On the Decidability of the Error-Detection Property*

Stavros Konstantinidis
Department of Mathematics and Computing Science
Saint Mary's University
Halifax, Nova Scotia, B3H 3C3, Canada
s.konstantinidis@stmarys.ca

Abstract: When the words of a language are communicated via a noisy channel, the language property of error-detection ensures that no word of the language can be transformed to another word of the language. In this work we model channels using finite transducers and we show that the problem of whether a given regular language is error-detecting for a given rational channel can be decided in polynomial time. On the other hand, the problem is undecidable for context-free languages. Moreover, we demonstrate constructively that the class of rational channels includes all the SID-channels which can be used to model the effects of various error combinations. Finally, we briefly discuss the properties of error-correction and unique decodability for rational channels.

1 Introduction

Consider a language whose words are communicated via a noisy channel capable of altering those words. If the language is error-detecting *for* the given channel then no word of the language can be transformed to another word of the language. This property is basic as it allows one to determine whether or not the received information is correct. In principle, a communications language could be any language, but usually it is a finite set of words or a free monoid generated by a code (a uniquely decodable set of words). A channel could be any storage or communications medium possibly capable of replacing symbols with other symbols, or even inserting and deleting symbols in the transmitted/stored words.

*Research partially supported by Grant OGP220259 of the Natural Sciences and Engineering Research Council of Canada

In this work we use finite transducers to model channels. In Section 2 we give the definitions of channel, error-detection, finite automaton and transducer. In Section 3, we consider the problem of deciding whether a given language is error-detecting for a given rational channel. When the given language is regular it is shown that the problem is decidable in polynomial time. On the other hand, the problem is undecidable for context-free languages. In Section 4, we consider the class of SID-channels which can be used to model various combinations of substitution, insertion, and deletion errors. For example, a channel that permits up to 3 substitutions in any 35 consecutive symbols of a message and a total of at most 2 insertions and deletions in any 52 consecutive symbols is an SID-channel. We show constructively that every SID-channel can be realized by a transducer. Finally, in Section 5 we discuss the notions of error-detection and unique decodability in the general framework of rational channels.

2 Basic Notions and Notation

An alphabet is a non-empty set of symbols. If V is an alphabet then V^* is the set of all words over V, including the empty word denoted by 1. We use the notation $|v|$ for the length of the word v. A language is any set of words. In the sequel we shall use the symbols X and Y to denote two finite alphabets.

Channels and Error-Detection

A *channel (over X)* is a binary relation $\gamma \subseteq X^* \times X^*$. When $(x, y) \in \gamma$ it is meant that the message (word) y can be received from x through the channel γ. Note that, in general, the channel is noisy, meaning that, for $(x, y) \in \gamma$, it is possible that $x \neq y$; that is, y is received from x with errors. A channel γ is called *domain preserving* if $(x, x) \in \gamma$ for every word x in $\{x \mid (x, y) \in \gamma, \text{ for some } y\}$; that is, it is possible that a transmitted message can be received through γ with no errors. *In this paper, we only consider channels which are domain preserving and we agree that the term channel means a domain preserving channel.*

A channel γ is called *rational* if it is a rational subset of $X^* \times X^*$ – see page 55 of [2]; or equivalently, if γ is realized by a finite transducer – see below. For a word x, we write $\langle x \rangle_\gamma$ to denote the set of all possible words that can be received from x through γ; that is, $\langle x \rangle_\gamma = \{y \mid (x, y) \in \gamma\}$. A language L is called *error-detecting for* γ – see [9] – if the following

condition is satisfied

$$\forall\, x, y \in L \cup \{1\},\ y \in \langle x \rangle_\gamma \implies x = y.$$

The meaning of the above condition is as follows: If a word y is received and y is in L then y must be equal to the transmitted word x; therefore y can be processed correctly. Moreover, a non-empty word of L cannot be received from the empty word, and the empty word cannot be received from a non-empty word of L. Of particular interest is the case where the language L is a free monoid generated by a code K; that is, $L = K^*$. The code K is called $(\gamma, *)$-*detecting*, for some channel γ, if the language K^* is error-detecting for γ – see [5].

Automata and Transducers

For the basic definitions of finite automata and transducers we follow [2]: A *deterministic finite automaton* A is a quintuple (X, Q, q_0, Q_+, δ) such that X is a finite alphabet, Q is the (finite) set of states, $q_0 \in Q$ is the initial state, $Q_+ \subseteq Q$ is the set of final states, and δ is the transition function of $Q \times X$ into Q. In this paper we assume that every deterministic automaton A is complete and that Q and X are always disjoint. A *transition* of A is a word pap' in QXQ such that $p' = \delta(p, a)$. A *computation* of A is a word of the form $p_0 a_1 p_1 a_2 p_2 \cdots p_{n-1} a_n p_n$, for some non-negative integer n, such that $p_{i-1} a_i p_i$ is a transition of the automaton for all $i = 1, \ldots, n$. If p_0 is the initial state and p_n is a final state of the automaton then the computation is called *successful*. The *label* of a computation is the word $a_1 a_2 \cdots a_n$ formed by concatenating the symbols of X appearing in the computation. If $n = 0$ the label of the computation is the empty word. We write $L(A)$ to denote the *language accepted by* A; that is the set of labels appearing in the successful computations of the automaton. The automaton is called *trim* if every state of the automaton can be reached from the initial state.

A *finite transducer* T is a sextuple $(X, Y, Q, q_0, Q_+, \Delta)$ such that X is a finite alphabet – the input alphabet –, Y is a finite alphabet – the output alphabet –, Q is the (finite) set of states, $q_0 \in Q$ is the initial state, $Q_+ \subseteq Q$ is the set of final states, and Δ is a finite subset of $Q \times X^* \times Y^* \times Q$ – the *set of transitions* of T. As before, we always assume that Q and $X \cup Y$ are disjoint. Moreover, we prefer to write the elements of Δ as words of the form px/yp'. A *computation* of T is a word of the form $p_0 x_1 / y_1 p_1 x_2 / y_2 p_2 \cdots p_{n-1} x_n / y_n p_n$, for some non-negative integer n, such that $p_{i-1} x_i / y_i p_i$ is a transition of T for all $i = 1, \ldots, n$. If p_0 is the initial state and p_n is a final state of the transducer then the computation is called *successful*. The *label* of a computation is the pair of words

$(x_1x_2\cdots x_n, y_1y_2\cdots y_n) \in X^* \times Y^*$ formed by concatenating the words of X^* and the words of Y^* appearing in the computation. If $n = 0$ the label of the computation is $(1, 1)$. We write $R(T)$ to denote the *relation realized by* T; that is the set of labels appearing in the successful computations of the transducer. Moreover, for a word x in X^*, we write $T(x)$ for the set $\{y \in Y^* \mid (x, y) \in R(T)\}$; that is, the set of all possible outputs of the transducer T when x is used as input. As before, the transducer is called *trim* if every state appears in some successful computation of the transducer. The transducer is said to be in *standard form* if $x \in X \cup \{1\}$ and $y \in Y \cup \{1\}$ in every transition px/yp' of the transducer. It can be shown that for every finite transducer T there is a finite transducer T' in standard form such that $R(T) = R(T')$ – see [2]. Finally, a transducer $T = (X, X, Q, q_0, Q_+, \Delta)$ is called a *channel transducer*, if the relation $R(T)$ is a channel.

3 Deciding Error-Detection

In this section we consider the problem of deciding, for given channel γ and language L, whether the language is error-detecting for γ. We consider the cases where L is regular and L is context-free. For the channel we assume it is represented using a transducer. We show that for regular languages the problem is decidable in polynomial time. On the other hand, the problem is not decidable for context-free languages. As a first step we obtain a necessary and sufficient condition for a given language L to be error-detecting for a given channel γ. The condition involves the relation γ/L which describes the effects of the channel γ on the words of L.

Definition 1 *Let $\rho \subseteq X^* \times Y^*$ be a binary relation.*

(i) The relation ρ is called functional if the following holds for every $x \in X^*$ and $y_1, y_2 \in Y^*$

$$(x, y_1) \in \rho,\ (x, y_2) \in \rho \implies y_1 = y_2$$

(ii) The relation ρ^{-1} consists of all the pairs (y, x) such that (x, y) is in ρ.

(iii) Let M be a language. The relation ρ/M consists of all the pairs (x, y) in ρ with $x \in M$; that is $\rho/M = \rho \cap (M \times Y^*)$.

Lemma 1 *Let γ be a channel and let L be a language. The language L is error-detecting for γ if and only if the relation $((\gamma/L_1)^{-1}/L_1)^{-1}$ is functional, where $L_1 = L \cup \{1\}$.*

Proof. First note that $(x, y) \in ((\gamma/L_1)^{-1}/L_1)^{-1}$ if and only if $(x, y) \in \gamma$ and $x, y \in L_1$. For the 'only if' part assume L is error-detecting for γ and consider any two pairs (x, y_1) and (x, y_2) in $((\gamma/L_1)^{-1}/L_1)^{-1}$. Then, $x, y_1, y_2 \in L_1$ and $y_1 \in \langle x \rangle_\gamma$ and $y_2 \in \langle x \rangle_\gamma$. By the assumption, $y_1 = x$ and $y_2 = x$. Hence, $y_1 = y_2$.

For the 'if' part, assume $((\gamma/L_1)^{-1}/L_1)^{-1}$ is functional and consider any words x, y in L_1 such that $y \in \langle x \rangle_\gamma$. Then, $(x, y) \in \gamma$. Moreover, as $(x, x) \in \gamma$, the assumption implies $x = y$. Hence, L is error-detecting for γ. □

If the channel γ is rational and the language L is regular then the relation $((\gamma/L_1)^{-1}/L_1)^{-1}$ is rational as well. This follows from the fact that the relation ρ/M is rational for every rational relation ρ and every regular language M. We prove this fact in two ways. First using results from [2] and, then, using a *constructive* proof that involves a finite transducer and a finite automaton representing ρ and M, respectively.

Proposition 1 *Let X and Y be alphabets. For every rational relation $\rho \subseteq X^* \times Y^*$ and for every regular language $M \subseteq X^*$, the relation ρ/M is rational.*

Proof. First, recall that $\rho/M = \rho \cap (M \times Y^*)$. Now both M and Y^* are recognizable – see Example 1.3 on page 52 of [2]. This implies that the set $M \times Y^*$ is a recognizable subset of $X^* \times Y^*$ – see Theorem 1.5 on page 54 of [2]. Then the claim follows when we note that the intersection of a recognizable and a rational subset of $X^* \times Y^*$ is always rational – see Proposition 2.6 on page 57 of [2]. □

Given a transducer T and a finite automaton A, we define a transducer T/A with the property that $(x, y) \in R(T/A)$ if and only if $(x, y) \in R(T)/L(A)$, and we show that the construction of T/A can be realized in polynomial time. Following [1], the size $|A|$ of an automaton A is the number of transitions of A. The size $|T|$ of a transducer T is the sum of the sizes of its transitions, where the size of a T-transition px/yp' is $1 + |xy|$.[1] The construction is based on the well-known construction of

[1] We note that [1] uses $|xy|$ for the size of the transition px/yp'. We prefer $1 + |xy|$, however, to account for the case of $x = y = 1$ – this does not affect the results of [1] which are used here.

a deterministic finite automaton that accepts the intersection of the languages of two given deterministic finite automata – see [11] for instance.

Proposition 2 *The following problem is computable in time* $O(|T||A|)$.

Input: A trim transducer T *in standard form and a trim deterministic finite automaton* A *with the same input alphabets.*

Output: The transducer T/A *such that* $R(T/A) = R(T)/L(A)$.

Moreover, the size of the transducer T/A *is* $\Theta(|T||A|)$.

Proof. Let $T = (X, Y, Q, q_0, Q_+, \Delta)$ be the given transducer and let $A = (X, P, p_0, P_+, \delta)$ be the given automaton. We construct the transducer $T/A = (X, Y, S, s_0, S_+, \Gamma)$ as follows:

- $S = QP$, $s_0 = q_0 p_0$, $S_+ = Q_+ P_+$;
- Initialize the set of transitions Γ of T/A to $\emptyset$. Then, for each transition qx/yq' in Δ and for each state p in P, if $x \neq 1$ insert $qpx/yq'p'$ in Γ, where $p' = \delta(p, x)$; else insert $qp1/yq'p$ in Γ.

For the time complexity of the construction we note that S can be computed in time $O(|P||Q|)$ and Γ requires time $O(|P||T|)$ assuming δ is implemented using a hash table (or array) with $O(1)$ time for accessing the entries of the table. As the given automaton and transducer are trim, it follows that $|P| = \Theta(|A|)$ and $|Q| = O(|T|)$. Hence, the construction can be performed in time $O(|T||A|)$. Moreover, the size of T/A is $|P||T| = \Theta(|A||T|)$.

For the correctness of the construction we need to show $(x, y) \in R(T/A)$ if and only if $x \in L(A)$ and $(x, y) \in R(T)$. For the 'if' part, consider a pair (x, y) in $R(T)$ such that $x \in L(A)$. Then, there is a successful computation $q_0 x_1/y_1 q_1 \cdots x_n/y_n q_n$ of T and a successful computation $p_0 z_1 p_1 \cdots z_m p_m$ of A such that $x_1 \cdots x_n = z_1 \cdots z_m = x$, $y_1 \cdots y_n = y$, and $x_i \in X \cup \{1\}$ for $i = 1, \ldots, n$, and $z_j \in X$ for $j = 1, \ldots, m$. Hence, $|x| = m \leq n$. Then, there are indices $i_1, \ldots, i_m$ with $1 \leq i_1 < \cdots < i_m \leq n$ such that $x = x_{i_1} \cdots x_{i_m}$ and $x_{i_l} = z_l$ for all $l = 1, \ldots, m$. Now consider the word $q_0 r_0 x_1/y_1 q_1 r_1 \cdots x_n/y_n q_n r_n$ such that $r_0 = \cdots = r_{i_1 - 1} = p_0$, $r_{i_1} = \cdots = r_{i_2 - 1} = p_1$, $\ldots$, $r_{i_{m-1}} = \cdots = r_{i_m - 1} = p_{m-1}$, $r_{i_m} = \cdots = r_n = p_m$. By the construction of T/A, it follows that $q_0 r_0 x_1/y_1 q_1 r_1 \cdots x_n/y_n q_n r_n$ is a successful computation of T/A and, therefore, $(x, y) \in R(T/A)$ as required.

For the 'only if' part of the correctness, consider a successful computation $q_0p_0x_1/y_1\cdots x_n/y_nq_np_n$ of T/A with $q_np_n \in Q_+P_+$, $x_1\cdots x_n = x$, and $y_1\cdots y_n = y$. By construction, it follows that $q_0x_1/y_1\cdots x_n/y_nq_n$ is a computation of T which implies $(x,y) \in R(T)$ as q_n is a final state of T. Now for the word $p_0x_1p_1\cdots x_np_n$, one has that, for every $i = 1,\ldots,n$, $p_{i-1}x_ip_i$ is either a transition of the automaton A if $x_i \neq 1$, or equal to the word $p_{i-1}p_{i-1}$ if $x_i = 1$. Then, by removing from each factor $p_{i-1}p_{i-1}$ of $p_0x_1p_1\cdots x_np_n$ the first p_{i-1} one obtains a computation $p_{i_0}x_{i_1}p_{i_1}\cdots x_{i_k}p_{i_k}$ of A which is successful, $p_{i_0} = p_0$, and $x_{i_1}\cdots x_{i_k} = x$. Hence, $x \in L(A)$ as required. □

The following statement follows easily from the definition of a transducer – see also [11].

Lemma 2 *Given a transducer $T = (X,Y,Q,q_0,Q_+,\Delta)$, consider the transducer $T^{-1} = (Y,X,Q,q_0,Q_+,\Gamma)$ such that $py/xp' \in \Gamma$ if and only if $px/yp' \in \Delta$. Then, $R(T^{-1}) = R(T)^{-1}$ and $|T^{-1}| = |T|$.*
□

Theorem 1 *The following problem is decidable in time $O(|T|^2|A|^4)$.*

Input: A trim channel transducer T in standard form and a trim deterministic finite automaton A with the same input alphabets.

Output: "YES" or "NO" depending on whether the language $L(A)$ is error-detecting for the channel $R(T)$.

Proof. The algorithm is as follows

(a) Construct a trim deterministic finite automaton A_1 such that $L(A_1) = L(A) \cup \{1\}$.

(b) Construct the transducer $(T/A_1)^{-1}$ using Proposition 2 and Lemma 2.

(c) Construct the transducer $((T/A_1)^{-1}/A_1)^{-1}$ as above.

(d) Decide whether $((T/A_1)^{-1}/A_1)^{-1}$ is functional.

As $\{1\}$ can be accepted by a deterministic finite automaton with two states, step (a) can be performed in time $O(|A|)$ and the size of the automaton A_1 is $O(|A|)$ – see [6], for instance. The effectiveness of step (d) has been shown in [10]. Moreover, in [1] it is shown that transducer functionality can be decided in time quadratic with respect to the size of the transducer. Hence, step (d) can be performed in time

$O(|((T/A_1)^{-1}/A_1)^{-1}|^2) = O(|T/A_1|^2|A|^2) = O(|T|^2|A|^4)$ which dominates the complexity of the algorithm. The correctness of the algorithm follows by Lemma 1 when we note that $R(((T/A_1)^{-1}/A_1)^{-1}) = ((R(T)/L(A_1))^{-1}/L(A_1))^{-1}$. □

When K is a regular code then K^* is a regular language. Hence, the following obtains easily from the above theorem.

Corollary 1 *The following problem is decidable.*

Input: A trim channel transducer T in standard form and a trim deterministic finite automaton A accepting a code.

Output: "YES" or "NO" depending on whether the code $L(A)$ is $(R(T), *)$-detecting.

□

We note that using the algorithm in Theorem 1, the above problem is not guaranteed to be decidable in polynomial time. This follows from the fact that there are deterministic finite automata A for which the size of any deterministic automaton accepting $L(A)^*$ is exponential with respect to $|A|$ – see [6].

Proposition 3 *The following problem is undecidable.*

Input: A channel transducer T and a context-free grammar G.

Output: "YES" or "NO" depending on whether the language $L(G)$ is error-detecting for the channel $R(T)$.

Proof. We assume the problem is decidable and obtain a contradiction by showing that the Post Correspondence Problem (PCP) is decidable as well. We use the following version of PCP – see [4]: Given an alphabet $\Sigma = \{a_1, \ldots, a_n\}$ and two morphisms $g, h : \Sigma^* \to \{s, t\}^*$ decide whether $E(g, h) = \{z \in \Sigma^+ \mid g(z) = h(z)\}$ is empty. Using the assumption we can construct the following algorithm whose input are the morphisms g and h:

(a) Let T be the transducer $(X, X, \{q_0\}, q_0, \{q_0\}, \Delta)$, where $X = \{s, t\} \cup \{\$, \#, \&\}$, with $\{\$, \#, \&\} \cap \{s, t\} = \emptyset$, and $\Delta = \{q_0 x / x q_0 \mid x \in X\} \cup \{q_0\$/\&q_0\}$. Thus, T leaves every input symbol unchanged, except possibly for \$ which could be replaced with &.

(b) Let $I = \{1, \ldots, n\}$ and let G be a context-free grammar generating the language $L_g \cup L_h$, where

$L_g = \{g(a_{i_1} \cdots a_{i_k})\$s^{i_k}\# \cdots \#s^{i_1} \mid k \geq 1; i_1, \ldots, i_k \in I\}$ and

$L_h = \{h(a_{j_1} \cdots a_{j_m})\&s^{j_m}\# \cdots \#s^{j_1} \mid m \geq 1; j_1, \ldots, j_m \in I\}$.

(c) Output "YES" or "NO" depending on whether $L_g \cup L_h$ is error-detecting for $R(T)$.

The contradiction arises if we show that $E(g,h)$ is empty if and only if $L_g \cup L_h$ is not error-detecting for $R(T)$. First suppose that $L_g \cup L_h$ is not error-detecting for $R(T)$. Then, there are words w_1, w_2 in $L_g \cup L_h$ such that $w_1 \neq w_2$ and $w_2 \in T(w_1)$. By the construction of T, it follows that w_1 must contain a \$ which is replaced with & and this is the only error that occurs in w_1 to get w_2. Hence, w_1 is of the form $g(a_{i_1} \cdots a_{i_k})\$s^{i_k}\# \cdots \#s^{i_1}$ and w_2 is $g(a_{i_1} \cdots a_{i_k})\&s^{i_k}\# \cdots \#s^{i_1}$. On the other hand, as w_2 contains &, it must also be of the form $h(a_{j_1} \cdots a_{j_m})\&s^{j_m}\# \cdots \#s^{j_1}$. Hence, $g(a_{i_1} \cdots a_{i_k}) = h(a_{j_1} \cdots a_{j_m})$ and $s^{i_k}\# \cdots \#s^{i_1} = s^{j_m}\# \cdots \#s^{j_1}$ which implies that $a_{i_1} \cdots a_{i_k} = a_{j_1} \cdots a_{j_m}$ and, therefore, $a_{j_1} \cdots a_{j_m} \in E(g,h)$. For the converse, suppose z is in $E(g,h)$. Then, there are words $w_1 = g(z)\$u$ in L_g and $w_2 = h(z)\&u$ in L_h, for some $u \in s^+(\#s^+)^*$. As $g(z) = h(z)$, it follows that $w_2 \in T(w_1)$ and, therefore, $L_g \cup L_h$ is not error-detecting for $R(T)$. □

4 Rational Channels Include SID-Channels

Usually, it is not too difficult to construct a transducer for realizing a specific SID-channel. Here we show a general construction method to realize any SID-channel by a transducer. This result demonstrates that the expressive power of transducers as error models is very large. First we outline the basic concepts involved in defining SID-channels – see [8] for details.

Let X be an alphabet containing the symbols a and b. To model the effects of channels on words over X, we consider *error functions* which are applied on words on a symbol by symbol basis. Consider, for instance, the word $x = abab$ and consider a channel that would allow one substitution, one insertion and one deletion in x. As $x = 1a1b1a1b1$, we see that there are four possible positions for a substitution (the four symbols of x), five possible positions for an insertion (the five 1s), and four possible positions for a deletion. Thus, $baaa$ is a possible output from x by inserting a b in front of x, substituting the first b of x with a, and deleting the last

b of x. This effect can be expressed by applying the sequence of *basic error functions* $\mathbf{i}_b, \mathbf{e}, \mathbf{e}, \mathbf{s}, \mathbf{e}, \mathbf{e}, \mathbf{e}, \mathbf{d}, \mathbf{e}$ to each of the nine positions of x, respectively, where $\mathbf{e}$ is the identity function (no error at that position), $\mathbf{i}_b$ is a function that replaces 1 by b (insertion in that position), $\mathbf{d}$ is a function that replaces a symbol in X by 1 (deletion at that position), and $\mathbf{s}$ is any function that maps an $x \in X$ onto a symbol in $X \setminus \{x\}$. Thus, if we consider the error function $\mathbf{h} = \mathbf{i}_b\mathbf{eeseeede}$, then

$$\mathbf{h}(x) = \mathbf{i}_b(1)\mathbf{e}(a)\mathbf{e}(1)\mathbf{s}(b)\mathbf{e}(1)\mathbf{e}(a)\mathbf{e}(1)\mathbf{d}(b)\mathbf{e}(1) = baaa.$$

Generally, when x is a word of length n, an error function $\mathbf{h}$ can be applied on x provided that $|\mathbf{h}| = 2n + 1$. Hence, every error function is of odd length. We use the symbol $\mathcal{H}$ to denote the *set of error functions*. Any subset of $\mathcal{H}$ is called an *SID-language*. If F is a finite non-empty SID-language, we use the symbol ℓ_F for the integer with the property that $2\ell_F + 1$ is the length of a longest error function in F.

The set of error functions is equipped with a product operation, '$\cdot$', such that $(\mathcal{H}, \cdot)$ is a monoid whose neutral element is $\mathbf{e}$. Specifically, if $\mathbf{h}$ and $\mathbf{g}$ are error functions, the product $\mathbf{h} \cdot \mathbf{g}$ is defined as the usual concatenation of words, except at the point where the last symbol of $\mathbf{h}$, say $\mathbf{h}_{2n}$, and the first symbol of $\mathbf{g}$, say $\mathbf{g}_0$, are concatenated; these symbols become one symbol, $\mathbf{c}$, as follows:

$$\mathbf{c} = \begin{cases} \mathbf{h}_{2n}, & \text{if } \mathbf{g}_0 = \mathbf{e}; \\ \mathbf{g}_0, & \text{if } \mathbf{h}_{2n} = \mathbf{e}; \\ \mathbf{i}_{u_1 u_2}, & \text{if } \mathbf{h}_{2n} = \mathbf{i}_{u_1} \text{ and } \mathbf{g}_0 = \mathbf{i}_{u_2}. \end{cases}$$

For example, $(\mathbf{ede}) \cdot (\mathbf{i}_a\mathbf{se}) = \mathbf{edi}_a\mathbf{se}$, $(\mathbf{ede}) \cdot (\mathbf{ese}) = \mathbf{edese}$, and $(\mathbf{edi}_b) \cdot (\mathbf{i}_a\mathbf{se}) = \mathbf{edi}_{ba}\mathbf{se}$.
When $\mathbf{h}$ can be written as $\mathbf{f}_1 \cdot \mathbf{g} \cdot \mathbf{f}_2$, the error function $\mathbf{g}$ is called an *$\mathcal{H$-infix*} of $\mathbf{h}$. Similarly, if $\mathbf{h} = \mathbf{g} \cdot \mathbf{f}$ then $\mathbf{g}$ is an *$\mathcal{H$-prefix*} of $\mathbf{h}$.

Definition 2 *An SID-support is a finite SID-language F such that the following conditions hold for every $\mathbf{h}$ in F:*

1. *If $\mathbf{g}$ is an $\mathcal{H$-infix of $\mathbf{h}$ then $\mathbf{g}$ is in F.*
2. *If $\mathbf{g}$ is of the form $\mathbf{e}^i\mathbf{h}\mathbf{e}^j$, for some non-negative integers i and j, and $|\mathbf{g}| \leq 2\ell_F + 1$ then $\mathbf{g}$ is in F.*

An SID-language Z is said to be of *bounded error effects* if there is an SID-support F such that $\mathbf{h}$ is in Z if and only if $\mathbf{g}$ is in F for every $\mathcal{H}$-infix $\mathbf{g}$ of $\mathbf{h}$ with $|\mathbf{g}| \leq 2\ell_F + 1$. In this case, we write $Z = \lfloor F \rfloor$.

Given an SID-language E, we write $R(E)$ to denote the relation

$$\{(x, y) \mid x \in X^*,\ y = \mathbf{h}(x),\ \text{for some } \mathbf{h} \in E\}.$$

If the language E is of bounded error effects with support F, that is $E = \lfloor F \rfloor$, then $R(\lfloor F \rfloor)$ is a channel. In this case, we also write $\mathrm{chan}(F)$ for $R(\lfloor F \rfloor)$.

Using SID-supports one can define the class of SID-channels which captures the effects of various error combinations including the cases where errors are scattered or they occur in bursts. For example, the channel that permits a burst of up to 4 substitutions in any 30 consecutive input symbols and, possibly at the same time, a total of at most 3 (scattered) insertions and deletions in any 58 consecutive input symbols is an SID-channel. Here we only give the formal definition of the SID-channel $\sigma \odot \delta(m, \ell)$, where m and ℓ are positive integers with $m \leq \ell$, that permits at most m (scattered) substitutions and deletions in every ℓ consecutive input symbols – see [8] for the full class of SID-channels. The channel is equal to $\mathrm{chan}(S_{m,\ell})$, where $S_{m,\ell}$ is the SID-support

$$\{\mathbf{h} \in \mathcal{H}_{\sigma\odot\delta} \mid |\mathbf{h}| \leq 2\ell + 1,\ \nu(\mathbf{h}) \leq m\}.$$

Here $\mathcal{H}_{\sigma\odot\delta}$ is the set of all error functions containing only symbols $\mathbf{e}$, $\mathbf{d}$, or $\mathbf{s}$ (for any possible substitution function $\mathbf{s}$), and $\nu(\mathbf{h})$ is the number of substitution and deletion symbols that occur in $\mathbf{h}$.

Proposition 4 *For every SID-support F there effectively exists a transducer T_F such that $\mathrm{chan}(F) = R(T_F)$. Hence, every SID-channel is a rational channel. On the other hand, there is a rational channel γ for which no SID-support F exists with $\gamma = \mathrm{chan}(F)$.*

Proof. The transducer T_F is equal to $(X, X, Q, \alpha, \{\omega\}, \Delta_F)$ such that $Q = \{\alpha, \omega\} \cup \hat{F}$, where $\hat{F} = \{\mathbf{h} \in F \mid |\mathbf{h}| = 2\ell_F + 1\}$, and Δ_F consists of the following transitions, for $\mathbf{g}, \mathbf{h} \in \hat{F}$ and $x, x' \in X^*$:

- $\alpha 1/1\mathbf{g}$
- $\mathbf{g}x/x'\mathbf{h}$, where $|x| = \ell_F$ and $\mathbf{g} \cdot \mathbf{h} \in \lfloor F \rfloor$ and $x' = \mathbf{g}(x)$
- $\mathbf{h}x/x'\omega$, where $|x| \leq \ell_F$ and $x' = \mathbf{f}(x)$ for some $\mathcal{H}$-prefix $\mathbf{f}$ of $\mathbf{h}$.

Intuitively, if T_F is at state $\mathbf{g}$ then the channel will apply the error function $\mathbf{g}$ on the next ℓ_F input symbols, or a prefix of $\mathbf{g}$ on the last block of the input. For the correctness of the construction, first assume $(x, y) \in$

$R(T_F)$. Then, there is a computation $\alpha 1/1\mathbf{h}_1 x_1/x'_1 \cdots \mathbf{h}_n x_n/x'_n\omega$ of T_F, with n positive integer and $x = x_1 \cdots x_n$ and $y = x'_1 \cdots x'_n$, such that for $i < n$ one has $\mathbf{h}_i \cdot \mathbf{h}_{i+1} \in \lfloor F \rfloor$ and $|x_i| = \ell_F$ and $x'_i = \mathbf{h}_i(x_i)$. Also, $x'_n = \mathbf{f}(x_n)$ for some $\mathcal{H}$-prefix $\mathbf{f}$ of $\mathbf{h}_n$. Then $\mathbf{g} \in \lfloor F \rfloor$, where $\mathbf{g} = \mathbf{h}_1 \cdot \ldots \cdot \mathbf{h}_{n-1} \cdot \mathbf{f}$. As $y = \mathbf{h}_1(x_1) \cdots \mathbf{h}_{n-1}(x_{n-1})\mathbf{f}(x_n) = \mathbf{g}(x)$, it follows that $(x, y) \in \mathrm{chan}(F)$ as required.

For the converse, assume $(x, y) \in \mathrm{chan}(F)$. Then, there is an error function $\mathbf{h} \in \lfloor F \rfloor$ such that $y = \mathbf{h}(x)$. Moreover, there is a positive integer n such that $x = x_1 \cdots x_n$ with $x_i \in X^{\ell_F}$ for $i < n$ and $x_n \in X^*$ is of length at most ℓ_F. Then, there are error functions $\mathbf{h}_1, \ldots, \mathbf{h}_n$ such that $\mathbf{h} = \mathbf{h}_1 \cdot \ldots \cdot \mathbf{h}_n$ and $\mathbf{h}(x) = \mathbf{h}_1(x_1) \cdots \mathbf{h}_n(x_n)$ – see [7]. If $|\mathbf{h}_n| < 2\ell_F + 1$ then there is an error function $\mathbf{g}$ in $\hat{F}$ of the form $\mathbf{h}_n \cdot \mathbf{e}^+$. Also, if $n > 1$ then $\mathbf{h}_{n-1} \cdot \mathbf{g}$ is in $\lfloor F \rfloor$. Now let $y_i = \mathbf{h}_i(x_i)$; then,

$$\alpha 1/1\mathbf{h}_1 x_1/y_1 \cdots \mathbf{h}_{n-1} x_{n-1}/y_{n-1}\mathbf{g}x_n/y_n\omega$$

is a computation of T_F and, therefore, $(x, y) \in R(T_F)$ as required.

Now consider the channel $\gamma = \{(w, w), (wx, w) \mid w \in X^*, x \in X\}$ which is realized by the transducer $(X, X, \{q_0, q_1\}, q_0, \{q_0, q_1\}, \Delta)$ with transitions $\Delta = \{q_0 x/xq_0 \mid x \in X\} \cup \{q_0 x/1q_1\}$. Assume $\gamma = \mathrm{chan}(F)$ for some SID-support F and consider two different symbols $x_1, x_2 \in X$ and any word $w \in X^*$. As $(wx_1, w) \in \gamma$, there is an error function $\mathbf{h} \in \lfloor F \rfloor$ such that $\mathbf{h}(wx_1) = w$. Then also $\mathbf{h} \cdot \mathbf{eee} \in \lfloor F \rfloor$ and, as $(\mathbf{h} \cdot \mathbf{eee})(wx_1x_2) = wx_2$, one has $(wx_1x_2, wx_2) \in \gamma$. This is impossible, however, as $x_1 \neq x_2$. □

5 Some Remarks on Unique Decodability and Error-Correction

A language L is called *error-correcting for* a channel γ if $\langle w_1 \rangle_\gamma \cap \langle w_2 \rangle_\gamma \neq \emptyset$ implies $w_1 = w_2$, for all $w_1, w_2 \in L \cup \{1\}$. According to this definition, no two different words of $L \cup \{1\}$ can result into the same output via the channel γ. Thus, if z is received with errors then there is a unique w in $L \cup \{1\}$ which resulted in z. Hence, in principle, z can be decoded as w correcting thus the errors caused by γ. The use of $L \cup \{1\}$ as opposed to L is necessary as in the definition of error-detection. It is easy to verify that a language which is error-correcting for γ is also error-detecting for γ.

Now if K is a code and the language K^* is error-correcting for a channel γ then we say that K is *uniquely decodable for* γ, or that K is

$(\gamma, *)$-*correcting* – see [5]. From [7] we recall that the following problem is decidable:[2]

Input: An SID-support F and a finite code K.

Output: "YES" or "NO" depending on whether K is uniquely decodable for $\mathrm{chan}(F)$.

A question that arises now is whether the following analogue of the problem in Theorem 1 is decidable as well.

Input: A trim channel transducer T in standard form and a trim deterministic finite automaton A with the same input alphabets.

Output: "YES" or "NO" depending on whether $L(A)$ is error-correcting for $R(T)$.

An affirmative answer would imply that the following problem is decidable as well:

Input: A trim channel transducer T in standard form and a trim deterministic finite automaton A accepting a code.

Output: "YES" or "NO" depending on whether $L(A)$ is uniquely decodable for $R(T)$.

We note that in [3] it is shown that the problem of whether a given regular language L is 1-error correctable is decidable. In our terminology, 1-error-correctable means error-correcting for the channel that allows at most one substitution error in any input word. A transducer realizing this channel is the following:

$T_1 = (X, X, \{q_0, q_1\}, q_0, \{q_0, q_1\}, \Delta_1)$,

where $\Delta_1 = \{qx/xq \mid x \in X, q \in \{q_0, q_1\}\} \cup \{q_0x/x'q_1 \mid x, x' \in X, x \neq x'\}$. Furthermore in [3] it is shown that the problem of whether a given context-free language is 1-error correctable is undecidable. Hence, the following analogue of Proposition 3 obtains.

Proposition 5 *The following problem is undecidable.*

Input: A channel transducer T and a context-free grammar G.

Output: "YES" or "NO" depending on whether the language $L(G)$ is error-correcting for the channel $R(T)$.

□

Acknowledgment The author is indebted to Professor J. Sakarovitch for his advice on transducers during the 'Half Century of Automata Theory' event, University of Western Ontario, July 2000.

[2] We note that in [7] the term 'error-correcting for γ' is the same as 'uniquely decodable for γ'.

References

[1] M-P. Béal, O. Carton, C. Prieur, J. Sakarovitch: Squaring transducers: an efficient procedure for deciding functionality and sequentiality of transducers. In G. Gonnet, D. Panario, A. Viola (eds.) *LATIN 2000. Lecture Notes in Computer Science 1776,* pp 397–406, 2000.

[2] J. Berstel: *Transductions and Context-Free Languages,* B. G. Teubner, Stuttgart, 1979.

[3] J. Dassow, V. Mitrana, G. Păun: Point mutations in context-free languages. In S. Bozapalidis (ed.), *Proceedings of the 3rd International Conference, Developments in Language Theory,* pp 429–446, 1997.

[4] T. Harju, J. Karhumäki: Morphisms. In G. Rozenberg and A. Salomaa (eds.) *Handbook of Formal Languages,* vol I, pp 439–510, 1997.

[5] H. Jürgensen, S. Konstantinidis: Codes. In G. Rozenberg and A. Salomaa (eds.) *Handbook of Formal Languages,* vol I, pp 511–607, 1997.

[6] K. Salomaa, S. Yu, Q. Zhuang: The state complexities of some basic operations on regular languages. *Theoretical Computer Science 125,* pp 315–328, 1994.

[7] S. Konstantinidis: Structural analysis of error-correcting codes for discrete channels that involve combinations of three basic error types. *IEEE Transactions on Information Theory 45,* pp 60–77, 1999.

[8] S. Konstantinidis: An algebra of discrete channels that involve combinations of three basic error types. *Information and Computation,* to appear.

[9] S. Konstantinidis: Error-detecting properties of languages. In M. Ito (ed.) *Third International Colloquium on Words, Languages and Combinatorics, 2000* to appear.

[10] M. P. Schützenberger: Sur les relations rationnelles. In H. Brakhage (ed.) *Automata Theory and Formal Languages. Lecture Notes in Computer Science 33,* pp 209–213, 1975.

[11] S. Yu: Regular Languages. In G. Rozenberg and A. Salomaa (eds.) *Handbook of Formal Languages,* vol I, pp 41–110, 1997.