Journal of Universal Computer Science, vol. 8, no. 2 (2002), 278-291
submitted: 15/9/01, accepted: 31/1/02, appeared: 28/2/02 © *J.UCS*

Transducers and the Properties of Error-Detection, Error-Correction, and Finite-Delay Decodability[1]

Stavros Konstantinidis
(Dept of Mathematics and Computing Science, Saint Mary's University, Canada[2]
Email: s.konstantinidis@stmarys.ca)

Χρόνια πολλά κύριε Γιούργκενσεν. Εύχομαι να τα εκατοστήσετε.

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. On the other hand, the property of error-correction ensures that the channel cannot transform two different words of the language to the same word. In this work we use transducers to model noisy channels and consider a few simple transducer operations that can be used to reduce the language properties of error-detection and error-correction to the transducer property of functionality. As a consequence, we obtain simple polynomial-time algorithms for deciding these properties for regular languages. On the other hand the properties are not decidable for context-free languages. In addition we show that, in a certain sense, the class of rational channels can be used to model various error combinations. Using the same tools, we also obtain simple polynomial-time algorithms for deciding whether a given regular language is thin and whether a given regular code has decoding delay d, for given d, and for computing the minimum decoding delay of a given regular code.

Key Words: channel, decidability, decoding delay, error-correction, error-detection, regular language, transducer, unique decodability.
Category: F

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 using the errors permitted by the channel. This property is basic as it allows one to determine whether or not the received information is correct. On the other hand, if the language is error-correcting *for* the given channel then no two different words of the language can be transformed to the same word. 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

[1] C. S. Calude, K. Salomaa, S. Yu (eds.). *Advances and Trends in Automata and Formal Languages. A Collection of Papers in Honour of the 60th Birthday of Helmut Jürgensen.*

[2] Research partially supported by Grant OGP220259 of the Natural Sciences and Engineering Research Council of Canada

in the transmitted/stored words – see [Jürgensen and Konstantinidis 1996] and [Jürgensen and Konstantinidis 1997] for details on channels. In this work we use transducers to model noisy channels and consider a few simple transducer operations that can be used to reduce the language properties of error-detection and error-correction to the transducer property of functionality. Using the tools considered here, we also obtain simple polynomial-time algorithms for deciding whether a given regular language is thin and whether a given regular code has decoding delay d, for given nonnegative integer d, and for computing the minimum decoding delay of a given regular code.

It turns out that the idea of using transducer functionality to decide code related properties is not new. It has been considered in [Head and Weber 1993] where a remarkably simple and efficient algorithm is given for deciding whether a given regular language is a (uniquely decodable) code. We also note that this decidability question is answered in [McCloskey 1996] with the same time complexity.

The paper is organized as follows. In the next section we give the basic notation about words, languages, automata, relations and channels. In Section 3, we consider certain simple transducer operations and discuss the complexity of these operations. Moreover, we give a simple algorithm for deciding whether a given regular language is thin. In Section 4, we discuss the decidability of the error-detection property and in Section 5 the decidability of the properties of error-correction and unique decodability in the presence of errors. The property of decodability with finite delay for regular codes is considered in Section 6. Finally, in Section 7 we examine the expressive power of transducers as error models by showing that every SID channel is a rational channel.

2 Basic Notions and Notation

For a set S, we denote by $|S|$ the cardinality of S. 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 λ, whereas $V^+ = V^* \setminus \{\lambda\}$. We use the notation $|v|$ for the length of the word v. A language is any set of words. A non-empty language C is called a *uniquely decodable code* or simply a *code* if, for all positive integers m and n and for all words $v_1, \ldots, v_n, u_1, \ldots, u_m \in C$, the equation $v_1 v_2 \cdots v_n = u_1 u_2 \cdots u_m$ implies that $m = n$ and $u_i = v_i$ for every $i = 1, \ldots, n$. In the sequel we shall use the symbols X and Y to denote two finite alphabets. The reader is referred to [Rozenberg and Salomaa 1997] for information on formal languages in general, and to [Berstel and Perrin 1985] and [Jürgensen and Konstantinidis 1997] for information on codes.

Binary relations and channels

A binary relation ρ (over X and Y) is a subset of $X^* \times Y^*$. The relation is called *functional* if $(x, y_1) \in \rho$ and $(x, y_2) \in \rho$ imply that $y_1 = y_2$. The inverse of ρ, denoted by ρ^{-1}, is the relation $\{(y, x) \mid (x, y) \in \rho\}$ over Y and X. The concatenation $\rho_1 \cdot \rho_2$ of two relations ρ_1 and ρ_2 is the relation $\{(x_1 x_2, y_1 y_2) \mid (x_1, y_1) \in \rho_1, (x_2, y_2) \in \rho_2\}$. If L is a language then $\rho \downarrow L$ is the relation $\rho \cap (L \times Y^*) = \{(x, y) \in \rho \mid x \in L\}$ and $\rho \uparrow L$ is the relation $\rho \cap (X^* \times L) = \{(x, y) \in \rho \mid y \in L\}$. Obviously, $\rho \uparrow L = (\rho^{-1} \downarrow L)^{-1}$. Also, we define $\rho(L) = \{y \mid (x, y) \in \rho$ for some $x \in L\}$. The *diagonal* of the language L is the relation $\mathrm{D}_L = \{(x, x) \mid x \in L\}$.

In this paper, a *channel (over X)* is a binary relation $\gamma \subseteq X^* \times X^*$ that is domain preserving; that is, $(x,x) \in \gamma$ for all $x \in \{z \mid (z,y) \in \gamma$ for some word $y\}$. When (x,y) is in the channel γ we say that y can be received from x, the input word, through γ. In case $x \neq y$, we say that y can be received from x with errors. The fact that γ is domain preserving means that error-free communication is possible for input words. The channel γ is called a *rational channel,* if it is realized by a finite transducer. In this case, the transducer is called a *channel transducer* – see next section for transducers.

Automata

A *finite automaton with empty transitions* (a λ-*NFA* for short) is a quintuple $\mathcal{A} = (X, Q, q_0, Q_+, E)$ such that X is the input alphabet, the pair (Q, E) is a labeled directed graph with set of vertices Q, called the set of states, and set of labeled edges E, called the set of transitions, q_0 is the initial state in Q, and Q_+ is a subset of Q, called the set of final states. The transitions in E are words of the form qxq' such that $q, q' \in Q$ and $x \in X \cup \{\lambda\}$. In this paper we assume that X and Q are always disjoint. A *computation of* $\mathcal{A}$ is a word ω of the form $p_0x_1p_1 \cdots x_np_n$ such that each factor $p_{i-1}x_ip_i$ of ω is in E. The language accepted by the automaton $\mathcal{A}$ is denoted by $L(\mathcal{A})$. A word w is in $L(\mathcal{A})$ if and only if there is a computation that starts with the initial state q_0 and ends with a final state in Q_+ such that w is equal to the concatenation of the X^*-words appearing in the computation.

The automaton is said to *have accessible states* if, for every state q in Q, there is a path from q_0 to q. The automaton is called *trim* if it has accessible states and, for every state q, there is a path from q to a final state in Q_+ (when Q_+ is not empty). The size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, is equal to $|Q| + |E|$. If $\mathcal{A}$ has accessible states then $|Q| \leq |E| + 1$ and, therefore, $|\mathcal{A}| = \Theta(|E|)$. The automaton $\mathcal{A}$ is an *NFA* if, for every transition qxq', x is not empty. It is called *deterministic* if it is an NFA and, for every transitions qxq' and qxq'', $q' = q''$.

For a word x in X^* we write $\mathcal{A}_x$ for a trim λ-NFA of size $O(1+|x|)$ such that $L(\mathcal{A}_x) = \{x\}$. The symbol $\mathcal{A}_{\text{all}}$ denotes a trim λ-NFA of size $\Theta(1)$ such that $L(\mathcal{A}_{\text{all}}) = X^*$. Given two λ-NFAs $\mathcal{A}$ and $\mathcal{A}'$ one can construct the following – see [Yu 1997].

- $\mathcal{A} \cup \mathcal{A}'$: a λ-NFA of size $O(|\mathcal{A}| + |\mathcal{A}'|)$ such that $L(\mathcal{A} \cup \mathcal{A}') = L(\mathcal{A}) \cup L(\mathcal{A}')$.
- $\mathcal{A} \cdot \mathcal{A}'$: a λ-NFA of size $O(|\mathcal{A}| + |\mathcal{A}'|)$ such that $L(\mathcal{A} \cdot \mathcal{A}') = L(\mathcal{A})L(\mathcal{A}')$.
- $\mathcal{A}^*$: a λ-NFA of size $O(|\mathcal{A}|)$ such that $L(\mathcal{A}^*) = L(\mathcal{A})^*$.
- $\mathcal{A}^d$, for some $d \geq 0$: a λ-NFA of size $O(1 + d|\mathcal{A}|)$ such that $L(\mathcal{A}^d) = L(\mathcal{A})^d$.

If $\mathcal{A}$ and $\mathcal{A}'$ are trim then also the constructed automata are trim.

3 Transducers

A *(finite) transducer* $\mathcal{T}$ is a sextuple (X, Y, Q, q_0, Q_+, E) such that Y is the output alphabet and the components X, Q, q_0, Q_+, and E are as in the definition of λ-NFAs with the following difference: the transitions in E are of the form qx/yq' with $q, q' \in Q$, $x \in X^*$, and $y \in Y^*$. As before, the sets Q and $X \cup Y$ are assumed to be disjoint. A computation of $\mathcal{T}$ is a word ξ of the form $p_0x_1/y_1p_1 \cdots x_n/y_np_n$ such that each factor $p_{i-1}x_i/y_ip_i$ of ξ is in E. The *relation realized by* $\mathcal{T}$ is denoted by $R(\mathcal{T})$. A pair of words (w, z) is in $R(\mathcal{T})$ if and only if there is a computation that starts with q_0 and ends with a final state in Q_+ such that w is equal to the concatenation of the X^*-words and z of the Y^*-words that appear in the computation.

The transducer is called *trim* if, for every state $q \in Q$ there is a path from q_0 to q and a path from q to a final state (when $Q_+ \neq \emptyset$).

The *size*, $|\mathcal{T}|$, of the transducer $\mathcal{T}$ is equal to $|Q| + \sum_{qx/yq' \in E}(|x/y|)$. If $\mathcal{T}$ is trim then $|\mathcal{T}| = \Theta(\sum_{qx/yq' \in E}(|x/y|))$. The transducer is in *standard form* if, for every transition qx/yq', one has that $x \in X \cup \{\lambda\}$ and $y \in Y \cup \{\lambda\}$. Note that the size of a trim transducer in standard form is $\Theta(|E|)$. We refer the reader to [Berstel 1979] for more information on transducers.

The transducer $\mathcal{T}$ is called *functional* if the relation $R(\mathcal{T})$ is functional. In [Schützenberger 1975] it is shown that the problem of whether a given finite transducer is functional is decidable. This result has been improved in terms of the efficiency of the decision procedure in [Gurari and Ibarra 1983] and [Béal et al. 2000] – see also [Head and Weber 1993]. If the given transducer $\mathcal{T}$ is real-time then deciding functionality requires time $O(|\mathcal{T}|^2)$ [Béal et al. 2000]. In [Head and Weber 1993] it is stated that the problem is decidable in time $O(|\mathcal{T}|^2 \log |\mathcal{T}|)$ when $\mathcal{T}$ is in standard form. For the case where $\mathcal{T}$ is a (nondeterministic) sequential machine, the decision procedure is simpler and requires time $O(|\mathcal{T}|^2)$ [Head and Weber 1993]. We summarize the above results as follows (the alphabet size is assumed to be fixed).

Proposition 1 *The problem of whether a given transducer $\mathcal{T}$ is functional is decidable in time $O(\mathrm{tf}(|\mathcal{T}|))$, for some polynomially bounded function* tf.

Given two transducers $\mathcal{T}$ and $\mathcal{T}'$, and two λ-NFAs $\mathcal{A}$ and $\mathcal{A}'$, one can construct the following:

• A transducer $\mathcal{T}^{-1}$ of size $O(|\mathcal{T}|)$ such that $R(\mathcal{T}^{-1}) = R(\mathcal{T})^{-1}$ – see [Yu 1997], for instance.

• A transducer $\mathcal{T} \cdot \mathcal{T}'$ of size $O(|\mathcal{T}| + |\mathcal{T}'|)$ such that $R(\mathcal{T} \cdot \mathcal{T}') = R(\mathcal{T}) \cdot R(\mathcal{T}')$. The construction of $\mathcal{T} \cdot \mathcal{T}'$ is analogous to that of the automaton $\mathcal{A} \cdot \mathcal{A}'$.

• A λ-NFA $\mathcal{A}_\mathcal{T}$ such that $L(\mathcal{A}_\mathcal{T}) = (R(\mathcal{T}))(L(\mathcal{A}))$ – see below for details on the size of $\mathcal{A}_\mathcal{T}$.

• A transducer $\mathcal{D}_\mathcal{A}$ of size $O(|\mathcal{A}|)$ such that $R(\mathcal{D}_\mathcal{A}) = \mathrm{D}_{L(\mathcal{A})}$. The transducer $\mathcal{D}_\mathcal{A}$ obtains from $\mathcal{A}$ by replacing each transition qxq' of $\mathcal{A}$ with the transition qx/xq'.

• A transducer $\mathcal{A} \times \mathcal{A}'$ of size $O(|\mathcal{A}| + |\mathcal{A}'|)$ such that $R(\mathcal{A} \times \mathcal{A}') = L(\mathcal{A}) \times L(\mathcal{A}')$. The transducer $\mathcal{A} \times \mathcal{A}'$ obtains from $\mathcal{A}$ and $\mathcal{A}'$ by replacing each transition qxq' of $\mathcal{A}$ with $qx/\lambda q'$, and each transition pyp' of the automaton $\mathcal{A}'$ with $p\lambda/yp'$, and then 'connecting' each final state, say f, of $\mathcal{A}$ with the initial state, say q_0', of $\mathcal{A}'$ using the transition $f\lambda/\lambda q_0'$.

In the above, if the given λ-NFAs are trim and the given transducers are trim and in standard form, then also the constructed automata are trim and the constructed transducers are trim and in standard form.

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

Input: a trim λ-NFA $\mathcal{A}$ and a trim transducer $\mathcal{T}$ in standard form.

Output: a trim transducer, denoted $\mathcal{T} \downarrow \mathcal{A}$ (respectively, $\mathcal{T} \uparrow \mathcal{A}$), in standard form, such that $R(\mathcal{T} \downarrow \mathcal{A}) = R(\mathcal{T}) \downarrow L(\mathcal{A})$ (respectively, $R(\mathcal{T} \uparrow \mathcal{A}) = R(\mathcal{T}) \uparrow L(\mathcal{A})$).

Proof. See the appendix. □

We note that the object $\mathcal{T} \downarrow \mathcal{A}$ 'contains' more information than $\mathcal{A}_\mathcal{T}$ does in the sense that $\mathcal{A}_\mathcal{T}$ can be constructed from $\mathcal{T} \downarrow \mathcal{A}$ by simply removing the input part from each of the transitions of $\mathcal{T} \downarrow \mathcal{A}$. Note that a construction of $\mathcal{A}_\mathcal{T}$ is given in [Yu 1997] where the automaton $\mathcal{A}$ is assumed to be deterministic and, therefore, there the proof of correctness is less complex. In this paper, the fact that $\mathcal{A}$ is nondeterministic is essential in the sections 4 and 5.

Corollary 1. *The following problem is computable in time $O(|\mathcal{A}||\mathcal{T}|)$.*

Input: a trim λ-NFA $\mathcal{A}$ and a trim transducer $\mathcal{T}$ in standard form.
Output: a λ-NFA, denoted by $\mathcal{A}_\mathcal{T}$, such that $L(\mathcal{A}_\mathcal{T}) = (R(\mathcal{T}))(L(\mathcal{A}))$. □

A language L is called *thin* [Yu 1997] if L contains no two different words of the same length. Let b be any alphabet symbol and let $\beta_L = \{(w, b^{|w|}) \mid w \in L\}$. The next result follows easily from the definition of functional relation.

Proposition 3 *A language L is thin if and only if the relation β_L^{-1} is functional.*

Corollary 2. *The following problem is decidable in time $O(\mathrm{tf}(|\mathcal{A}|))$.*

Input: a trim λ-NFA $\mathcal{A}$.
Output: YES/NO, depending on whether the language $L(\mathcal{A})$ is thin.

Proof. One constructs a trim transducer $\mathcal{T}$ in standard form, of size $O(|\mathcal{A}|)$, such that $px/b^{|x|}p'$ is a transition of $\mathcal{T}$ if and only if pxp' is a transition of $\mathcal{A}$. As $R(\mathcal{T}) = \beta_{L(\mathcal{A})}$, the claim follows from the above proposition. □

We note that, if the given automaton $\mathcal{A}$ is an NFA, then the transducer constructed in the above proof is a (nondeterministic) sequential machine; in this case, the problem is decidable in time $O(|\mathcal{A}|^2)$.

4 Error-Detection

Let γ be a channel. A language L is *error-detecting for* γ, [Konstantinidis and O'Hearn 2002], if $(w_1, w_2) \in \gamma$ implies $w_1 = w_2$ for all words $w_1, w_2 \in L_\lambda$, where $L_\lambda = L \cup \{\lambda\}$. Assuming that only words from L_λ are sent into the channel, if w_2 is retrieved from the channel and w_2 is in L_λ then w_2 must be correct; that is, equal to the word that was sent into the channel. On the other hand, if the word retrieved from the channel is not in L_λ then an error is detected. The use of L_λ as opposed to L ensures that λ cannot be received from a word in $L \setminus \{\lambda\}$ and that no word in $L \setminus \{\lambda\}$ can be received from λ. 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 [Jürgensen and Konstantinidis 1997].

Proposition 4 *Let γ be a channel. A language L is error-detecting for γ if and only if the relation $\gamma \downarrow L_\lambda \uparrow L_\lambda$ is functional.*

Proof. First note that $\gamma \downarrow L_\lambda \uparrow L_\lambda$ is equal to $\{(w_1, w_2) \in \gamma \mid w_1 \in L_\lambda, w_2 \in L_\lambda\}$. For the 'if' part, consider a pair $(w_1, w_2) \in \gamma$ with $w_1, w_2 \in L_\lambda$. As γ is domain preserving, also $(w_1, w_1) \in \gamma$. As $\gamma \downarrow L_\lambda \uparrow L_\lambda$ is functional, w_1 must be equal to w_2. Hence, L is error-detecting for γ. For the converse, consider two pairs (z, w_1) and (z, w_2) in $\gamma \downarrow L_\lambda \uparrow L_\lambda$. As L is error-detecting for γ, $z = w_1$ and $z = w_2$ which implies that $w_1 = w_2$. Hence, $\gamma \downarrow L_\lambda \uparrow L_\lambda$ is functional. □

Corollary 3. *The following problem is decidable in time $O(\mathrm{tf}(|\mathcal{T}||\mathcal{A}|^2))$.*

Input: a trim λ-NFA $\mathcal{A}$ and a trim channel transducer $\mathcal{T}$ in standard form.
Output: YES/NO, depending on whether the language $L(\mathcal{A})$ is error-detecting for the channel $R(\mathcal{T})$.

Proof. The algorithm consists of, first, constructing the trim transducer $(\mathcal{T} \downarrow (\mathcal{A} \cup \mathcal{A}_\lambda)) \uparrow (\mathcal{A} \cup \mathcal{A}_\lambda)$ realizing the relation $R(\mathcal{T}) \downarrow L(\mathcal{A})_\lambda \uparrow L(\mathcal{A})_\lambda$, and then testing whether the transducer is functional. The correctness and complexity of the algorithm follow from the above proposition and Propositions 1 and 2. □

When K is a regular code then K^* is a regular language. Also, as the λ-NFA $\mathcal{A}^*$ can be constructed from $\mathcal{A}$ in time $O(|\mathcal{A}|)$, the following obtains easily from the above.

Corollary 4. *The following problem is decidable in time $O(\mathrm{tf}(|\mathcal{T}||\mathcal{A}|^2))$.*

Input: A trim channel transducer $\mathcal{T}$ in standard form and a trim λ-NFA $\mathcal{A}$ accepting a code.
Output: YES/NO depending on whether the code $L(\mathcal{A})$ is $\big(R(\mathcal{T}), *\big)$-detecting. □

The fact that Proposition 2 works for λ-NFAs allows the polynomial-time complexity in the above result. If that proposition involved a deterministic finite automaton $\mathcal{A}$, then we would need to construct a deterministic $\mathcal{A}^*$ from $\mathcal{A}$ which, in general, requires an exponential number of states – see [Salomaa et al. 1994].

Proposition 5 *The following problem is undecidable.*

Input: A channel transducer $\mathcal{T}$ and a context-free grammar G.
Output: YES/NO depending on whether the language $L(G)$ is error-detecting for the channel $R(\mathcal{T})$.

Proof. We assume that 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 [Harju and Karhumäki 1997]: 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 $\mathcal{T}$ 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, $\mathcal{T}$ 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/NO depending on whether $L_g \cup L_h$ is not error-detecting for $R(\mathcal{T})$.

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 $R(\mathcal{T})$. First suppose that $L_g \cup L_h$ is not error-detecting for $R(\mathcal{T})$. Then, there are words w_1, w_2 in $L_g \cup L_h$ such that $w_1 \neq w_2$ and $(w_1, w_2) \in R(\mathcal{T})$. By the construction of $\mathcal{T}$, 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 $(w_1, w_2) \in R(\mathcal{T})$ and, therefore, $L_g \cup L_h$ is not error-detecting for $R(\mathcal{T})$. □

5 Error-Correction and Unique Decodability (noisy case)

Let γ be a channel. A language L is *error-correcting for* the channel γ, if $(w_1, z), (w_2, z) \in \gamma$ implies that $w_1 = w_2$ for all words $z \in X^*$ and $w_1, w_2 \in L_\lambda$, where $L_\lambda = L \cup \{\lambda\}$. Assuming that only words from L_λ are sent into the channel, if z is retrieved from the channel then there is exactly one word from L_λ that have resulted in z. Therefore, even if z has been received with errors then, in principle, one can find the word $w \in L_\lambda$ with $(w, z) \in \gamma$, correcting thus the errors in w. In the definition, the use of L_λ ensures that the empty word and a nonempty word of L can never result in the same output through the channel γ. It should be clear that if a language is error-correcting for γ then it is also error-detecting for γ, assuming that the language is included in the domain of the channel γ.

Proposition 6 *Let γ be a channel. A language L is error-correcting for γ if and only if the relation $\gamma^{-1} \uparrow L_\lambda$ is functional.*

Proof. The claim follows by the definitions of error-correcting language and relation functionality when we note that $\gamma^{-1} \uparrow L_\lambda = \{(z, w) \mid (w, z) \in \gamma$ and $w \in L_\lambda\}$. □

Corollary 5. *The following problem is decidable in time $O(\mathrm{tf}(|\mathcal{T}||\mathcal{A}|))$.*

Input: a trim λ-NFA $\mathcal{A}$ and a trim channel transducer $\mathcal{T}$ in standard form.
Output: YES/NO, depending on whether the language $L(\mathcal{A})$ is error-correcting for the channel $R(\mathcal{T})$.

Proof. The statement follows by Propositions 1 and 2, and by the above proposition when we note that the relation $R(\mathcal{T})^{-1} \uparrow L(\mathcal{A})_\lambda$ is realized by the trim transducer $\mathcal{T}^{-1} \uparrow (\mathcal{A} \cup \mathcal{A}_\lambda)$, which can be constructed in time $O(|\mathcal{T}||\mathcal{A}|)$. □

As a special case of the above result, consider the transducer $\mathcal{T}_1 = (X, X, \{q_0, q_1\}, q_0, \{q_0, q_1\}, E_1)$, where $E_1 = \{qx/xq \mid x \in X, q \in \{q_0, q_1\}\} \cup \{q_0 x/x' q_1 \mid x, x' \in X, x \neq x'\}$. Clearly, the transducer realizes the channel that allows up to 1 substitution error in any input word. Therefore, the problem of whether a given regular language is 1-error-correctable, which is considered in [Dassow et al. 1997], can be decided efficiently using transducer functionality. An important consequence of Proposition 6 is that unique decodability in the presence of errors can also be decided efficiently. A *code* K is *uniquely decodable for* the channel γ (or $(\gamma, *)$-correcting [Jürgensen and Konstantinidis 1997]) if the language K^* is error-correcting for γ. As one can construct the automaton $\mathcal{A}^*$ from the automaton $\mathcal{A}$ in time $O(|\mathcal{A}|)$, the following obtains easily.

Corollary 6. *The following problem is decidable in time* $O(\mathrm{tf}(|\mathcal{T}||\mathcal{A}|))$.

Input: a trim λ-NFA $\mathcal{A}$ accepting a code and a trim channel transducer $\mathcal{T}$ in standard form.
Output: YES/NO, depending on whether the code $L(\mathcal{A})$ is uniquely decodable for the channel $R(\mathcal{T})$. □

We note that the question of deciding whether a given *finite code* is uniquely decodable for *certain* channels has also been considered in [Hartnett 1968], [Sato 1979], and [Konstantinidis 1999].

In [Dassow et al. 1997] 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 5 obtains.

Proposition 7 *The following problem is undecidable.*

Input: A channel transducer $\mathcal{T}$ and a context-free grammar G.
Output: YES/NO depending on whether the language $L(G)$ is error-correcting for the channel $R(\mathcal{T})$. □

6 Unique Decodability with Finite Delay (noiseless case)

A code K is said to have finite decoding (also called deciphering) delay, if there is a nonnegative integer d such that, for every $v_1, v_2 \in K$ and for every $z \in K^d X^*$, $v_1 z \in v_2 K^*$ implies $v_1 = v_2$ [Berstel and Perrin 1985]. In this case, we say that K has decoding delay d. The problem of deciding whether a given regular code has finite decoding delay has been shown to be decidable in [Devolder et al. 1994]. Here, we consider a different version of the problem: decide whether the code K has decoding delay d, for given regular code K and given nonnegative integer d.

Proposition 8 *Let K be a code and let d be a nonnegative integer. The code K has decoding delay d if and only if the relation* $\mathrm{D}_K \cdot (K^d X^* \times \{\lambda\})$ *is functional.*

Proof. First note that $\mathrm{D}_K \cdot (K^d X^* \times \{\lambda\})$ is equal to $\{(vz, v) \mid v \in K, z \in K^d X^*\}$. Thus, it is sufficient to show that K has decoding delay d if and only if $v_1 K^d X^* \cap v_2 K^d X^* \neq \emptyset$ implies $v_1 = v_2$ for all v_1 and v_2 in K.

Assume that K has decoding delay d and consider words $v_1, v_2 \in K$, $u_1, u_2 \in K^d$, and $y_1, y_2 \in X^*$ such that $v_1 u_1 y_1 = v_2 u_2 y_2$. Without loss of generality, suppose that v_1u_1 is no longer than v_2u_2; then $v_2u_2 = v_1u_1y_1'$, for some prefix y_1' of y_1. As $u_1y_1' \in K^dX^*$ and $v_1(u_1y_1') \in v_2K^*$, it follows that $v_1 = v_2$ as required. For the converse, consider words $v_1, v_2 \in K$ and $z \in K^dX^*$ such that $v_1z \in v_2K^*$. Then, as $v_1zv_1^d \in v_2K^dX^*$, one has that $v_1 = v_2$, as required. □

Corollary 7. *The following problem is decidable in time* $O(\mathrm{tf}((1+d)|\mathcal{A}|))$.
Input: a nonnegative integer d *and a trim* λ*-NFA* $\mathcal{A}$ *accepting a code.*
Output: YES/NO, depending on whether $L(\mathcal{A})$ *has decoding delay* d.

Proof. The statement follows from the above proposition when we note that the relation $\mathrm{D}_{L(\mathcal{A})} \cdot (L(\mathcal{A})^dX^* \times \{\lambda\})$ can be realized by the trim transducer $\mathcal{D}_{\mathcal{A}} \cdot ((\mathcal{A}^d \cdot \mathcal{A}_{\mathrm{all}}) \times \mathcal{A}_\lambda)$ of size $O((1+d)|\mathcal{A}|)$. □

We continue with a few facts from [Devolder et al. 1994]. The adherence of a language L, denoted by $\mathrm{Adh}(L)$, consists of all the right infinite words w such that every prefix of w is a prefix of some word in L. The following is shown in [Devolder et al. 1994].

Lemma 8. *The following statements hold true for every regular code* L.
1. If L *has a finite decoding delay then* $L^\omega \cap L^* \mathrm{Adh}(L) = \emptyset$.
2. If L *is an* ω*-code and* $L^\omega \cap L^* \mathrm{Adh}(L) = \emptyset$*, then* L *has a finite decoding delay.* □

For a regular language L, let $\mathrm{minstates}(L)$ be the smallest number of states in any *NFA* accepting L. The proof of the second statement of the lemma can be modified appropriately to obtain the following result.

Proposition 9 *If a regular code* L *has minimum finite decoding delay* d*, then* $d \leq \mathrm{minstates}(L)$. □

Corollary 9. *The following problem is computable in time* $O(\mathrm{tf}(|\mathcal{A}|^2)\log|\mathcal{A}|)$.
Input: a trim NFA $\mathcal{A}$ *accepting a code.*
Output: the minimum decoding delay of $L(\mathcal{A})$.

Proof. Let d be the number of states of $\mathcal{A}$. Then, $\mathrm{minstates}(L(\mathcal{A})) \leq d$. As $d \leq |\mathcal{A}|$, the minimum decoding delay of $L(\mathcal{A})$ is either infinite or equal to one of the values $0, \ldots, |\mathcal{A}|$. The algorithm performs binary search on this list of values such that, for each current value m, it tests whether $L(\mathcal{A})$ has decoding delay m in time $O(\mathrm{tf}((1+m)|\mathcal{A}|))$. The time complexity follows when we note that each m is no greater than $|\mathcal{A}|$ and that the search performs at most $\log|\mathcal{A}|$ steps. □

7 Rational Channels and SID-Channels

Usually, it is not too difficult to construct transducers for realizing certain SID-channels. Here we show a general construction method to realize any SID-channel by a transducer. This 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 [Konstantinidis 2001] 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 certain types of words, called *error functions,* which are applied on words over X 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 $x = \lambda a\lambda b\lambda a\lambda b\lambda$, we see that there are four possible positions for a substitution (the four symbols of x), five possible positions for an insertion (the five λs), 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 on $X \cup \{\lambda\}$ (no error at that position), $\mathbf{i}_b$ is the function on $\{\lambda\}$ that maps λ to b (insertion in that position), $\mathbf{d}$ is the function that maps every symbol in X to λ (deletion at that position), and $\mathbf{s}$ is any *substitution function:* a function that maps any $x \in X$ onto a symbol in $X \setminus \{x\}$. Thus, if we consider the error function $\mathbf{h} = \mathbf{i}_b\mathbf{eeseeede}$, then

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

Note that the alphabet of basic error functions is infinite as it contains $\mathbf{i}_u$, for every $u \in X^+$. 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:

$$\mathbf{c} = \begin{cases} \mathbf{h}_{2n}, & \text{if } \mathbf{g}_0 = \mathbf{e}; \\ \mathbf{g}_0, & \text{if } \mathbf{h}_{2n} = \mathbf{e}; \\ \mathbf{i}_{u_1u_2}, & \text{if } \mathbf{h}_{2n} = \mathbf{i}_{u_1} \text{ and } \mathbf{g}_0 = \mathbf{i}_{u_2}. \end{cases}$$

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 10. 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.

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. Here we only give the formal definition of the SID-channel $\sigma \odot \delta(m,\ell)$, where m and ℓ are positive integers with $m < \ell$, that permits at most m (scattered) substitutions and deletions in every ℓ consecutive input symbols – see [Konstantinidis 2001] for the full class of SID-channels. The channel is equal to $R(S_{m,\ell})$, where $S_{m,\ell}$ is the SID-support

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

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 10 *For every SID-support F there effectively exists a transducer $\mathcal{T}_F$ such that $R(\lfloor F \rfloor) = R(\mathcal{T}_F)$. 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 $\gamma = R(\lfloor F \rfloor)$.*

Proof. The transducer $\mathcal{T}_F$ 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 every $\mathbf{g}, \mathbf{h} \in \hat{F}$ and $x, x' \in X^*$:

- $\alpha\lambda/\lambda\mathbf{g}$
- $\mathbf{g}x/x'\mathbf{h}$, 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 $|x| \le \ell_F$ and $x' = \mathbf{f}(x)$ for some $\mathcal{H}$-prefix $\mathbf{f}$ of $\mathbf{h}$.

Intuitively, if $\mathcal{T}_F$ 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 $(x,y) \in R(\mathcal{T}_F)$. Then, there is a computation $\alpha\lambda/\lambda\mathbf{h}_1 x_1/x'_1 \cdots \mathbf{h}_n x_n/x'_n\omega$ of $\mathcal{T}_F$, 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 $(x,y) \in R(\lfloor F \rfloor)$ as required.

For the converse, assume $(x,y) \in R(\lfloor F \rfloor)$. 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 $\mathbf{h}(x) = \mathbf{h}_1(x_1) \cdots \mathbf{h}_n(x_n)$. 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,

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

is a computation of $\mathcal{T}_F$ and, therefore, $(x,y) \in R(\mathcal{T}_F)$ 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 $\Delta = \{q_0x/xq_0 \mid x \in X\} \cup \{q_0x/\lambda q_1\}$. Assume that $\gamma = R(\lfloor F \rfloor)$ 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 that $(wx_1x_2, wx_2) \in \gamma$, which is impossible. $\square$

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.

References

[Béal et al. 2000] Béal, M-P., Carton, O., Prieur, C., Sakarovitch, J.: "Squaring Transducers: an Efficient Procedure for Deciding Functionality and Sequentiality of Transducers"; Proc. LATIN 2000, Lecture Notes in Computer Science 1776, Springer-Verlag, Berlin (2000), 397–406.

[Berstel 1979] Berstel, J.: "Transductions and Context-Free Languages"; B. G. Teubner, Stuttgart (1979).

[Berstel and Perrin 1985] Berstel, J., Perrin, D.: "Theory of Codes"; Academic Press, Orlando (1985).

[Dassow et al. 1997] Dassow J., Mitrana V., Păun G.: "Point Mutations in Context-free Languages"; Proc. 3rd International Conference, Developments in Language Theory, Aristotle University, Thessaloniki (1997), 429–446.

[Devolder et al. 1994] Devolder J., Latteux M., Litovsky I., Staiger L.: "Codes and Infinite Words"; Acta Cybernetica, 11 (1994), 241–256.

[Gurari and Ibarra 1983] Gurari, E. M., Ibarra O. H.: "Finite-valued and Finitely Ambiguous Transducers"; Math. Systems Theory, 16 (1983), 61–66.

[Hartnett 1968] Hartnett W. E.: "Generalization of Tests for Certain Properties of Variable-Length Codes"; Information and Control, 13 (1968), 20–45.

[Harju and Karhumäki 1997] Harju, T., Karhumäki, J.: "Morphisms". In [Rozenberg and Salomaa 1997], 439–510.

[Head and Weber 1993] Head, T., Weber, A.: "Deciding Code Related Properties by Means of Finite Transducers"; Proc. Sequences II, Methods in Communication, Security, and Computer Science, Springer-Verlag, Berlin (1993), 260–272.

[Jürgensen and Konstantinidis 1996] Jürgensen, H., Konstantinidis, S.: "Error-Correction for Channels with Substitutions, Insertions, and Deletions"; Proc. Information Theory and Applications 2, 4th Canadian Workshop on Information Theory, Lecture Notes in Computer Science 1133, Springer-Verlag, Berlin (1996), 149–163.

[Jürgensen and Konstantinidis 1997] Jürgensen, H., Konstantinidis, S.: "Codes". In [Rozenberg and Salomaa 1997], 511–607.

[Salomaa et al. 1994] Salomaa, K., Yu, S., Zhuang, Q.: "The State Complexities of Some Basic Operations on Regular Languages"; Theoretical Computer Science, 125 (1994), 315–328.

[Konstantinidis 1999] Konstantinidis, S.: "Structural Analysis of Error-Correcting Codes for Discrete Channels that Involve Combinations of Three Basic Error Types"; IEEE Transactions on Information Theory, 45 (1999), 60–77.

[Konstantinidis 2001] Konstantinidis, S.: "An Algebra of Discrete Channels that Involve Combinations of Three Basic Error Types"; Information and Computation, 167 (2001), 120–131.

[Konstantinidis and O'Hearn 2002] Konstantinidis, S., O'Hearn, A.: "Error-Detecting Properties of Languages"; Theoretical Computer Science, to appear.
[McCloskey 1996] McCloskey, R.: "An $O(n^2)$ Time Algorithm for Deciding Whether a Regular Language is a Code"; Journal of Computing and Information, 2 (1996), 79–89.
[Rozenberg and Salomaa 1997] Rozenberg, G., Salomaa, A.: "Handbook of Formal Languages", Vol. I, Springer-Verlag, Berlin (1997).
[Sato 1979] Sato, K.: "A Decision Procedure for the Unique Decipherability of Multi-valued Encodings"; IEEE Transactions on Information Theory, 25 (1979) 356–360.
[Schützenberger 1975] Schützenberger, M. P.: "Sur les Relations Rationnelles"; Proc. Automata Theory and Formal Languages, Lecture Notes in Computer Science 33, Springer-Verlag, Berlin (1975), 209–213.
[Yu 1997] Yu, S.: "Regular Languages". In [Rozenberg and Salomaa 1997], 41–110.

Appendix

In this appendix we give the proof of Proposition 2.

Proof. Because of symmetry we only give the proof for $\mathcal{T} \downarrow \mathcal{A}$. Let $\mathcal{T} = (X, Y, Q, q_0, Q_+, E)$ and let $\mathcal{A} = (Z, P, p_0, P_+, D)$. The required transducer $\mathcal{T} \downarrow \mathcal{A}$ is equal to $(X, Y, PQ, p_0q_0,\ P_+Q_+, C)$, where the set of transitions C is constructed as follows: Initially, $C = \emptyset$; then:

- for each $p \in P$ and for each $q\lambda/yq' \in E$, add $pq\lambda/ypq'$ in C;
- for each $q \in Q$ and for each $p\lambda p' \in D$, add $pq\lambda/\lambda p'q$ in C;
- for each $pzp' \in D$ and $qx/yq' \in E$, if $z = x$ add $pqx/yp'q'$ in C.

By the above construction it follows that, if $pqx/yp'q'$ is in C then, for pxp' we have that pxp' is in D or, $x = \lambda$ and $p = p'$. Moreover, for qx/yq' we have that qx/yq' is in E, or $x = y = \lambda$ and $q = q'$. Obviously, the transducer $\mathcal{T} \downarrow \mathcal{A}$ is in standard form and $|C| \leq |P||E| + |Q||D| + |E||D|$. Also, by the assumptions about $\mathcal{T}$ and $\mathcal{A}$, we have that $|P| \leq |D| + 1$ and $|Q| \leq |E| + 1$. Hence, the size of $\mathcal{T} \downarrow \mathcal{A} = O(|\mathcal{A}||\mathcal{T}|)$. In general, the transducer might not be trim, but we can transform it to an equivalent trim transducer in standard form in time $O(|\mathcal{T} \downarrow \mathcal{A}|)$.

For the correctness of the construction we need to show that $R(\mathcal{T} \downarrow \mathcal{A}) = R(\mathcal{T}) \downarrow L(\mathcal{A})$. First assume that $(w, z) \in R(\mathcal{T} \downarrow \mathcal{A})$; then there is a computation $p_0q_0w_1/z_1p_1q_1 \cdots w_n/z_np_nq_n$ such that $p_nq_n \in P_+Q_+$, $w = w_1 \cdots w_n$, $z = z_1 \cdots z_n$, and $p_{i-1}q_{i-1}w_i/z_ip_iq_i$ is in C for all $i \in \{1, \ldots, n\}$. Consider the words $\omega = p_0w_1p_1 \cdots w_np_n$ and $\xi = q_0w_1/z_1q_1 \cdots w_n/z_nq_n$ and note that $\mathcal{A}$ is equivalent to $\mathcal{A}' = (Z, P, p_0, P_+, D')$, where $D' = D \cup \{p\lambda p \mid p \in P\}$, and that $\mathcal{T}$ is equivalent to the transducer $\mathcal{T}' = (X, Y, Q, q_0, Q_+, E')$, where $E' = E \cup \{q\lambda/\lambda q \mid q \in Q\}$. Then, as every $p_{i-1}w_ip_i$ is in D', the word ω is a computation of $\mathcal{A}'$ such that $p_n \in P_+$ which implies $w_1 \cdots w_n \in L(\mathcal{A}')$. Hence, $w \in L(\mathcal{A})$. Similarly, the word ξ is a computation of $\mathcal{T}'$ with $q_n \in Q_+$ which implies that $(w, z) \in R(\mathcal{T})$, as required.

Now assume that $(w, z) \in R(\mathcal{T})$ and $w \in L(\mathcal{A})$. Then there is a computation $\omega = p_0w_1p_1 \cdots w_mp_m$ of $\mathcal{A}$ with $p_m \in P_+$, $w = w_1 \cdots w_m$ and $p_{i-1}w_ip_i \in D$, and a computation $\xi = q_0u_1/z_1q_1 \cdots u_n/z_nq_n$ of $\mathcal{T}$ with $q_n \in Q_+$, $w = u_1 \cdots u_n$, $z = z_1 \cdots z_n$, and $q_{i-1}u_i/z_iq_i \in E$. Let $\ell = |w|$; then $w = w_{i_1} \cdots w_{i_\ell} = u_{j_1} \cdots u_{j_\ell}$, where the symbols w_{i_r} and u_{j_r} are in X for all $r \in \{1, \ldots, \ell\}$. Consider the words ω' and ξ' that obtain from ω and ξ, respectively, as follows:

– For $r = 1, \ldots, \ell$, let $d_r = (i_r - i_{r-1}) - (j_r - j_{r-1})$, where we assume that $i_0 = j_0 = 0$. If $d_r < 0$ insert in ω the word $(\lambda p_{i_r-1})^{|d_r|}$ before $w_{i_r} p_{i_r}$. If $d_r > 0$ insert in ξ the word $(\lambda/\lambda q_{j_r-1})^{d_r}$ before $u_{j_r}/z_{j_r} q_{j_r}$. Moreover, let $d = (m - i_\ell) - (n - j_\ell)$. If $d < 0$ append the word $(\lambda p_m)^{|d|}$ at the end of ω. If $d > 0$ append the word $\lambda/\lambda q_n$ at the end of ξ.

It follows that ω' is of the form $p'_0 v_1 p'_1 \cdots v_k p'_k$ and ξ' is of the form $q'_0 v_1/y_1 q'_1 \cdots v_k/y_k q'_k$ such that $w = v_1 \cdots v_k$ and $z = y_1 \cdots y_k$, where $v_i \in X \cup \{\lambda\}$ and $y_i \in Y \cup \{\lambda\}$. We argue now that the word $\zeta = p'_0 q'_0 v_1/y_1 p'_1 q'_1 \cdots v_k/y_k p'_k q'_k$ is a computation of the transducer $\mathcal{T} \downarrow \mathcal{A}$; therefore, $(w, z) \in R(\mathcal{T} \downarrow \mathcal{A})$. Indeed, consider each $\zeta_t = p'_{t-1} q'_{t-1} v_t/y_t p'_t q'_t$. Then, $\omega'_t = p'_{t-1} v_t p'_t$ occurs in ω' and $\xi'_t = q'_{t-1} v_t/y_t q'_t$ occurs in ξ'. Moreover, for ω'_t we have that either it is in D or that $v_t p'_t = \lambda p'_{t-1}$ was inserted in ω; and for ξ'_t, either it is in E or $v_t/y_t q'_t = \lambda/\lambda q'_{t-1}$ was inserted in ξ. Hence, there are four possibilities for the structure of ω'_t and ξ'_t each of which implies that ζ_t is in C. □