ASA (PTDC/MAT/65481/2006) and C.M.U.P., financed by F.C.T. (Portugal) through the programmes POCTI and POSI, with national and European Community structural funds.

various error models, including the rational channels as well as various models of *SID channels.* Rational channels are exactly those realized by finite transducers. Informally, an SID channel is specified by an expression of the form $\tau(m,l)$, where τ is an error type and m,l are nonnegative integers with $m<l$. This channel consists of all pairs (w,z) of words such that z can result if we apply at most m errors of type τ in any segment of w of length l. An *error type* is an expression in

$$\{\sigma,\iota,\delta,\sigma\odot\iota,\sigma\odot\delta,\iota\odot\delta,\sigma\odot\iota\odot\delta\}$$

The three simple error types σ,ι,δ denote *substitutions, insertions,* and *deletions*, respectively. The symbol $\odot$ is used simply as a connective for the simpler types. In [8] we showed how to compute all maximal error-detecting capabilities of a given regular language with respect to the error-model

$$\mathbb{C}_\tau=\{\tau(m,l):\ \text{for any } m \text{ and } l \text{ with } 0\le m<l\}$$

when $\tau=\sigma$, and we left open the problem when $\tau\neq\sigma$. In this paper we settle this problem. In doing so, we also settle all containment relationships between any two channels $\tau(m_1,l_1)$ and $\tau(m_2,l_2)$ and, in particular, we show that two such channels are equal if and only if $m_1=m_2$ and $l_1=l_2$ (for any error type τ).

A consequence of the above result is that the SID-maximal error-detecting capabilities of a given regular language L are computable as well, that is, the maximal error-detecting capabilities of L with respect to the error model

$$\text{SID}=\{\tau(m,l):\text{for any error type } \tau \text{ and } m,l \text{ with } 0\le m<l\}.$$

We also consider the problem of finding the inverses of channels $\tau(m,l)$, in view of the fact that L is error-detecting for a channel γ if and only if it is error-detecting for the inverse of γ. Counter to one's intuition, we show that the channels $(\iota\odot\delta)(m,l)$ and $(\sigma\odot\iota\odot\delta)(m,l)$ are not symmetric, and only the inverses of $\iota(m,l)$ and $\delta(m,l)$ are channels of the form $\tau(m',l')$. This implies that, in computing maximal error-detecting capabilities for the SID error model, one needs to consider six of the seven error types.

Finally, we note a simple and natural connection between (string) distances and combinatorial channels. In particular, for every distance d taking only integer values, we can define an error model $\mathbb{C}_d=\{\gamma_d(0),\gamma_d(1),\ldots\}$ such that the following holds: for any language L, the (inner) distance $d(L)$ of L is $1+m$, where m is the largest value for which $\gamma_d(m)$ is the maximal error-detecting capability of L with respect to $\mathbb{C}_d$. This observation provides a method for computing the distance of a given regular language.

The paper is organized as follows. In the next section we give some basic notions and notation, including a formal definition for SID channels via the concept of *edit string.* In Section 3, we prove a few basic combinatorial results about SID channels and error-detection. These results are needed to establish the correctness of the algorithms of Section 4 for computing maximal error-detecting capabilities of a given regular language with respect to SID-channel models. Moreover, the last subsection of Section 4 discusses the method of maximal error-detecting capabilities for computing distances of regular languages. Finally, Section 5 contains a few concluding remarks.

2 Basic definitions and background

We use the symbol Σ to denote a fixed, but arbitrary, *alphabet* containing at least the two symbols 0 and 1. The symbol Σ^* denotes the set of all *words* (or strings) over Σ, including the empty word λ. The *length* of a word w is denoted by $|w|$. If w can be written as xvy then the word v is called a *factor* of w. A *language* is any set of words. If K is a language then K_λ denotes the language $K\cup\{\lambda\}$. A binary relation over the alphabet Σ is a subset γ of $\Sigma^*\times\Sigma^*$. The domain of γ, denoted as $\operatorname{dom}\gamma$, is the set of all words w such that $(w,z)\in\gamma$ for some word z. The inverse of γ, denoted by γ^{-1}, is the relation consisting of all (z,w) such that (w,z) is in γ. A *(combinatorial) channel* is a binary relation γ over Σ that is *domain preserving*, that is, γ contains the pair (w,w) for all w in $\operatorname{dom}\gamma$. This requirement ensures that error-free communication is always possible via the channel. When $(w,z)\in\gamma$, we say that z is a possible output of the channel when w is used as input.

In the sequel we shall consider mainly *regular languages*, that is, languages accepted by finite automata, and *SID channels*. Recall that a finite automaton has finitely many states, one of which is its initial state and some of them are its final states, and a set of transitions of the form (p,a,q). Such a transition says that if the automaton is in state p and the current input starts with $a\in\Sigma$ then it can enter the state q. The automaton can consume a given input word by following its transitions and, in this case, *accepts* the word when it is in some final state. The set of words accepted by a finite automaton A is denoted by $L(A)$ – see [15], for instance, for more details on finite automata and transducers.

An *edit string* is a word over the alphabet E of *edit operations*

$$E=\{(x/y):x,y\in\Sigma\cup\{\lambda\}\ \text{ and }\ xy\neq\lambda\}.$$

The empty word of E^* is denoted with (λ/λ). If the edit operation (x/y) is such that $x\neq y$ then (x/y) is called an *error*. In particular, if $x=\lambda$ then (x/y) is an insertion error, if $y=\lambda$ then it is a deletion error, otherwise it is a substitution error. For an error type τ, we write E_τ for the subset of E containing all non-errors and all errors of type τ. For example,

$$E_\delta=\{(x/x):x\in\Sigma\}\cup\{(x/\lambda):x\in\Sigma\}.$$

An edit string $(x_1/y_1)\cdots(x_n/y_n)$ describes a possible sequence of edit operations that can be used to transform the word $x_1\cdots x_n$ to the word $y_1\cdots y_n$. For example, the edit string

$$h=(0/0)(0/1)(0/0)(0/\lambda)(0/0)(0/0)(\lambda/1)$$

transforms the word $w=000000$ to the word $z=010001$. In this case, we say that w is the *input part* and z is the *output part* of h, and we use the notation $h\pi_1$ and $h\pi_2$ for the input and output parts of h, respectively. The *weight* $|h|_{\neq}$ of an edit string h is the number of errors in h – this is equal to 3 in the above h. The *input size* of an edit string is the length of its input part, that is, the quantity $|h\pi_1|$ – this is equal to 6 in the above h, whereas its length is 7.

An *edit system* [5] is a set H of edit strings, that is, a subset of E^*. The relation $\operatorname{ch}(H)$ defined by H is

$$\operatorname{ch}(H)=\{(u,v):u=h\pi_1\text{ and }v=h\pi_2,\ \text{ for some }h\in H\}.$$

We shall define SID channels via edit systems. Informally, the SID channel $\tau(m,l)$ consists of all pairs (w,z) such that z results by applying at most m errors in any factor of w of length l [10]. Moreover, any insertion error occurring to the left of an input symbol counts as part of that symbol. We formalize SID channels using the concept of *l-segment* of an edit string h. This is a factor g of $h(1/1)^l$ of input size equal to l such that g ends with no insertion errors. For example, the edit strings $(0/\lambda)(0/0)(0/0)(\lambda/1)(1/1)$ and $(0/0)(0/\lambda)(0/0)(0/0)$ are 4-segments of the h displayed above.

An SID edit system $H_{\tau,m,l}$ is defined using three parameters, an error type τ and two nonnegative integers m and l, with $m<l$, as follows

$$H_{\tau,m,l}\ =\ \{h\in E_\tau^*:\ \text{if } g \text{ is an } l\text{-segment of } h \text{ then } |g|_{\neq}\le m\}.\tag{1}$$

For example, the edit string h displayed above is in both, $H_{\sigma\odot\iota\odot\delta,3,6}$ and $H_{\sigma\odot\iota\odot\delta,2,4}$, but not in $H_{\sigma\odot\iota\odot\delta,2,6}$ as the 6-segment $(0/1)(0/0)(0/\lambda)(0/0)(0/0)(\lambda/1)(1/1)$ of h contains more than 2 errors. Using SID edit systems we can formally define the *SID channel* $\tau(m,l)$ as follows

$$\tau(m,l) \triangleq \mathrm{ch}(H_{\tau,m,l}).$$

Remark 1 *For every edit string $h \in H_{\tau,m,l}$, it is easy to confirm that (i) every factor of h is in $H_{\tau,m,l}$, (ii) $(a/a)^m h(b/b)^n \in H_{\tau,m,l}$ for all $a,b \in \Sigma$ and $m,n \geq 0$, (iii) for every factor g of h, if the input size of g is $< l$, or if $|g| < l$, then $|g|_{\neq} \leq m$.*

We note that $H_{\tau,m,l}$ can be defined by requiring that, in (1), g is simply a factor of h of input length at most l – as in [5]. Although this is slightly different in theory, it makes very little difference in practice. Moreover, some results like Lemma 3 are rather simpler with the present definition.

3 Combinatorial Results

In this section we derive a few combinatorial results – in particular Theorems 1,2,3 – that will be used to establish the correctness of our algorithms in Section 4 for computing maximal error-detecting capabilities with respect to SID channels.

3.1 A channel bound for error-detectability

Lemma 1 *For all words u, v, u', v', if $uv \neq u'v'$ then, for every word w, $uwv \neq u'wv'$ or $uw^2v \neq u'w^2v'$.*

Proof. We view the set of all words Σ^* as a submonoid of the free group $\Sigma^{\circledast}$, [1], where we write w^{-1} for the inverse of the word w. Let $x = u'^{-1}u$ and $y = v'v^{-1}$, and suppose, for the sake of contradiction, that there is a word w such that $uwv = u'wv'$ and $uw^2v = u'w^2v'$. Then, $xwy^{-1} = u'^{-1}uwvv'^{-1} = w$ and similarly $xw^2y^{-1} = w^2$. Hence,

$$wywy^{-1} = xwy^{-1}ywy^{-1} = w^2,$$

which implies that $ywy^{-1} = w$, that is, $y^{-1}w = wy^{-1}$. Thus, $xwy^{-1} = w$ yields $xy^{-1} = \lambda \Rightarrow x = y \Rightarrow u'^{-1}u = v'v^{-1} \Rightarrow uv = u'v'$, a contradiction. ■

Theorem 1 *Let τ be an error type, other than σ, let m be a positive integer, and let A be a finite automaton accepting at least two words, including the empty word. The language $L(A)$ is error-detecting for the channel $\tau(m,l)$, for some $l > m$, if and only if it is error-detecting for $\tau(m, 1 + s_A^2(m+1))$, where s_A is the number of states in A.*

For the proof of the theorem we shall need the automaton A_τ, which is constructed from the automaton A and the error type τ as follows:

1. The set of states of A_τ is $\{(p,q) : p, q \text{ are states of } A\}$.
2. The alphabet of A_τ consists of all edit operations $(x/x') \in E$ of type τ.
3. The set T of transitions is as follows: For every two transitions (p_1, a_1, q_1) and (p_2, a_2, q_2) of the automaton A, add in T the transitions $((p_1,p_2),(a_1/a_2),(q_1,q_2))$, $((p_1,p_2),(\lambda/a_2),(p_1,q_2))$, and $((p_1,p_2),(a_1/\lambda),(q_1,p_2))$ whose label is of type τ or of the form (x/x).

4. The start state of A_τ is the pair (s, s), where s is the start state of A. The final states of A_τ are all pairs (p, p'), where p and p' are final states in A.

It is not difficult to verify that the language accepted by A_τ consists of all edit strings h containing only errors of type τ (if any) such that $h\pi_1, h\pi_2 \in L(A)$.

Proof of Theorem 1. The 'if' part is trivial. For the 'only if' part, suppose that $L(A)$ is error-detecting for the channel $\tau(m,l)$, for some $l > m$. Let $k = 1 + s_A^2(m+1)$. We shall obtain a contradiction by assuming that $L(A)$ is not error-detecting for $\tau(m,k)$. Thus, there exist two distinct words u and v in $L(A)$ such that $(u,v) \in \tau(m,k)$ – and $(u,v) \notin \tau(m,l)$. Moreover, there is an edit string $h \in H_{\tau,m,k}$ such that $u = h\pi_1$ and $v = h\pi_2$. We establish the following claim.

Claim 1: $|h| > k$. Indeed, first note that $|h|_{\neq} > m$, as otherwise $(u,v) \in \tau(m,l)$. Then, as $h \in H_{\tau,m,k}$ and h contains more than m errors, we have that the claim holds.

Let $B = \{(a/a) : a \in \Sigma\}$ = all edit operations that are not errors. Consider the automaton A_τ and *define the set* Q'_0 of states p in this automaton such that there is a path from p to p in which the edit string formed along the path is in B^+. Let P be an accepting path $(q_0, h_1, q_1, \ldots, h_t, q_t)$ in A_τ such that $h = h_1 \cdots h_t$. By Claim 1, $t > k$. We establish the following two claims.

Claim 2: Every part P' of P of the form

$$P' = (q_i, h_{i+1}, q_{i+1}, \ldots, h_{i+k-1}, q_{i+k-1})$$

contains a state $q_r \in Q'_0$ for some r with $i < r < i+k-1$. Let $g = h_{i+1} \cdots h_{i+k-1}$. As the length of g is less than k, Remark 1 implies that $|g|_{\neq} \le m$. So $g = g_0 e_1 g_1 \cdots e_s g_s$, where each e_j is an error and each g_j is in B^* and $0 \le s \le m$. As $|g_0 \cdots g_s| = |g| - s = k - 1 - s$, it follows that $|g_0 \cdots g_s| > (s_A^2 - 1)(s+1)$, which implies that $|g_j| \ge s_A^2$ for some index j. Then the path P' contains a part P'' of the form

$$(q_\phi, h_{\phi+1}, q_{\phi+1} \ldots, h_{\phi+s_A^2}, q_{\phi+s_A^2}),$$

with each h_i being in B. Moreover, as P'' involves more than s_A^2 states, there are two indices θ, θ' such that $\phi \le \theta < \theta' \le \phi + s_A^2$ and $q_\theta = q_{\theta'}$. As $h_{\theta+1} \cdots h_{\theta'} \in B^+$, we have that $q_\theta \in Q'_0$. If $\theta' < i+k-1$ then $i \le \theta < \theta'$ and we define $r = \theta'$. If $\theta' = i+k-1$ then $\theta < \theta' \le i+k-1$ and $\theta \ge \theta' - s_A^2 > i$, and we define $r = \theta$. In either case, $q_r \in Q'_0$ and $i < r < i+k-1$, as required.

Claim 3: In the path P of A_τ, there are $n+1$ states $p_0, \ldots, p_n$, with $n \ge 2$, such that $p_0 = q_0$, $p_n = q_t$, every state p_j with $j \in \{1, \ldots, n-1\}$ is in Q'_0, and $|g_i| \le k-1$ for all $i \in \{1, \ldots, n\}$, where each g_i is the edit string formed in P between the states p_{i-1} and p_i. For the correctness of this claim, we consider the part

$$(q_0, h_1, q_1, \ldots, h_{k-1}, q_{k-1})$$

of P which, by Claim 2, contains a state $q_r \in Q'_0$ with $0 < r < k-1$, hence $0 < r < t$. Let $p_1, \ldots, p_{n-1}$ be the sequence of all the states in Q'_0, other than q_0 and q_t, as they appear in the path P, and let g_i be the edit string that is formed in P between the states p_{i-1} and p_i, where we let $p_0 = q_0$ and $p_n = q_t$. If any of the g_i's were of length greater than $k-1$ then, by Claim 2, there would be states in Q'_0 occurring in the path P, other than the p_j's, which is a contradiction. Hence, $|g_i| \le k-1$ for all i, as required.

The above claim implies that the edit string h can be written as

$$h = g_1 \cdots g_n.$$

Moreover, the same claim implies that, for every $j \in \{1, \ldots, n-1\}$, there is a path in the automaton A_τ from state p_j to p_j with some label f_j of length $|f_j| \geq l$. Then, also f_j^2 is the label of a path from p_j to p_j. Thus, any edit string of the form

$$f = g_1 f_1^{m_1} g_2 \cdots f_{n-1}^{m_{n-1}} g_n, \text{ where each } m_j \in \{1, 2\},$$

is in $L(A_\tau)$, which implies that $f\pi_1, f\pi_2 \in L(A)$. As $h\pi_1 \neq h\pi_2$, successive applications of Lemma 1 imply that there is a choice of $m_1, \ldots, m_{n-1}$ such that $f\pi_1 \neq f\pi_2$. Moreover, each $f_j^{m_j} \in B^+$. We shall derive a contradiction by showing that $(f\pi_1, f\pi_2) \in \tau(m, l)$. Consider any l-segment g of f. We shall prove that $|g|_{\neq} \leq m$. As every $|f_j^{m_j}| \geq l$, the string g is a factor of $g_1 f_1^{m_1}$, or $f_{n-1}^{m_{n-1}} g_n$, or $f_{j-1}^{m_{j-1}} g_j f_j^{m_j}$ for some j. In any case, the number of errors in g is less than or equal to the number of errors in some factor g_i of f. As g_i is also a factor of $h \in H_{\tau,m,k}$ and $|g_i| < k$ (by Claim 3), Remark 1 implies $|g_i|_{\neq} \leq m$, as required. ■

3.2 Inclusions between SID channels

In accordance to one's intuition, the SID channels considered in this paper are pairwise distinct. We confirm this basic result next.

Theorem 2 *For every SID channels $\tau(m, l)$ and $\tau(m', l')$, with $m, m' > 0$, we have that $\tau(m, l) = \tau(m', l')$ if and only if $m' = m$ and $l' = l$.*

For the proof of this theorem we need the next two lemmata – we note that, in fact, the theorem can be proven without Lemma 3, however, that lemma is needed in the next section. The expression $l\%l'$ denotes the remainder of the integer division l/l'.

Lemma 2 *[8] For any SID channels $\tau(m, l)$ and $\tau(m', l')$, if $m' = m$ and $l < l'$ then $\tau(m', l') \subsetneq \tau(m, l)$.*

Lemma 3 *For every two SID channels $\tau(m, l)$ and $\tau(m', l')$, with $m, m' > 0$, we have that $\tau(m, l) \supseteq \tau(m', l')$ if and only if*

$$\begin{aligned} m &\geq m' \lfloor \frac{l}{l'} \rfloor + \min\{l\%l', m'\}, \ \textit{if } \iota \textit{ is not in } \tau \\ m &\geq m' \lfloor \frac{l}{l'} \rfloor + m', \ \textit{if } \iota \textit{ is in } \tau \end{aligned}$$

Proof. Let $q = \lfloor \frac{l}{l'} \rfloor$, $r = l\%l'$ and $r_0 = \min\{r, m'\}$. For the 'only if' part, consider the following words

$$\begin{array}{lll} u_1 = 1^{r_0}(0^{l'-m'}1^{m'})^q & u_2 = 0^{(l'-m')q} & \\ u_3 = 0^{l'q} & u_4 = 1^{m'}(0^{l'}1^{m'})^q & u_5 = 0^{r_0+l'q} \end{array}$$

It is easy to verify that

$$\begin{array}{ll} (u_1, u_2) \in \tau(m', l'), & \text{if } \delta \text{ occurs in } \tau, \\ (u_3, u_4) \in \tau(m', l'), & \text{if } \iota \text{ occurs in } \tau, \\ (u_1, u_5) \in \tau(m', l'), & \text{if } \sigma \text{ occurs in } \tau. \end{array}$$

By the assumption, at least one of the above pairs, say (v, v') must be in $\tau(m, l)$. In any case, as the first word of (v, v') is of length at most l, the second word v' can be obtained from v using at most m errors. On the other hand, any edit string that turns v to v' must have at least $r_0 + m'q$ errors if $v = u_1$ and τ contains no ι, or at least $m' + m'q$ errors if $v = u_3$ and τ contains ι. For example, as u_5 contains only 0's, to turn u_1 to u_5, all the $r_0 + m'q$ 1's in u_1 must be deleted or substituted with 0's.

For the 'if' part, consider any pair $(u, v) \in \tau(m', l')$. Then, there is an edit string $h \in H_{\tau,m',l'}$ such that $u = h\pi_1$ and $v = h\pi_2$. We shall show that $h \in H_{\tau,m,l}$, hence also $(u, v) \in \tau(m, l)$. So let g be any l-segment of h. It is sufficient to show that $|g|_{\neq} \leq m$. We can write $g = g_0 g_1 \cdots g_q$ such that the input size of each g_j is l', for $j = 1, \ldots, q$, and the input size of g_0 is $l\%l'$. Moreover, we agree that there are no insertion errors at the ends of the g_j's – this is obvious for $j = q$, as g is an l-segment, and possible for each $j < q$, as otherwise any insertions at the end of g_j could be "moved" to the beginning of g_{j+1}.

Next we consider two cases. In the first case, $m \geq r_0 + m'q$ and ι is not in τ. Then $|g|_{\neq} \leq \min\{m', |g_0|\} + qm'$. Moreover, as ι is not in τ, the input size of g_0 is equal to $|g_0|$, that is, $|g_0| = l\%l'$. Hence, $|g|_{\neq} \leq r_0 + qm' \leq m$, as required. Now consider the case where $m \geq m'(1+q)$ and ι is in τ. Obviously there can be at most m' errors in each g_i, hence, $|g|_{\neq} \leq (1+q)m' \leq m$, as required. ■

Proof of Theorem 2. The 'if' part is trivial. For the 'only if' part, assume without loss of generality that $m \leq m'$. By Lemma 2, it is sufficient to show that $m = m'$. We use contradiction by assuming $m < m'$. Let $q = \lfloor \frac{l}{l'} \rfloor$. If ι is in τ then Lemma 3 implies that $m \geq m'q + m'$, hence, $m' > m'q + m'$, a contradiction. Now assume that ι is not in τ. By Lemma 3, $m \geq m'q + \min\{l\%l', m'\}$. As $m < m'$, we must have $q = 0$, $l\%l' = l$ and $l < l'$. This implies $m \geq 0 + \min\{l, m'\}$, that is, $m \geq m'$ or $m \geq l$, which is a contradiction. Hence, $m = m'$. ■

3.3 Inverses of SID channels

The symmetry in the definition of error-detection implies the following.

Remark 2 *A language L is error-detecting for some channel γ if and only if it is so for γ^{-1}.*

This observation motivates the study of the inverses of combinatorial channels. In particular, in the next theorem, we focus on SID channels of the form $\tau(m, l)$. We show that $\sigma(m, l)$ is a symmetric channel, and the inverse of $\iota(m, l)$ is $\delta(m, l + m)$. On the other hand, counter to one's intuition, the channels $(\iota \odot \delta)(m, l)$ and $(\sigma \odot \iota \odot \delta)(m, l)$ are not symmetric. For example, for any parameters m, l with $l > m > 0$, we have that

$$(1^m 0^{l-m} 1, 0^{l-m}) \in (\iota \odot \delta)(m, l),$$

via the edit string $h = (1/\lambda)^m (0/0)^{l-m} (1/\lambda) \in H_{\iota\odot\delta,m,l}$. On the other hand, there is no edit string $h' \in H_{\iota\odot\delta,m,l}$ such that $h'\pi_1 = 0^{l-m}$ and $h'\pi_2 = 1^m 0^{l-m} 1$, as this would require at least $m + 1$ insertions of 1's in $l - m$ input symbols.

Theorem 3 *Let m, l be positive integers with $m < l$. The following statements hold true.*

1. *The inverse of the channel $\iota(m, l)$ is $\delta(m, l + m)$.*
2. *The inverse of the channel $\delta(m, l)$ is $\iota(m, l - m)$, when $l > 2m$.*
3. *The inverse of the channel $\sigma(m, l)$ is $\sigma(m, l)$, that is, this channel is symmetric.*

4. Let τ be an error type other than σ, ι, δ. We have that

$$\tau(m,l)^{-1} \neq \tau'(m',l'),$$

for every error type τ' and $m', l' > 0$.

Proof. For an edit operation (x/y), the notation $(x/y)^{-1}$ represents the edit operation (y/x). This notation is extended homomorphically to edit strings h, and then to edit systems H, that is, $H^{-1} = \{h^{-1} : h \in H\}$. It is easy to verify that, $|h|_{\neq} = |h^{-1}|_{\neq}$ for every edit string h, and that for every edit system H, we have

$$\mathrm{ch}(H)^{-1} = \mathrm{ch}(H^{-1}).$$

For the first statement, we first consider any $h \in H_{\delta,m,l+m}$ and we show that $h^{-1} \in H_{\iota,m,l}$. Obviously, h^{-1} can contain only non-errors and insertion errors. For any l-segment g of h^{-1} we need to show that $|g|_{\neq} \leq m$. Equivalently, we need to show $|g^{-1}|_{\neq} \leq m$. The segment g contains exactly l non-errors and is of the form

$$g = (x_1/a_1) \cdots (x_t/a_t),$$

where each a_i is in Σ, each x_i is in $\Sigma \cup \{\lambda\}$, and (x_t/a_t) is a non-error. Note that g^{-1} has exactly l non-errors and belongs to $H_{\delta,m,l+m}$. We shall derive a contradiction by assuming $|g^{-1}|_{\neq} \geq m+1$. Indeed, consider the shortest prefix g_1, say, of g^{-1} containing exactly $m+1$ errors. Then, (a_t/x_t) is not part of g_1, so g_1 contains up to $l-1$ non-errors. As $g_1 \in H_{\delta,m,l+m}$, we have that the input size of g_1 is $\geq l+m+1$ and, therefore, g_1 contains at least l non-errors; a contradiction.

For the converse of the first statement, we consider any $h \in H_{\iota,m,l}$ and we show that $h^{-1} \in H_{\delta,m,l+m}$. Let g be any $(l+m)$-segment of h^{-1}. We shall show that $|g|_{\neq} \leq m$. As the edit string g^{-1} is a factor of h, possibly padded with $(1/1)$'s at the end, we have that g^{-1} is in $H_{\iota,m,l}$ and of the form

$$g^{-1} = (x_1/a_1) \cdots (x_{l+m}/a_{l+m}),$$

where each a_i is in Σ and each x_i is in $\Sigma \cup \{\lambda\}$. Let f be equal to g^{-1} if g^{-1} ends with a non-error, or equal to $g^{-1}(1/1)$, if g^{-1} ends with an insertion error. In any case we have that $|f| \leq l+m+1$. It is sufficient to show $|f|_{\neq} \leq m$. We shall derive a contradiction by assuming $|f|_{\neq} \geq m+1$. Indeed, as $f \in H_{\iota,m,l}$, the input size of f must be at least $l+1$, which means that f contains at least $l+1$ non-errors. Hence, $|f| \geq (l+1)+(m+1) > l+m+1$; a contradiction.

The second statement follows easily from the first one.

The third statement follows when we show that $H_{\sigma,m,l}^{-1} = H_{\sigma,m,l}$. This can be done without complications by following the definition of $H_{\sigma,m,l}$.

For the fourth statement, we shall consider only two cases and we leave the rest to the reader.

Case of $\tau = \sigma \odot \iota$: Assume for the sake of contradiction that $\tau(m,l)^{-1} = \tau'(m',l')$ for some τ', m', l'. Obviously, it must be that $\tau' = \sigma \odot \delta$. If $m' > m$ then, as $(a^{m'}, \lambda) \in \tau'(m',l')$, we have that $(\lambda, a^{m'}) \in \tau(m,l)$, which is impossible. If $m' < m$ then, as $(\lambda, a^m) \in \tau(m,l)$, we have that $(a^m, \lambda) \in \tau'(m',l')$, which is again impossible. Hence, $m' = m$. If $l' > l$ then, as

$$(0/1)^m (0/0)^{l-m} (\lambda/1) \in H_{\tau,m,l},$$

we have that $(0^m 0^{l-m}, 1^m 0^{l-m} 1) \in \tau(m,l)$. Hence, $(1^m 0^{l-m} 1, 0^m 0^{l-m}) \in \tau'(m,l')$, which is impossible, as we would need $m+1$ errors to eliminate the 1s in the factor $1^m 0^{l-m} 1$ of length $l+1 \leq l'$. Now if $l' \leq l$ one considers

$$h = (1/\lambda)^m (0/0)^{l'-m} (1/\lambda) \in H_{\tau',m,l'},$$

and verifies that $(h\pi_1, h\pi_2) \in \tau'(m,l')$, but $(h\pi_2, h\pi_1) \notin \tau(m,l)$.

Case of $\tau = \iota \odot \delta$: As above, assume for the sake of contradiction that $\tau(m,l)^{-1} = \tau'(m',l')$ for some τ', m', l'. Obviously, it must be that $\tau' = \tau$. Again one shows that $m' = m$. If $l' > l$ then one considers the edit string

$$h = (1/\lambda)^m (0/0)^{l-m} (1/\lambda) \in H_{\tau,m,l},$$

and verifies that $(h\pi_1, h\pi_2) \in \tau(m,l)$, but $(h\pi_2, h\pi_1) \notin \tau(m,l')$. If $l' \le l$ then one uses the edit string used in the above case for $l' \le l$. ■

We note that the second statement of Theorem 3 remains valid for $0 < l \le m$, provided we accept the definition of the relation $\iota(m',l')$ even for any l' with $0 < l' \le m'$.

4 Algorithmic Consequences

This section consists of two parts. The first one concerns the problem of computing the maximal error-detecting capabilities of a given regular language with respect to SID error models. The second part consists of Section 4.5 and concerns the problem of computing the (inner) distance of a given regular language.

Now we assume that a regular language $L(A)$ is given via a finite automaton A accepting at least two different words, including the empty word λ – otherwise one can simply add λ to $L(A)$. We shall utilize the following results from [6, 8].

Theorem 4 *[6] The following problem is decidable in polynomial time.*
Input: a finite automaton A and a finite transducer realizing some channel γ.
Output: Y/N, depending on whether the language $L(A)$ is error-detecting for γ.

Proposition 1 *[6, 5] For each error type τ and integers m and l, with $0 \le m < l$, there is (effectively) a finite transducer realizing the SID channel $\tau(m,l)$.*

We note that the construction of a transducer realizing $\tau(m,l)$ would normally require a very large number of states and transitions – exponential with respect to m [5]. Thus, the algorithms before Section 4.5 are practical only for small values of m and l.

Lemma 4 *[8] Let A be a finite automaton accepting at least two words, and let $\tau(m,l)$ be an SID channel such that the language $L(A)$ is error-detecting for $\tau(m,l)$. If $\tau \ne \sigma$ then m is less than the length of a shortest nonempty word in $L(A)$.*

4.1 The error model $\mathbb{C}^1_\tau[l] = \{\tau(m,l) :$ for any m with $0 \le m < l\}$

Here one fixes a word length l that appears to be appropriate for capturing the statistics of errors within words of length l. This error model is finite and consists of the channels $\tau(m,l)$, for $m = 0, \ldots, l-1$. Using Proposition 1 and Theorem 4, one computes the largest value of m such that the given language $L(A)$ is error-detecting for $\tau(m,l)$. In this case, the required maximal error-detecting capability of $L(A)$ is $\tau(m,l)$.

4.2 The error model $\mathbb{C}^2_\tau[m] = \{\tau(m,l) :$ for any l with $l > m\}$

Here, for a fixed maximum number of errors m, we wish to compute the shortest word length l', if any, such that the given language $L(A)$ is error-detecting for $\tau(m,l')$. This case was solved in [8] when $\tau = \sigma$. Now we can solve it for any error type $\tau \ne \sigma$ using Theorem 1. Indeed, as

$\tau(m,l) \subsetneq \tau(m,l')$ when $l > l'$ we need to compute the smallest $l' \in \{m+1,\ldots,1+s_A^2(m+1)\}$ (if any) such that $L(A)$ is error-detecting for $\tau(m,l')$. This is possible using Proposition 1 and Theorem 4. Note that the channel $\tau(m,l')$ would be different from the one we get, say $\tau(m',l)$, in the case of the error model $\mathbb{C}_\tau^1[l]$ if the fixed m is greater than m'.

4.3 The error model $\mathbb{C}_\tau = \{\tau(m,l) :$ for any m and l with $0 \le m < l\}$

Again this case was solved in [8] when $\tau = \sigma$. Now assume $\tau \neq \sigma$. We have the following algorithm.

```
Let $M$ be the length of a shortest word in $L(A)$.
Let $s_A$ be the number of states in $A$.
For each $m = 1,\ldots,M-1$
    For each $l = m+1,\ldots,1+s_A^2(m+1)$
        Construct the transducer for $\tau(m,l)$
        Decide whether $L(A)$ is error-detecting for $\tau(m,l)$
        If yes, add $\tau(m,l)$ to a set $C_\tau$.
If $C_\tau = \emptyset$, let $C'_\tau = \{\tau(0,1)\}$.
Else, use Lemmata 2,3 to compute the set $C'_\tau$ of all maximal channels in $C_\tau$.
Output $C'_\tau$.
```

Of course there are several ways to improve the performance of the above algorithm. For example, one can modify the linear search for the value of l to a binary search. Moreover, if for some value of m we have that there is *no* l such that $L(A)$ is error-detecting for $\tau(m,l)$, then there is no need to run the outer loop for any larger values of m.

4.4 The error model SID

Recall the error model SID was defined in the introduction. In this case, one can use the result of [8] for $\tau = \sigma$ and the algorithm in Section 4.3 to compute the set C'_τ of all $\mathbb{C}_\tau$-maximal error detecting capabilities of $L(A)$, for each error type $\tau \neq \sigma$. However, in view of Remark 2 and Theorem 3, we can omit running the algorithm for $\tau = \iota$, and determine the $\mathbb{C}_\iota$-maximal error-detecting capabilities of the given language via the inverses of the channels in C'_δ. We note that, in this error model, one has to resolve all containment relationships between SID channels involving different error types. For example, it can be shown that $\sigma(m,l) \subsetneq (\iota \odot \delta)(2m,l)$. We leave the details of this exercise to the interested reader.

4.5 Computing distances of regular languages

The concept of (string) distance has been a useful tool in various information processing domains, starting with the Hamming distance [4, 11] and Levenshtein (or edit) distance [10] in error control coding, and then with various versions of the edit distance in applications like speech processing and bioinformatics [14]. Moreover, some interesting distances have been considered also in the context of formal languages [2, 9, 13]. Here we discuss a natural connection between the concepts of (inner) distance and maximal error-detecting capability for formal languages.

A *(string) distance* is defined to be a function

$$d : \Sigma^* \times \Sigma^* \to [0,\infty],$$

such that $d(w,z) = 0$ if and only if the words w, z are equal. We allow $d(w,z) = \infty$ to represent the fact that the word w cannot be compared to z via the distance d. This is possible for instance when $d = \text{Hamm}$, the Hamming distance, which is meant to relate words of the same length. We note that some authors use the term 'distance' for what others define to be a metric, that is, a distance in our sense that also is symmetric and satisfies the triangle inequality. Now, for any number $t \in [0, \infty)$, we define the channel

$$\gamma_d(t) = \{(w, z) : d(w, z) \leq t\}.$$

The next lemma follows easily from the above definitions.

Lemma 5 *Let d be a distance and L be a language. The following statements hold true.*

1. *L is error-detecting for $\gamma_d(0)$.*
2. *For all $t, t' \in [0, \infty)$ with $t \leq t'$, we have that $\gamma_d(t) \subseteq \gamma_d(t')$. Hence, if L is not error-detecting for some $\gamma_d(t)$ then it is not error-detecting for $\gamma_d(t')$ for all $t' \geq t$.*

In the sequel, we focus on *integral distances*. A distance d is called integral if $d(w, z)$ is an integer for all words w, z. For any language L containing at least two words, we define the *(inner) distance* of L as

$$d(L) \triangleq \min\{d(w, z) : w, z \in L_\lambda,\ w \neq z\}.$$

Theorem 5 *Let d be an integral distance and let L be a language containing at least two words.*

1. *For every integer $m \geq 0$, L is error-detecting for $\gamma_d(m)$ if and only if $d(L) > m$.*
2. *The value $d(L)$ is equal to $m+1$, where m is the largest value such that $\gamma_d(m)$ is the (unique) maximal error-detecting capability of L with respect to the error model $\mathbb{C}_d = \{\gamma_d(0), \gamma_d(1), \ldots\}$.*

Proof. The first statement follows easily from the definitions, so we deal with the second statement. By definition, $d(L) = d(u_1, v_1)$ for some words $u_1, v_1 \in L_\lambda$ with $u_1 \neq v_1$. Let $m = d(L) - 1$. As $(u_1, v_1) \in \gamma_d(m+1)$, L is not error-detecting for $\gamma_d(m+1)$. Now we show that L is error-detecting for $\gamma_d(m)$. So let $u_0, v_0 \in L_\lambda$ such that $(u_0, v_0) \in \gamma_d(m)$. As $d(u_0, v_0) < 1 + m = d(L)$, it must be that $d(u_0, v_0) = 0$ and, therefore, $u_0 = v_0$. Hence, L is error-detecting for $\gamma_d(m)$ and, as it is not error-detecting for any $\gamma_d(m')$ with $m' > m$, we have that $\gamma_d(m)$ is the unique maximal error-detecting capability of L with respect to $\mathbb{C}_d$. ■

The first statement of the above theorem is a natural extension of the classical fact of error control coding [4, 11] that a fixed-length code (language) C is m-error-detecting (with respect to substitution errors) if and only if $\text{Hamm}(C) > m$. The second statement can be used to address the following problem.

- **Problem 1** Let d be a fixed integral distance. Given a finite automaton A accepting at least two words, compute the integer $d(L(A))$.

We discuss here a method to address the above problem and mention a couple of its applications to specific distances. As the details of this method constitute a project on its own, here we restrict ourselves to the main points of the method. We make two assumptions regarding the distance d: (i) for each finite automaton A, there is (effectively) an upper bound $\beta_d(A)$ for the quantity $d(L(A))$; and (ii) for each nonnegative integer n, there is (effectively) a transducer $T_{d,n}$ realizing the channel $\gamma_d(n)$. With these assumptions, the method computes the largest value $m \in \{0, \ldots, \beta_d(A) - 1\}$ satisfying the condition "$L(A)$ is error-detecting for the channel $\gamma_d(m)$."

This condition can be tested via Theorem 4 using the transducer $T_{d,m}$ for realizing the channel $\gamma_d(m)$. Then, by Theorem 5, the required output is $d(L(A)) = 1 + m$.

The first application of the above method is for the case of $d = \mathrm{ed}$, the edit distance, that is,

$$\mathrm{ed}(u, v) \triangleq \min\{|g|_{\neq} : g \in E^*,\, g\pi_1 = u,\, g\pi_2 = v\}.$$

We note that [7] establishes an upper bound for $\mathrm{ed}(L(A))$ and solves Problem 1 in polynomial time using different tools. The same problem can now be solved using the above method, again in polynomial time. As stated already, the details of the new algorithm and its performance compared to the one in [7] are questions outside the aims of the present paper.

The second application of the above method is for the case of $d = d_l$, for some $l > 0$, a distance introduced in [9] and translated in our context as follows:

$$d_l(u, v) \triangleq \min\{\nu_l(h) : h \in E^*,\, h\pi_1 = u,\, h\pi_2 = v\},$$

where $\nu_l(h) = \max\{|g|_{\neq} : g \text{ is an } l\text{-segment of } h\}$. This distance definition is based on the largest number of errors in the l-segments of an edit string h that transforms u to v. Our method applies again here because (i) $d_l(u, v) \leq l$ for all words u and v and (ii) the required transducer $T_{d_l,m}$ can be obtained using Proposition 1 when we take into account the following lemma whose proof is based on the definitions of the channels involved.

Lemma 6 *For any integer m with $0 \leq m < l$, we have that $\gamma_{d_l}(m) = (\sigma \odot \iota \odot \delta)(m, l)$.*

Corollary 1 *The following problem is computable.*
Input: finite automaton A accepting at least two words and integer $l > 0$.
Output: the value $d_l(L(A))$.

5 Discussion

We have considered the problem of computing maximal error-detecting capabilities of regular languages with respect to combinatorial channels. In particular, we have resolved the problem for the class of SID channels $\tau(m, l)$, for any error type τ, a problem that was left open in [8]. Some related questions for further research can be found in [8]. For example:

- Implement the proposed algorithms and possibly apply them to real world languages, computing thus the inherent error-detecting capabilities of such languages. The difficulty here is that the efficiency of these algorithms is affected by the high descriptional complexity of SID channels. A first attempt towards this end is in progress in [3].
- In view of the above comment, investigate the maximal error-detecting capability problem for homophonic channels [8], as they have much lower descriptional complexities.
- Investigate the analogous concept of maximal error-correcting capability for formal languages.

Having noted the natural connection between string distances and combinatorial channels, we also propose the following topics for further research:

- Investigate further the method of Section 4.5 for computing the distance of a regular language, in particular versions of the edit distance.
- Investigate the channels corresponding to the additive distances considered in [2, 13].

References

[1] J. Berstel and D. Perrin. *Theory of Codes.* Academic Press, Inc., Orlando Florida, 1985.

[2] C. Calude, K. Salomaa, and S. Yu. Additive distances and quasi-distances between words, *Journal of Universal Computer Science* **8** (2002), 141–152.

[3] A. Daka. *Maximal Error-detecting Capabilities of Regular Languages* (tentative title). MSc APS thesis, Dept. Math. and Computing Sci., Saint Mary's University, Canada (in progress).

[4] J. Duske and H. Jürgensen. *Codierungstheorie.* BI Wissenschaftsverlag, Manheim, 1977.

[5] L. Kari and S. Konstantinidis. Descriptional complexity of error/edit systems, *Journal of Automata, Languages and Combinatorics* **9** (2004) 2/3, 293-309.

[6] S. Konstantinidis. Transducers and the properties of error-detection, error-correction and finite-delay decodability, *Journal of Universal Computer Science* **8** (2002), 278–291.

[7] S. Konstantinidis. Computing the edit distance of a regular language, *Information and Computation* **205** (2007), 1307-1316. A shorter version appears as "Computing the Levenshtein distance of a regular language" in: Proceedings of *IEEE Information Theory Workshop on Coding and Complexity*, Rotorua, New Zealand, Aug. 29 - Sep. 1, 2005, pp 113116.

[8] S. Konstantinidis and P.V. Silva. Maximal error-detecting capabilities of formal languages, *Journal of Automata, Languages and Combinatorics* **13** (2008), 55–71.

[9] C. L. McAloney. Error-correction and finite transductions. MSc thesis, Dept. Computing and Information Sci., Queen's University, Canada, 2003.

[10] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Dokl.* **10** (1966), 707–710.

[11] W. W. Peterson and E. J. Weldon, Jr. *Error-Correcting Codes.* MIT Press, Cambridge, second ed., 1972.

[12] G. Rozenberg and A. Salomaa (eds). *Handbook of Formal Languages, Vol. I.* Springer-Verlag, Berlin, 1997.

[13] K. Salomaa and P. Schofield. State complexity of additive weighted finite automata. *International Journal of the Foundations of Computer Sci.* **18** (2007), 1407-1416.

[14] D. Sankoff and J. Kruskal. *Time Warps, String Edits, and Macromolecules: The theory and practice of sequence comparison.* CSLI Publications, 1999.

[15] S. Yu. Regular Languages. In [12], 41–110.