# Maximal Error-Detecting Capabilities of Formal Languages[1]

Stavros Konstantinidis[†] and Pedro 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 is a set of pairs of words describing all the possible input-output channel situations. We introduce the concept "maximal error-detecting capability" of a formal language, with respect to a certain class of combinatorial channels. The new concept is intended to address formally the question of "finding the largest amount of errors that a language can detect". We focus on rational channels (those described by finite transducers) and on regular languages, and consider the problem of computing maximal error-detecting capabilities of a given regular language for various classes of rational channels. We also discuss briefly the concept "maximal error-correcting capability" of a formal language.

**Key words**: algorithm, automaton, channel, combinatorial channel, error correction, error detection, error model, maximal, formal language, regular language.

## 1 Introduction

A (combinatorial) channel is a set of pairs of words describing all the possible input-output situations permitted by the channel. The fact that the pair $(w, z)$ is in the channel means that the word $z$ can be received via the channel when the word $w$ is used as input. In this case, if $w \neq z$ then we say that $z$ contains errors, or that $w$ was received with errors. A language $L$ is *error-detecting for* some channel $\gamma$, [5], if there is no pair $(w, z)$ in the channel such that $w \neq z$ and both $w$ and $z$ are in $L_\lambda$; that is, the channel cannot transform a word of $L_\lambda$ into another different word of $L_\lambda$. This fact allows one to detect that a received word $z$ contains errors exactly when $z$ is not in $L_\lambda$. Here $\lambda$ is the empty word and $L_\lambda = L \cup \{\lambda\}$.

In this paper we introduce the concept "maximal error-detecting capability of a formal language" in order to address the informal question of "finding the largest amount (or set) of errors that a given language can detect". This concept is defined with respect to a certain *error model*, which is simply a set of channels. We focus on the problem of computing maximal error-detecting capabilities of regular languages with respect to various error models of rational channels (those realized by finite transducers). We believe that this problem is important. First, it is interesting from a theoretical

[1]Research supported by (i) a Discovery Research Grant of NSERC, Canada, and (ii) the Centro de Matemtica da Universidade do Porto (CMUP), financed by FCT (Portugal) through the programmes POCTI and POSI, with national and European Community structural funds. This work was presented in part (a) in the 10th WSEAS Conference on Computers, July 2006, Vouliagmeni, Greece; and (b) in the SCRA 2006—FIM XIII on Interdisciplinary Mathematical and Statistical Techniques, September 2006, Tomar, Portugal.

point of view. Second, it can be viewed as the converse of the following classic problem of coding theory: find the largest language (or code) that is error-detecting for a given channel. Third, any algorithms for computing maximal error-detecting capabilities of languages can potentially be applied on real languages and codes, such as gene languages and codes used in protocols of data communication. In the former case, we would know the inherent error-detecting capabilities of a gene language. In the latter case, we would be able to find out whether an actual code possesses more error-detecting capabilities than those for which it was designed.

To our knowledge, the concept of maximal error-detecting capability has not been studied in the past. In this study we intend to obtain some basic results in connection with certain important error models. Our study is not meant to be exhaustive and we choose to leave several questions that arise here for future research.

The paper is organized as follows. The next section contains the basic terminology and background, introduces formally the concepts of maximal error-detecting and -correcting capabilities of a formal language, and provides a couple of examples on these concepts. *Section 3* gives a few basic results for the error model of all rational channels. *Section 4* focuses on error models consisting of certain rational channels that are called SID channels (substitution-insertion-deletion channels). An SID channel is specified by the type of errors (symbol changes) permitted in the input words as well as by two positive integers $m, l$ indicating that no more than $m$ errors can be applied in any segment of length $l$ of an input word. That section contains basic results on SID channels that concern bounds on $m, l$ when a language is error-detecting for such a channel, as well as subset relationships between SID channels. These results are used in *Section 5* as tools to address the problem of computing maximal error-detecting capabilities of a given regular language with respect to a given error model. In that section we also discuss error models of certain homophonic channels. Finally, *Section 6* contains a few concluding remarks and states a few problems for future research.

## 2 Basic definitions and background

We use the symbol $\Sigma$ to denote a fixed, but arbitrary *alphabet*. Then the symbol $\Sigma^*$ denotes the set of all *words* (or strings) over $\Sigma$, including the empty word $\lambda$. The *length* of a word $w$ is denoted by $|w|$. A *language* is any set of words. If $K$ is a language then $K_\lambda$ denotes the language $K \cup \{\lambda\}$. A binary relation over $\Sigma$ is a subset of $\Sigma^* \times \Sigma^*$. A *(combinatorial) channel* is a binary relation $\gamma$ over the alphabet $\Sigma$, that is, $\gamma \subseteq \Sigma^* \times \Sigma^*$, such that $\gamma$ is *domain preserving*, that is, $\gamma$ contains the pair $(v, v)$ when the word $v$ is a possible input to the channel. This requirement ensures that error-free communication is always possible via the channel. The *domain* $\operatorname{dom}\gamma$ of the channel $\gamma$ is the set of all possible inputs to $\gamma$, that is,

$$\operatorname{dom}\gamma = \{w \mid (w, z) \in \gamma, \text{ for some word } z\}.$$

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. Besides the error model, we consider a language that is intended to be used for representing information. Depending on the application, it is required that the language satisfies certain properties, for instance, error-detection or error-correction with respect to a channel $\gamma$ in $\mathbb{C}$. The definition of error-detection is given in the Introduction. A language $L$ is *error-correcting for* a channel $\gamma$, [5], when no two different words of $L_\lambda$ can result via $\gamma$ into the same output; that is, if $(w_1, z)$ and $(w_2, z)$ are in $\gamma$ and $w_1, w_2$ are in $L_\lambda$ then $w_1$ must be equal to $w_2$. This fact allows one to correct a given channel output $z$ to a unique word of $L$.

In the sequel we shall focus on the error-detection property and we shall consider mainly *regular languages*, that is, languages accepted by finite automata, and *rational channels*, that is, channels realized by finite transducers. 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 $a$ 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 $L(A)$. A *finite transducer* also has an initial state and some final states, and a finite set of transitions of the form $(p, x/y, q)$. Such a transition says that if the transducer is in state $p$ and the current input starts with $x$ then it enters in state $q$ and outputs the word $y$. The transducer consumes a given input word and produces some output word by following its available transitions – see [11], for instance, for more details on finite automata and transducers.

We are interested in the following problem.

**Problem 1** *Let $A$ be a finite automaton accepting a language with at least two words, and let $\mathbb{C}$ be an error model. Compute a channel $\gamma$ in $\mathbb{C}$ such that $\gamma$ is a $\mathbb{C}$-maximal error-detecting capability of $L$ (or $\mathbb{C}$-maximal error-correcting capability of $L$).*

A channel $\gamma$ is said to be a $\mathbb{C}$-*maximal error-detecting capability of* $L$ if (i) $\gamma$ is in $\mathbb{C}$, (ii) $L$ is error-detecting for $\gamma$, and (iii) $L$ is not error-detecting for $\gamma'$, for any channel $\gamma'$ that properly contains $\gamma$, that is, $\gamma' \supsetneq \gamma$. The concept of maximal error-correcting capability is defined analogously.

An *error type* is an expression in

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

and is used to define certain simple channels as follows: For any $m \geq 0$ and error type $\tau$ the channel $\tau(m, \infty)$ consists of all pairs of words $(w, z)$ such that $z$ results by performing no more than $m$ errors of type $\tau$ in the word $w$. The three simple error types $\sigma, \iota, \delta$ denote *substitutions, insertions,* and *deletions*, respectively. The symbol $\odot$ is used simply as a connective for the simpler types

**Example 1** A typical channel considered in coding theory (in many cases implicitly) is the channel $\sigma(m, \infty)$ consisting of all pairs of words $(w, z)$ such that $z$ can be received using at most $m$ substitutions of symbols in $w$, that is, $H(w, z) \leq m$, where $H(w, z)$ is the ***Hamming distance*** between the words $w$ and $z$. For example, $(0000, 0110) \in \sigma(2, \infty)$, where we assume that 0 and 1 are elements of the alphabet. In this case, we consider the error model

$$\mathbb{C}_\sigma[\infty] = \{\sigma(m, \infty) \mid m \geq 0\}$$

and the error-detection version of Problem 1 is equivalent to computing, for a given language $L$, the maximum value of $m$ such that $H(L) > m$, where the quantity $H(L)$ is the smallest Hamming distance between any two different words in $L$ – indeed, note that $L$ is error-detecting for $\sigma(m, \infty)$ if and only if $H(L) > m$ [8]. This instance of the problem can be solved efficiently [4]. Using this result one can also solve the error-correction version of the problem by noting that $L$ is error-correcting for $\sigma(m, \infty)$ if and only if $H(L) > 2m$.

**Example 2** Another typical channel is the channel $(\sigma \odot \iota \odot \delta)(m, \infty)$ consisting of all pairs of words $(w, z)$ such that $z$ can be received using a total of at most $m$ substitutions, insertions, and

deletions of symbols in $w$, that is, $\Lambda(w,z) \leq m$, where $\Lambda(w,z)$ is the Levenshtein distance (also called edit distance) between the words $w$ and $z$. In this case, we consider the error model

$$\mathbb{C}_{(\sigma\odot\iota\odot\delta)}[\infty] = \{(\sigma \odot \iota \odot \delta)(m,\infty) \mid m \geq 0\}$$

and the error-detection version of Problem 1 is equivalent to computing, for a given language $L$, the maximum value of $m$ such that $\Lambda(L) > m$, where the quantity $\Lambda(L)$ is the smallest Levenshtein distance between any two different words in $L$ – indeed, note that $L$ is error-detecting for $(\sigma \odot \iota \odot \delta)(m,\infty)$ if and only if $\Lambda(L) > m$. This instance of the problem can be solved in polynomial time [6]. Using this result one can also solve the error-correction version of the problem by noting that $L$ is error-correcting for $(\sigma \odot \iota \odot \delta)(m,\infty)$ if and only if $\Lambda(L) > 2m$ [7].

We close this section by noting an interesting relationship between the concepts of error-detection and error-correction. Recall, that the composition $\gamma_2 \circ \gamma_1$ of two relations is the relation that consists of all pairs $(w_1, w_2)$ such that $(w_1, z)$ is in $\gamma_1$ and $(z, w_2)$ is in $\gamma_2$, for some word $z$. Moreover, the inverse of a relation $\gamma$ is $\gamma^{-1} = \{(z,w) : (w,z) \in \gamma\}$.

**Remark 1**. A language $L$ is error-correcting for $\gamma$ if and only if it is error-detecting for $\gamma^{-1} \circ \gamma$.

## 3 The Error Model of Rational Channels

In this section we consider the error model $\mathbb{C}[\text{rat}]$ of all *rational channels*, that is, all channels $\gamma$ that can be realized by finite transducers. Our focus on rational channels should not be considered as a restriction because, to our knowledge, most channels can be described by finite-state machines. We obtain a few basic results that confirm our intuition about the legitimacy of the concepts of maximal error-detection and -correction.

**Lemma 1** *Let $\gamma_1, \gamma_2$ be channels and $L$ be a language.*

1. *If $\gamma_1 \subseteq \gamma_2$ and $L$ is error-detecting (respectively, error-correcting) for $\gamma_2$ then $L$ is error-detecting (respectively error-correcting) for $\gamma_1$.*
2. *If $L$ is error-detecting for $\gamma_1$ and for $\gamma_2$ then $L$ is error-detecting for $\gamma_1 \cup \gamma_2$.*

*Proof.* The proof is based directly on the definition of error-detecting language and is left to the reader. ■

For the next statements we recall that a language $L$ is *maximal error-detecting* for a channel $\gamma$ if there is no word $w$ outside of $L_\lambda$ such that $L \cup \{w\}$ is error-detecting for $\gamma$. The concept of *maximal error-correcting* language for a channel $\gamma$ is defined analogously.

**Theorem 1** *Let $L$ be a nonempty language and let $\gamma$ be a channel such that all words of $L$ are possible inputs to $\gamma$, that is, $L \subseteq \operatorname{dom}\gamma$. If $\gamma$ is a $\mathbb{C}$[rat]-maximal error-detecting capability of $L$ then $L$ is maximal error-detecting for $\gamma$.*

*Proof.* Assume for the sake of contradiction that $L$ is not maximal error-detecting for $\gamma$. Then, there is a word $w$ not in $L_\lambda$ such that the language $L' = L \cup \{w\}$ is error-detecting for $\gamma$. We choose any word $v_0$ from $L$ and define the channel

$$\gamma' = \gamma \cup \{(v_0, w)\}.$$

Note that $v_0 \in \operatorname{dom}\gamma$ implies that indeed $\gamma'$ is domain preserving. Moreover, the channel $\gamma'$ is in $\mathbb{C}$[rat], as the class of rational relations is closed under union. Obviously $L'$ is not error-detecting for $\gamma'$. As $L'$ is error-detecting for $\gamma$, we have that $\gamma$ is a proper subset of $\gamma'$ and, as $\gamma$ is a maximal error-detecting capability of $L$, it follows that $L$ is not error-detecting for $\gamma'$. Thus, there are two different words $v_1, v_2$ in $L_\lambda$ such that $(v_1, v_2)$ is in $\gamma'$. As $v_2 \neq w$, it must be the case that $(v_1, v_2)$ is in $\gamma$, which contradicts the fact that $L$ is error-detecting for $\gamma$. ■

In the next result, for a given language $L$, the symbol $D_L$ denotes the *diagonal relation* $\{(w, w) : w \in L\}$.

**Theorem 2** *For every regular language $L$ there is exactly one $\mathbb{C}$[rat]-maximal error-detecting capability, denoted by $\mu_L$, and is equal to*

$$\mu_L = D_{L_\lambda} \cup [L_\lambda \times (\Sigma^* - L_\lambda)] \cup [(\Sigma^* - L_\lambda) \times \Sigma^*]$$

*Proof.* First one verifies that $L$ is indeed error-detecting for $\mu_L$. Next, for the sake of contradiction, assume that $L$ is error-detecting for some channel $\mu$ that properly contains $\mu_L$; thus, there is a pair $(w, z)$ in $\mu - \mu_L$. This fact leads to a contradiction when we consider the various cases of whether each of $w$ and $z$ is, or is not, in $L_\lambda$. Hence, $\mu_L$ is indeed maximal.

Now suppose that $\gamma$ is a rational maximal error-detecting capability of $L$. We show that the assumption $\mu_L \neq \gamma$ leads to a contradiction. As $\gamma$ and $\mu_L$ are different and maximal, they must be proper subsets of $\mu_L \cup \gamma$, which implies that $L$ is not error-detecting for $\mu_L \cup \gamma$. On the other hand, $L$ must be error-detecting for $\mu_L \cup \gamma$ by Lemma 1; a contradiction. ■

When the language $L$ is given via a deterministic finite automaton $A$, one can compute a transducer realizing $\mu_L$ in time linear with respect to the size of $A$ – this quantity is simply the number of states in $A$. Indeed, given $A$, one can construct in linear time an automaton for $\Sigma^* - L$, and transducers for each of the relations $D_L, L \times (\Sigma^* - L), (\Sigma^* - L) \times \Sigma^*$.

It turns out that the analogue of Theorem 1 for the case of error-correction holds true as well.

**Theorem 3** *Let $L$ be a nonempty language and let $\gamma$ be a channel such that all words of $L$ are possible inputs to $\gamma$, that is, $L \subseteq \operatorname{dom}\gamma$. If $\gamma$ is a $\mathbb{C}$[rat]-maximal error-correcting capability of $L$ then $L$ is maximal error-correcting for $\gamma$.*

*Proof.* Assume for the sake of contradiction that $L$ is not maximal error-correcting for $\gamma$. Then, there is a word $w$ not in $L_\lambda$ such that the language $L' = L \cup \{w\}$ is error-correcting for $\gamma$. We choose a word $v_0$ from $L$ as follows:

- If there is a word $v$ in $L$ such that $(v, w)$ is in $\gamma$ then $v_0$ is any such $v$; hence, $(v_0, w)$ is in $\gamma$.
- If there is no word $v$ in $L$ such that $(v, w)$ is in $\gamma$ then $v_0$ is any word in $L$.

Note that $(v_0, v_0)$ is in $\gamma$, as $L \subseteq \operatorname{dom}\gamma$. Define the channel

$$\gamma' = \gamma \cup \{(w, w), (v_0, w)\}.$$

The channel $\gamma'$ is in $\mathbb{C}$[rat], as the class of rational relations is closed under union. Obviously $L'$ is not error-correcting for $\gamma'$. As $L'$ is error-correcting for $\gamma$, we have that $\gamma$ is a proper subset of $\gamma'$ and, as $\gamma$ is a maximal error-correcting capability of $L$, it follows that $L$ is not error-correcting for $\gamma'$. Thus, there are two different words $v_1, v_2$ in $L_\lambda$ such that $(v_1, z)$ and $(v_2, z)$ are in $\gamma'$, for some word $z$. We obtain a contradiction as follows.

Case 1: At least one of $(v_1, z)$ and $(v_2, z)$ is not in $\gamma$. Without loss of generality, suppose that $(v_1, z)$ is not in $\gamma$. Then, $(v_1, z)$ must be in $\{(w, w), (v_0, w)\}$. As $w$ is not in $L$ and $v_1$ is in $L$, it must be $(v_1, z) = (v_0, w)$. Also, as $v_0 \neq v_2$, the pair $(v_2, w)$ must be in $\gamma$. Then, by the choice of $v_0$, it follows that $(v_0, w)$ must be in $\gamma$, which is a contradiction.

Case 2: Both of $(v_1, z)$ and $(v_2, z)$ are in $\gamma$. This implies that $L$ is not error-correcting for $\gamma$, which is a contradiction. ■

Unlike the case of error-detection, a regular language $L$ can have more than one maximal rational error-correcting capability. To see this, consider the finite language $L = \{00001, 1001\}$ and the channels $\sigma(1, \infty)$ – see Example 1 – and $\delta(1, \infty)$ that consists of all pairs $(w, z)$ such that $z$ results by deleting at most one symbol from $w$. Firstly, note that there is no word $z$ such that both $(00001, z)$ and $(1001, z)$ are in $\delta(1, \infty)$, hence, $L$ is error-correcting for $\delta(1, \infty)$. Similarly, $L$ is error-correcting for $\sigma(1, \infty)$. If there were a unique maximal rational error-correcting capability of $L$, say $\gamma$, then $\gamma$ would include both $\delta(1, \infty)$ and $\sigma(1, \infty)$. Then, however, a contradiction arises when we note that both $(00001, 0001)$ and $(1001, 0001)$ would be in $\gamma$.

# 4 Error Models of SID Channels

The class $\mathbb{C}$[rat] of rational channels is interesting from a theoretical point of view but includes channels that do not relate in any way to physical channels. In this section, we turn our attention to error models consisting of SID channels (with scattered errors) [7, 5, 3]. We establish several theoretical results that are used in the next section to address Problem 1 for various SID error models.

An *SID channel* is specified by an expression of the form $\tau(m, l)$, where $\tau$ is an *error type* – see Section 1 – and $(m, l)$ is a pair of nonnegative integers with $m < l$. This channel consists of all pairs of words $(w, z)$ such that $z$ results by using no more than $m$ errors of type $\tau$ in any segment of length at most $l$ of the input word $w$.

**Example 3** The pair $(w, z)$ is in the channel $(\sigma \odot \delta)(2, 7)$ if and only if $z$ can be obtained by substituting and/or deleting no more than 2 symbols in every segment of length 7 of $w$. For example the pair

$$(100000000, 01000010)$$

is in $(\sigma \odot \delta)(2, 7)$, but the pair

$$(100000000, 01001000)$$

is not in $(\sigma \odot \delta)(2, 7)$ as one has to use more than 2 errors in the prefix 1000000 of 100000000 in order to obtain 01001000.

In [3], SID channels are defined in terms of edit strings (e-strings). An ***edit string*** is a word over the alphabet $E = \{(x/y) : x, y \in \Sigma \cup \{\lambda\} \text{ and } xy \neq \lambda\}$ whose symbols are called *edit operations*. If the edit operation $(x/y)$ is such that $x \neq y$ then $(x/y)$ is called an *error*. 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 $y_1 \cdots y_n$. In the preceding example, the edit string

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

can be used to transform the word $w = 100000000$ into the word $z = 01000010$. In this case, we say that $w$ is the *input part* and $z$ is the *output part* of $h$. An SID channel $\tau(m, l)$ consists of all

pairs of words $(w, z)$ such that $w$ and $z$ are the input and output parts, respectively, of some edit string $h$, where every factor of $h$ of *input size* at most $l$ contains no more than $m$ errors. The input size of an edit string is simply the length of its input part. Note that if $\tau$ contains no insertions, that is errors of the form $(\lambda/x)$, then the input size of the edit string is equal to its length. In the preceding example, as each factor of $h$ of length at most 7 contains no more than 2 errors, we have that $(w, z)$ is indeed in $(\sigma \odot \delta)(2, 7)$.

Addressing Problem 1 requires that we establish a few theoretical results such as bounds on $m$ and $l$, when $\tau(m, l)$ is a maximal error-detecting capability of the language $L(A)$, as well as containment relationships between any two SID channels $\tau(m_1, l_1)$ and $\tau(m_2, l_2)$.

**Lemma 2** *Let $\tau(m_1, l_1)$ and $\tau(m_2, l_2)$ be SID channels with $m_1 \leq m_2$.*

1. *If $l_1 \geq l_2$ then $\tau(m_1, l_1) \subseteq \tau(m_2, l_2)$.*
2. *If $m_1 = m_2$ and $l_1 < l_2$ then $\tau(m_2, l_2) \subsetneq \tau(m_1, l_1)$.*

*Proof.* We prove the first statement and leave the proof of the second one to the reader. Let $(w, z)$ be any element of $\tau(m_1, l_1)$. There is an edit string $h$ whose input and output parts are equal to $w$ and $z$, respectively, and every factor of $h$ of input size at most $l_1$ contains no more than $m_1$ errors. Now let $g$ be any factor of $h$ of input size at most $l_2$. As $l_2 \leq l_1$, the factor $g$ contains no more than $m_1$ errors, hence, no more than $m_2$ errors. Therefore, $(w, z)$ belongs to $\tau(m_2, l_2)$. ■

The above lemma leaves open the case where $m_1 < m_2$ and $l_1 < l_2$. We settle this case when $\tau = \sigma$ and leave the other cases for future research.

**Lemma 3** *Let $\sigma(m_1, l_1)$ and $\sigma(m_2, l_2)$ be SID channels with $m_1 < m_2$ and $l_1 < l_2$. Let $r_1 = \min\{m_1, l_2 \% l_1\}$, where $l_2 \% l_1$ is the remainder of the integer division $l_2/l_1$.*

1. *If $m_2 < \lfloor l_2/l_1 \rfloor m_1 + r_1$ then the channels $\sigma(m_1, l_1)$ and $\sigma(m_2, l_2)$ are subset-incomparable.*
2. *If $m_2 \geq \lfloor l_2/l_1 \rfloor m_1 + r_1$ then $\sigma(m_1, l_1) \subsetneq \sigma(m_2, l_2)$.*

*Proof.* First note that, for the word $z = 1^{m_1+1}0^{l_2-(m_1+1)}$, we have that $(0^{l_2}, z) \in \sigma(m_2, l_2) - \sigma(m_1, l_1)$.

If $m_2 < \lfloor l_2/l_1 \rfloor m_1 + r_1$ then let $u = (1^{m_1}0^{l_1-m_1})^{\lfloor l_2/l_1 \rfloor}1^{r_1}0^{l_2\%l_1-r_1}$. We have that

$$(0^{l_2}, u) \in \sigma(m_1, l_1) - \sigma(m_2, l_2).$$

Indeed, note that $0^{l_2}$ can be written as $(0^{l_1})^{\lfloor l_2/l_1 \rfloor}0^{l_2\%l_1}$. The channel $\sigma(m_1, l_1)$ can apply $m_1$ errors in each factor $0^{l_1}$ and $r_1$ errors in $0^{l_2\%l_1}$. Hence, $(0^{l_2}, u)$ is in $\sigma(m_1, l_1)$. Moreover, this process uses $\lfloor l_2/l_1 \rfloor m_1 + r_1$ errors in $0^{l_2}$. On the other hand, $\lfloor l_2/l_1 \rfloor m_1 + r_1 > m_2$ implies that $(0^{l_2}, u) \notin \sigma(m_2, l_2)$. Hence, the two channels are incomparable.

If $m_2 \geq \lfloor l_2/l_1 \rfloor m_1 + r_1$ then $\sigma(m_1, l_1)$ is a subset of $\sigma(m_2, l_2)$. Indeed, assume for the sake of contradiction that there is a pair $(u, v)$ in $\sigma(m_1, l_1) - \sigma(m_2, l_2)$, and let $h$ be the corresponding edit string. Then there is a factor $g$ of $h$ of length at most $l_2$ containing more than $m_2$ errors. Let $(u_2, v_2)$ be the pair corresponding to $g$. Then $(u_2, v_2)$ is in $\sigma(m_1, l_1) - \sigma(m_2, l_2)$, which implies that we can have no more than $m_1$ errors in every factor of $g$ of length at most $l_1$. Thus, there can be at most $\lfloor l_2/l_1 \rfloor m_1 + r_1$ errors in $g$; a contradiction. ■

The proofs of some of the next results make use of the concept of ***$\sigma$-automaton*** $A^\sigma$, [4], where $A$ is any finite automaton. More specifically, $A^\sigma$ is a finite automaton over the alphabet

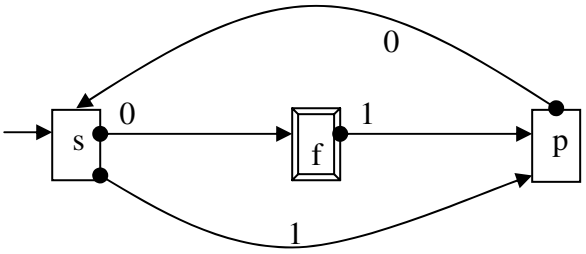


Figure 1: A finite automaton. Final states are denoted using double lines. The start state is indicated using an incoming arrow with no origin state. This automaton accepts the language $(10 + 010)^*0$.

$E_\sigma = \{(x/y) : x, y \in \Sigma\}$ such that the language $L(A^\sigma)$ consists of all edit strings over $E_\sigma$ that transform a word of $L(A)$ into a different word of $L(A)$; that is, all edit strings $(x_1/y_1)\cdots(x_n/y_n)$ with $x_1 \cdots x_n, y_1 \cdots y_n \in L(A)$ and $x_i \neq y_i$ for some index $i$. The automaton $A^\sigma$ results from $A$ using a simple product construction. It consists of transitions of the form $((p_1\$_1q_1)(x/y)(p_2\$_2q_2))$, where $(p_1, x, q_1)$ and $(p_2, y, q_2)$ are two transitions of $A$, and the symbols $\$_1$ and $\$_2$ are in $\{=, \neq\}$ indicating whether the input and output components of the edit string seen so far match or not. The start state is $(p_0 = p_0)$, where $p_0$ is the start state of $A$, and the final states are $(f_1 \neq f_2)$, where $f_1, f_2$ are final states of $A$. As the detailed view of these transitions is not needed here, we simplify the notation by using expressions of the form $(\bar{p}, (x/y), \bar{q})$ for the transitions of $A^\sigma$. We also note that $A^\sigma$ consists of at most $2s_A^2$ states, where $s_A$ is the number of states in $A$, and contains only states that are reachable from the start state – see Fig. 4.

The next statement refers specifically to thin languages, as they are special when it comes to error-detection for channels of the form $\sigma(m, l)$. A language $K$ is called *thin* if any two different words of $K$ have different lengths [11]. Obviously a thin language is error detecting for $\sigma(m, l)$, for every possible values of the parameters $(m, l)$.

**Lemma 4** *Let $A$ be a finite automaton accepting at least two words, and let $\tau(m, l)$ be an SID channel (hence, $m < l$) such that the language $L(A)$ is error-detecting for $\tau(m, l)$.*

1. *If $\tau \neq \sigma$ then $m$ is less than the length of a shortest nonempty word in $L(A)$.*
2. *If $\tau = \sigma$ and $L(A)$ is not thin, then $m < 2s_A^2 - 1$, where $s_A$ is the number of states in $A$.*

*Proof.* The first statement follows from the facts that (i) $\tau$ must contain at least one of $\iota$ and $\delta$, and (ii) if $m \geq |w|$ for some word $w \in L(A) - \{\lambda\}$, then we would get that $(\lambda, w) \in \tau(m, l)$ or $(w, \lambda) \in \tau(m, l)$, depending on whether $\iota$ or $\delta$ is in $\tau$.

For the second statement, as $L(A)$ is not thin, we can choose two different words $w_1, w_2$ of the same length, such that these are shortest with this property. Let $h$ be the edit string in $L(A^\sigma)$ that corresponds to the pair $(w_1, w_2)$. As $h$ must be a shortest word of $L(A^\sigma)$ its length cannot exceed the number of states in $A^\sigma$ minus one; hence, the length of $w_1, w_2$ is at most $2s_A^2 - 1$. Thus, if $m \geq 2s_A^2 - 1$ then $(w_1, w_2) \in \sigma(m, l)$, which is a contradiction. Hence, $m < 2s_A^2 - 1$ as required. ■

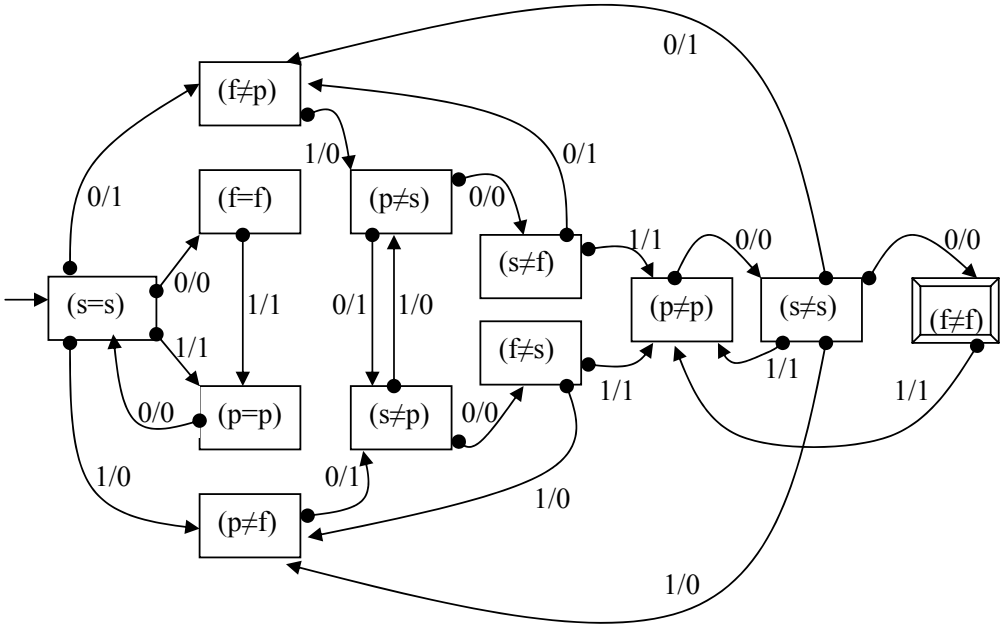


Figure 2: The $\sigma$-automaton corresponding to the automaton in Fig. 1. It accepts all edit strings that transform a word of (10+010)*0 into a different word of $(10 + 010)^*0$ using only substitution errors.

The next result provides an important upper bound on the smallest value of $l$ when a language $L(A)$ is error-detecting for channels of the form $\sigma(m, l)$.

**Theorem 4** *Let $m$ be a nonnegative integer and let $A$ be a finite automaton accepting at least two words. The language $L(A)$ is error-detecting for the channel $\sigma(m, l)$, for some $l > m$, if and only if it is error-detecting for $\sigma(m, 2ms_A^2 + 1)$, where $s_A$ is the number of states in $A$.*

Before we get to the proof of this theorem we define the N-automaton $A^{\mathrm{N}}$ of some finite automaton $A$, and obtain an intermediate lemma that relates the two equivalent statements of the theorem via this N-automaton concept.

Let $A$ be a finite automaton and let $\mathbb{N}_\infty = \mathbb{N}_0 \cup \{\infty\}$, where $\mathbb{N}_0$ denotes the set of nonnegative integers. Let $T_{\neq}$ be the set of transitions of the form $(\bar{p}, (x/y), \bar{q})$ in $A^\sigma$ with $x \neq y$. The automaton $A^{\mathrm{N}}$ consists of the following components.

- States: the elements of $T_{\neq}$. We use symbols of the form $\hat{t}$ for these. Obviously, each such $\hat{t}$ is of the form $(\bar{p}, (x/y), \bar{q})$.
- Alphabet: the set $\mathbb{N}_\infty$.
- Transitions: there is a transition from $(\bar{p}_1, (x_1/y_1), \bar{q}_1)$ to $(\bar{p}_2, (x_2/y_2), \bar{q}_2)$ with label $n$, if there is a path in $A^\sigma$ from $\bar{q}_1$ to $\bar{p}_2$ such that the edit string formed along this path contains no errors, and the supremum of all such paths is equal to $n$.
- Start states: all elements $(\bar{p}, (x/y), \bar{q})$ of $T_{\neq}$ such that there is a path in $A^\sigma$ from its start state to the state $\bar{p}$ and the edit string formed along this path contains no errors.

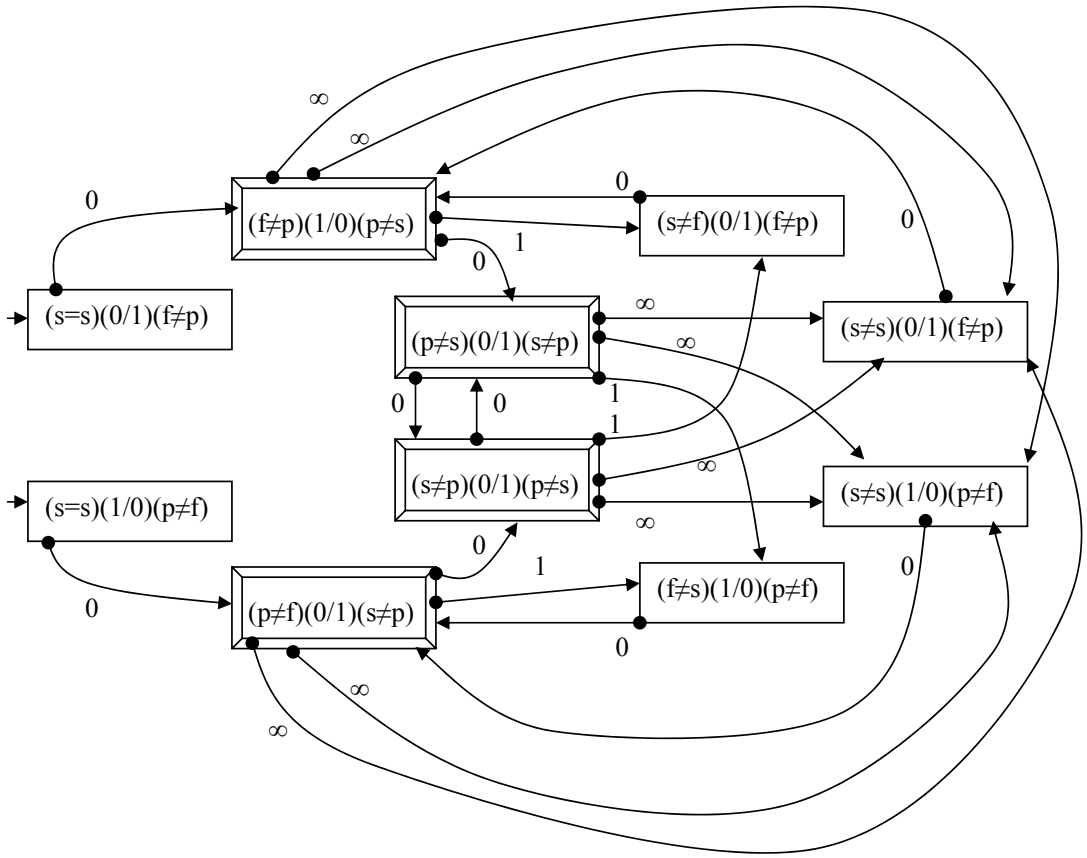


Figure 3: The N-automaton corresponding to the automaton of Fig. 1 – see also Fig. 2.

- Final states: all elements $(\bar{p}, (x/y), \bar{q})$ of $T_{\neq}$ such that there is a path in $A^{\sigma}$ from the state $\bar{q}$ to a final state of $A^{\sigma}$ and the edit string formed along this path contains no errors.

**Remark 2.** By the above construction, an accepting computation of $A^{\mathrm{N}}$ is a sequence

$$\hat{T} = (\hat{t}_0, n_1, \hat{t}_1, \ldots, n_k, \hat{t}_k)$$

such that each state $\hat{t}_i$ is of the form $(\bar{p}_i, x_i/y_i, \bar{q}_i)$. In this case, there is an edit string accepted by $A^{\sigma}$ of the form

$$h = h_0(x_0/y_0)h_1 \cdots h_k(x_k/y_k)h_{k+1}$$

such that each $h_i$ is in $E_0^*$, where $E_0 = \{x/x : x \in \Sigma\}$, and $|h_j| \leq n_j$ for $j = 1, \ldots, k$. Conversely, if the edit string $h$ shown above is accepted by $A^{\sigma}$ then there is a computation $\hat{T}$ of $A^{\mathrm{N}}$ as shown above such that each state $\hat{t}_i$ is of the form $(\bar{p}_i, x_i/y_i, \bar{q}_i)$, and $|h_j| \leq n_j$ for $j = 1, \ldots, k$, and each $n_j$ is not $\infty$ exactly when no path between $\bar{q}_{j-1}$ and $\bar{p}_j$ contains a cycle.

**Lemma 5** *Let $A$ be a finite automaton accepting at least two words and let $m$ be a positive integer. If the language $L(A)$ is error-detecting for $\sigma(m, l)$, for some $l > m$, then the language accepted by $A^{\mathrm{N}}$ is a subset of $\mathbb{N}_{\infty}^* \mathbb{N}_0^m \mathbb{N}_{\infty}^*$*

*Proof.* Assume for the sake of contradiction that $A^{\mathrm{N}}$ has an accepting computation

$$\hat{T} = (\hat{t}_0, n_1, \hat{t}_1, \ldots, n_k, \hat{t}_k)$$

with the property that, if for some $r \geq 0$ and $t \geq 2$ all values $n_{r+1}, \ldots, n_{r+t-1}$ are in $\mathbb{N}_0$, then $t - 1 < m$. Then there is an edit string

$$h = h_0(x_0/y_0)h_1 \cdots h_k(x_k/y_k)h_{k+1}$$

as specified in Remark 2. In particular, we can choose $h$ such that $|h_j| = n_j$ if $n_j \neq \infty$, and $|h_j| > l$ if $n_j = \infty$, for all $j = 1, \ldots, k$. As $L(A)$ is error-detecting for $\sigma(m, l)$, there is a factor $g$ of $h$ of length at most $l$ containing a number $t$ of errors such that $t > m$. Now $g$ must be of the form

$$h'_r(x_r/y_r)h_{r+1} \cdots (x_{r+t-1}/y_{r+t-1})h'_{r+t}$$

for some $r \geq 0$. As $|g| \leq l$, we have that $|h_{r+1}|, \ldots, |h_{r+t-1}| \leq l$. By the choice of the $h_j$'s in $h$, it must be the case that all of $n_{r+1}, \ldots, n_{r+t-1}$ are in $\mathbb{N}_0$. Also, by the assumption about the computation $\hat{T}$, we get that $t - 1 < m$ which contradicts $t > m$ when $g$ was chosen. ■

We proceed now with the proof of the theorem.

*Proof of Theorem 4.* The 'if' part is trivial. For the 'only if' part, it is sufficient to show that every edit string $h$ accepted by $A^\sigma$ has a factor $g$ of length at most $2ms_A^2 + 1$ containing more than $m$ errors. So, let $h$ be an edit string as in Remark 2. Then, there is an accepting computation $\hat{T}$ as in Remark 2. By Lemma 5, there must be an index $r$ such that all of $n_{r+1}, \ldots, n_{r+m}$ are in $\mathbb{N}_0$. As each $n_j$ is the length of an edit string along a path in $A^\sigma$ with no cycles, we have that each $n_j$ cannot exceed $2s_A^2 - 1$, which is the number of states in $A^\sigma$ minus 1. Now consider the factor

$$g = (x_r/y_r)h_{r+1} \cdots h_{r+m}(x_{r+m}/y_{r+m})$$

of $h$. Then $g$ contains exactly $m+1$ errors and is of length at most $1+m+m(2s_A^2-1) = 2ms_A^2+1$, as required. ■

## 5 Computing Maximal Error-detecting Capabilities

In this section we discuss how to solve Problem 1 for some of the error models considered in previous sections. We recall some existing results that are needed here.

**Theorem 5** *[5] The following problem is decidable in polynomial time.*
*Input: a finite automaton $A$ and a finite transducer realizing some channel $\gamma$.*
*Output: Y/N, depending on whether the language $L(A)$ is error-detecting for $\gamma$.*

**Proposition 1** *[5, 3] For each error type $\tau$ and integers $m$ and $l$, with $0 \leq m < l$, 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 $m$, see [3] for a relevant discussion.

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

In this error model, the parameters $\tau$ and $l$ are fixed and, therefore, there are only finitely many channels:

$$\tau(0,l), \ldots, \tau(l-1,l).$$

In this case, Problem 1 can be solved using Proposition 1 and Theorem 5 and Lemma 2. Indeed, for each possible value $0, \ldots, l-1$ of $m$, construct a finite transducer realizing $\tau(m,l)$ and test whether the given language $L(A)$ is error-detecting for $\tau(m,l)$. The process stops when the first $m > 0$ is found such that $L(A)$ is not error-detecting for $\tau(m,l)$. In this case, the output would be $\tau(m-1,l)$. We can speed up this process when the values of $m$ are visited using binary search.

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

In this error model, the parameters $\tau$ and $m$ are fixed. We want to find the smallest $l$ (if any) such that $L(A)$ is error-detecting for $\tau(m,l)$. This turns out to be difficult in general. However, we have managed to settle the case where $\tau = \sigma$ using Theorem 4. Indeed, as in Subsection 5.1, one needs to consider only finitely many channels:

$$\tau(m,m+1), \ldots, \tau(m, 2ms_A^2 - 1),$$

where recall that $s_A$ is the number of states in the given finite automaton $A$.

### 5.3 The error model $\mathbb{C}_\tau = \{\tau(m,l) : \textbf{ for any } m \textbf{ and } l \textbf{ with } m < l\}$

We shall consider first the case where the language $L(A)$ is thin and $\tau = \sigma$ – see Section 4. In this case it is evident that, for every channel $\sigma(m,l)$, if $(w,z)$ is in $\sigma(m,l)$ then $w$ and $z$ are of the same length; hence, the channel cannot transform a word of $L(A)$ to another word in $L(A)$. Thus $L(A)$ is error-detecting for $\sigma(m,l)$ for all $(m,l)$. Moreover, there is no $\mathbb{C}_\sigma$-maximal error-detecting capability of $L(A)$. Indeed, if $\sigma(m,l)$ is such a maximal capability then, as $\sigma(m,l) \subseteq \sigma(l-1,l)$, it must be $m = l-1$. At the same time, as $\sigma(l-1,l) \subsetneq \sigma(l,l+1)$ by Lemma 3(2), it must be that $L(A)$ is not error-detecting for $\sigma(l,l+1)$; a contradiction.

In the sequel we assume that either $\tau = \sigma$ or $L(A)$ is not a thin language – note that one can effectively test in quadratic time (with respect to the size of $A$) whether $L(A)$ is thin [5]. According to Lemma 4, there is an upper bound $M(\tau, A)$ on $m$ when $L(A)$ is error-detecting for any $\tau(m,l)$. Thus we can restrict our attention to channels of the form $\tau(m,l)$, where $m \in \{0, \ldots, M(\tau,A)\}$; hence, the problem reduces to the case of the error model of Subsection 5.2, which has been resolved for the case of $\tau = \sigma$ via Theorem 4. More specifically, the process is as follows:

- For each pair of values $(m,l)$ such that $m \in \{0, \ldots, M(\tau,A)\}$ and $l \in \{m+1, \ldots, 2ms_A^2 + 1\}$, construct a transducer for $\sigma(m,l)$ using Proposition 1, and test whether $L(A)$ is error-detecting for $\sigma(m,l)$ using Theorem 5.
- Let $C$ be the set of channels that are tested positive. Then, resolve all containment relationships of channels in $C$ using Lemma 2 and Lemma 3, and output all channels that are $C$-maximal.

Hence we have established the following result.

**Theorem 6** *The following problem is computable.*
*Input: A finite automaton $A$ accepting at least two words.*
*Output: The set of all $\mathbb{C}_\sigma$-maximal error-detecting capabilities of $L(A)$.*

### 5.4 Error models of homophonic channels

Homophonic channels are defined in [2] based on the idea of multivalued encodings [10, 1]. A multivalued encoding of a finite language $K$ is a mapping $\mu$ of $K$ into the finite nonempty subsets of $\Sigma^*$; that is, $\mu(w)$ is a finite nonempty set of words, for each $w$ in $K$. The homophonic channel defined by $K$ is the set of pairs of words $(w_1 \cdots w_n, z_1 \cdots z_n)$ such that $n \geq 0$, each $w_i$ is in $K$, and each $z_i$ is in $\mu(w_i)$. Obviously the words of the language $K^*$, or any subset of that, are meant to be the possible inputs to this channel.

In this paper we separate the language from the channel by defining the following homophonic-like channel, for every error type $\tau$ and parameters $m, l$:

$$\tau_{\mathrm{h}}(m,l) = \{(w_1 \cdots w_n, z_1 \cdots z_n) : n \geq 1, |w_1| = \cdots = |w_{n-1}| = l, |w_n| \leq l, \forall i (w_i, z_i) \in \tau(m, \infty)\}$$

Thus, the channel $\tau_{\mathrm{h}}$ applies errors in any input word $w$, say, as follows: no more than $m$ errors in the first block of $l$ symbols of $w$, then, independently of what happened in the first block, no more than $m$ errors in the second block of $l$ symbols of $w$, and so on. If the length of $w$ is not a multiple of $l$ then the last block in the above process would be of length less than $l$.

The main result here concerns the following error model

$$\mathbb{C}^1_{\tau_{\mathrm{h}}}[l] = \{\tau_{\mathrm{h}}(m,l) : \text{ for any } m \text{ with } m < l\}$$

Obviously this is a finite error model such that $\tau_{\mathrm{h}}(m_1, l)$ is a subset of $\tau_{\mathrm{h}}(m_2, l)$, for any $m_1$ and $m_2$ with $m_1 < m_2 < l$.

**Lemma 6** *For each error type $\tau$ and integers $m$ and $l$, with $0 \leq m < l$, there is (effectively) a finite transducer realizing the homophonic channel $\tau_{\mathrm{h}}(m,l)$. Moreover, the size of the transducer – and time to construct it – is of order $O(ml)$.*

*Proof.* It is sufficient to construct a finite automaton accepting all edit strings $h = h_1 \cdots h_{n-1} h_n$ such that each of $h_1, \ldots, h_{n-1}$ has input size $l$, $h_n$ has input size at most $l$, and no $h_i$ contains more than $m$ errors. To this end, first construct an automaton $A_{m,l}$ of $(m+1)(l+1)$ states accepting all edit strings of input size $l$ having no more than $m$ errors. The states of this automaton are all pairs $(i,j)$ with $0 \leq i \leq m$ and $0 \leq j \leq l$. Such a state represents the fact that the edit string seen so far has input size $j$ and exactly $i$ errors. The start state is $(0,0)$ and the final states are the pairs $(i,l)$ with $0 \leq i \leq m$. The transitions are defined such that the meaning of the state is preserved. For example, for $x, y \in \Sigma$ with $x \neq y$, $((i,j),(x/y),(i+1,j+1))$ is a transition when $i < m$ and $j < l$, and $((i,j),(\lambda/y),(i+1,j))$ is a transition when $i < m$.

Now let $B_1$ and $B_2$ be two copies of $A_{m,l}$ with the following modifications. In $B_1$, each final state has a transition to the state $(0,0)$ with label $(\lambda/\lambda)$, where $(\lambda/\lambda)$ denotes the empty edit string. Thus, $B_1$ accepts the language $L(A_{m,l})^+$. In $B_2$, all states are final; hence, $B_2$ accepts all edit strings that are prefix factors of edit strings in $L(A_{m,l})$. The reader can now verify that the required automaton can be constructed easily using $B_1$ and $B_2$ as components. ■

**Theorem 7** *The following problem is computable in polynomial time.*

*Input: A finite automaton $A$ accepting at least two words and a positive integer $l$.*

*Output: The $\mathbb{C}^1_{\tau_{\mathrm{h}}}[l]$-maximal error-detecting capability of $L(A)$.*

*Proof.* As the error model $\mathbb{C}^1_{\tau_{\rm h}}[l]$ is finite, for each $m = 1, \ldots, l-1$, one constructs the channel $\tau_{\rm h}(m, l)$ in polynomial time using Lemma 6, and then tests, in polynomial time whether $L(A)$ is error detecting for $\tau_{\rm h}(m, l)$ using Theorem 5. The process stops when the first $m > 0$ is found such that $L(A)$ is not error-detecting for $\tau_{\rm h}(m, l)$. In this case, the output would be $\tau_{\rm h}(m-1, l)$. We can speed up this process when the values of $m$ are visited using binary search. In any case, the process is of polynomial time complexity. ■

The study of the error models

$$\mathbb{C}^2_{\tau_{\rm h}}[m] = \{\tau_{\rm h}(m, l) : \text{ for any } l \text{ with } m < l\}$$

and

$$\mathbb{C}_{\tau_{\rm h}} = \{\tau_{\rm h}(m, l) : \text{ for any } m, l \text{ with } m < l\}$$

appears to require extra effort and is left for future research.

# 6 Discussion

We have introduced the concepts of maximal error-detecting and -correcting capability of a given language, with respect to a certain error model, and have argued that these concepts are meaningful at least from a theoretical point of view. As stated in the introduction, our present study is not meant to be exhaustive. So, we propose a few possible directions for future research.

- Obtain the analogue of Theorem 6 for error types other than $\sigma$. To this end, for each error type $\tau$, resolve the containment relations between any two channels $\tau(m_1, l_1)$ and $\tau(m_2, l_2)$.
- Investigate further error models of homophonic channels.
- Investigate to what extend the algorithmic methods for computing various maximal error-detecting capabilities can be improved in terms of efficiency.
- Apply the algorithms to real world languages such as gene languages and codes for data communications.
- In view of Remark 1 in the introduction, investigate when a given rational relation $\mu$ can be written in the form $\gamma^{-1} \circ \gamma$, where $\gamma$ is a rational relation. Note that the rational relation $\{(0, 1)\}$ cannot be written in that form (otherwise, $\{(1, 0)\}$ would have to be in the relation as well).
- Investigate maximal error-correcting capabilities of languages. In particular whether Remark 1 can aid in any way towards this end.

# References


[1] R. M. Capocelli, L. Gargano and U. Vaccaro. Decoders with Initial State Invariance for Multivalued Encodings, *Theoretical Computer Science* **86** (1991) 365–375.

[2] H. Jürgensen, S. Konstantinidis. Codes. In [9], 511–607.

[3] L. Kari, S. Konstantinidis. Descriptional Complexity of Error/Edit Systems, *Journal of Automata, Languages and Combinatorics* 9 (2004) 2/3, 293-309.

[4] L. Kari, S. Konstantinidis, S. Perron, G. Wozniak, J. Xu. Computing the Hamming Distance of a Regular Language in Quadratic Time, *WSEAS Transactions on Information Science & Applications* 1 (2004), pp 445-449. Also in: *Proceedings of "8th WSEAS International Conference on Computers, Vouliagmeni, Greece, July 12-15, 2004."*

[5] S. Konstantinidis. Transducers and the Properties of Error-Detection, Error-Correction and Finite-Delay Decodability, *Journal of Universal Computer Science* 8 (2002), 278-291.

[6] S. Konstantinidis. 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 113-116.*

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

[8] S. Roman. *Coding and Information Theory.* Springer-Verlag, New York, 1992.

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

[10] K. Sato. A decision procedure for the unique decipherability of multivalued encodings. *IEEE Transactions on Information Theory* **25** (1979) 356–360.

[11] S. Yu. Regular Languages. In [9], 41–110.