Recognizable Transductions, Saturated Transducers and Edit Languages

Stavros Konstantinidis\textsuperscript{1}, Nicolae S\u{a}ntean\textsuperscript{2}\textsuperscript{**}, and Sheng Yu\textsuperscript{2}

\textsuperscript{1} Saint Mary's University, Halifax B3H 3C3, NS, Canada
Department of Mathematics and Computing Science,
email: stavros@cs.smu.ca
\textsuperscript{2} University of Western Ontario, London N6A 5B8, ON, Canada
Department of Computer Science,
email: nic@csd.uwo.ca, syu@csd.uwo.ca

Abstract. Recognizable transductions constitute a well known, proper subclass of rational transductions. To our knowledge, there has been no characterization of recognizable transductions by a well-defined subclass of transducers. In this work we observe that there is a connection between recognizable transductions and languages consisting of edit strings. More specifically, we define a saturated transducer to be a transducer with the property that, for every pair of words it realizes, the transducer, when viewed as an automaton over the edit alphabet, accepts all possible edit strings transforming the first word of the pair into the second one. Our main result is a constructive proof that the class of recognizable transductions coincides with the class of saturated transductions. We also revisit closure properties of recognizable transductions using saturated transducers and discuss the natural role of these objects in edit distance problems.

Keywords: Transducer; Recognizable transduction; Saturated transducer; Edit string; Edit language; Edit distance

1 Introduction

Recognizable transductions constitute a well known, proper subclass of rational transductions, the latter being the class of all binary relations of words realized by finite transducers. A well known characterization of recognizable transductions is given by Mezei's theorem [Eilenberg, 1974, note at p. 75]. Until now, however, there has been no characterization of recognizable transductions by a well-defined, special subclass of transducers. In this work we observe that there is an intimate connection between recognizable transductions and edit languages, that is, languages consisting of edit strings. An edit string (or string alignment) is a special word consisting of edit operations, and describes the sequence of...

\textsuperscript{*} This work was supported by the Natural Science and Engineering Research Council of Canada grants R220259 and OGP0041630.

\textsuperscript{**} corresponding author
changes (substitutions, insertions and deletions of symbols) that can transform a word into another word. Edit strings can be used to define formally concepts related to distances between words [Sankoff, Kruskal, 1999] and, in fact, recently ([Kari, Konstantinidis, 2002], [Mohri, 2003],[Kari et al., 2003]) there have been systematic treatments of edit languages (also called e-systems) in the sense of language theory. In the context of word and language distances, the main difference between a transduction and an edit language is that the latter describes the exact changes that are permitted in transforming words to words, whereas the former describes the result of these transformations.

This paper introduces the concept of saturated transducer and observes that this concept constitutes a natural point of connection between recognizable transductions and edit languages. More specifically, a saturated transducer is a transducer with the property that, for every pair of words it realizes, the transducer, when viewed as an automaton over the edit alphabet, accepts all possible edit strings transforming the first word of the pair into the second one. The main result of the paper is that the class of recognizable transductions coincides with the class of transductions realized by saturated transducers. We also provide other basic results on saturated transducers and discuss their use in edit distance problems. In the next paragraph we give a short overview of our paper.

The paper is organized as follows. In the next section, we provide the formal definitions about rational transducers, recognizable transductions and edit languages. Section 3 introduces saturated transducers and discusses several basic operations on these objects. These operations can be used to provide constructive proofs (by means of saturated transducers) of closure properties of recognizable transductions, such as intersection, composition and concatenation. Moreover, the descriptive complexity of these operations can be used to evaluate the time complexity of algorithms utilizing them. Section 4 contains the main result of the paper about the equivalence of saturated and recognizable transductions. The proof is constructive in the following sense. Given a tuple consisting of an even number of finite automata — according to Mezei’s theorem such a tuple specifies a recognizable transduction — there is an effective construction of a saturated transducer realizing the transduction specified by the tuple. Moreover, there is a constructive proof for the converse problem. In Section 5 we elaborate on the use of saturated transducers in problems related to the edit distance of words and languages. The method here is not new, in the sense that certain examples of saturated transducers have already been used for such problems, but we believe that the method is better understood with our systematic study of saturated transducers. Our observations in this context lead us to the question of whether the transduction consisting of all pairs of distinct words of some regular language is recognizable. We show that it is not recognizable in the case of infinite languages. Finally, Section 6 contains a few concluding remarks.
2 Preliminary Notions and Notations

We assume known basic notions of finite automata: DFA (deterministic finite automaton), NFA (nondeterministic finite automaton) and $\varepsilon$-NFA (NFA with $\varepsilon$-transitions) – a review of these terms can be found in [Hopcroft and Ullman, 1979], [Yu, 1997]. We also assume known the basic notions of semigroup (monoid) theory [Howie, 1976] and of rational and recognizable sets in arbitrary monoids ([Berstel, 1979], [Eilenberg, 1974]). We recall that the class of monoids is closed under cartesian product.

Let $(M, \cdot, 1_M)$ be a monoid which consists of a carrier set $M$ equipped with a binary associative operation “$\cdot$” and an unit “$1_M$”. By $\text{Rat}(M)$ we denote the family of rational subsets of $M$ and by $\text{Rec}(M)$ we denote the family of recognizable subsets.

If $X$ and $Y$ are finite alphabets (nonempty sets of symbols), we denote by $X^*$ and $Y^*$ their freely generated monoids. Any element of $X^*$ or $Y^*$ is called a word, i.e., a finite string of symbols. We denote by $\varepsilon$ the word with no symbols, i.e., the empty word. By $X^* \times Y^*$ we understand the direct product of the monoids $X^*$ and $Y^*$, i.e., the monoid of word relations. We will use the terms “word relation” and “transduction” interchangeably. Notice that this monoid is finitely generated, in the sense that there exists a finite subset $G$, called a set of generators, such that $G^* = X^* \times Y^*$ (indeed, take $G = (X \times \{\lambda\}) \cup (\{\lambda\} \times Y)$). Notice also that $X^* \times Y^*$ is not necessarily a free monoid, in the sense that it may not exist a set of generators which generate each element of the monoid in a unique way (for example, $G$ above may generate an element in more than one way: $(x, y) = (x, \lambda) \cdot (\lambda, y) = (\lambda, y) \cdot (x, \lambda)$. As a consequence of McKnight’s theorem ([McKnight, 1964]) we have that

$$\text{Rec}(X^* \times Y^*) \subseteq \text{Rat}(X^* \times Y^*)$$

inclusion which is strict in general. For example, the transduction $\{(a^i, b^j)/i \geq 0\}$ can be proven to be rational without being recognizable.

In $X^* \times Y^*$, recognizable and rational sets may be specified by finite state machines. For example, each rational transduction $\tau$ is represented by some finite transducer $T = (Q, X, Y, \Delta, q_0, F)$, where

1. $Q$ is a finite set of states;
2. $\Delta \subseteq Q \times X^* \times Y^* \times Q$ is a finite set of transitions;
3. $q_0$ is an initial state, $F \subseteq Q$ is a set of final states;
4. a successful computation of $T$ is a sequence
   $$c = (q_0, x_1, y_1, q_1),\ldots,(q_{n-1}, x_n, y_n, q_n)$$
   where $(q_{i-1}, x_i, y_i, q_i) \in \Delta$ for all $i \in \{1,\ldots,n\}$, and $q_n \in F$. The label of $c$, denoted by $|c|$ is the pair of words $(x_1\ldots x_n, y_1\ldots y_n)$;
5. $\tau = |T| = \{(u, v)/(u, v) = |c|, \text{ for some successful computation } c\}$.

The alphabet $X$ is sometime called the input alphabet and $Y$ the output alphabet. It has been shown (for example in [Berstel, 1979, §III.6, p. 79]) that a
transducer with labels in $X^* \times Y^*$ is equivalent with a transducer having labels only in $(X \cup \{\epsilon\}) \times (Y \cup \{\epsilon\})$. We bring this observation further, by noticing that one can eliminate all “null” transitions, i.e., transitions of the form $(\epsilon, \epsilon)$. However, for the sake of formalism, it is useful to consider all states having null loops, i.e., we have a transition $(p, \epsilon, \epsilon, p)$ for each state $p$ of the transducer. Then we give the following definition:

**Definition 1.** A transducer is in **standard form** if it has transitions with labels in $(X \cup \{\epsilon\}) \times (Y \cup \{\epsilon\})$ and each state has an $(\epsilon, \epsilon)$-loop to itself.

Then each rational transduction is realized by a transducer in standard form.

We define the **size** of a finite state machine $M$ in general (hence of a transducer, in particular), as being the number of all its states together with all its transitions, and we denote it by $\text{size}(M)$.

In the case of recognizable transductions, one can use Mezei’s characterization (as in [Eilenberg, 1974, §3.12, Prop. 12.2 & note at p. 75]) to represent a transduction $\tau \in \text{Rec}(X^* \times Y^*)$ by a tuple of finite automata $(A_1, B_1, \ldots, A_n, B_n)$ such that

$$\tau = \bigcup_{i=1}^{n} \mathcal{L}(A_i) \times \mathcal{L}(B_i),$$

where by $\mathcal{L}(A)$ we understand the language accepted by the automaton $A$ (automata $A_i$ are over $X$ and automata $B_i$ are over $Y$). We say that any recognizable transduction is a finite union of blocks (a block is a direct product of two regular languages) - see for example [Sakarovitch, 2003, §II.2, p.272, Corollary 2.20].

As a general observation, not much effort has been spent on the study of finite machines designed to precisely accept recognizable sets. Our paper addresses this issue and reveals the close connection between recognizable sets and edit languages – defined in the following.

Let $E$ be the set consisting of all elements of the form $(a/\epsilon)$, $(\epsilon/b)$ and $(a/b)$, where $a \in X$ and $b \in Y$. We treat the elements of $E$ as symbols which denote the so-called **edit operations**: deletion, insertion and substitution (for example, the meaning of operation “$(a/\epsilon)$” is “deletion of $a$”). Then, by $E^*$ we denote the language of **edit strings**, i.e., the language of words over the alphabet $E$. The empty edit string over $E$ will be denoted by $(\epsilon/\epsilon)$.

Edit strings can implement transductions as the following example shows: if $X = Y = \{a, b\}$ then each of the following edit strings define the transduction $\{(aba, bab)\}$:

- $\epsilon = (a/b)(b/a)(a/b)$
- $f = (a/\epsilon)(b/b)(a/a)(\lambda/b)$
- $g = (a/\epsilon)(b/\epsilon)(a/\epsilon)(\epsilon/b)(\epsilon/a)(\epsilon/b)$

We say that each of the edit strings $\epsilon$, $f$ and $g$ “transforms the word $aba$
into the word $bab'$. Notation wise, we use the lowercase letters $e, f, g$ to denote edit strings.

In this paper we are interested in sets of edit strings, i.e., in edit languages. Such languages are simply subsets of $E^*$. 

3 Saturated Transducers: Definition and Basic Results

The notion of saturated transducer originates in the simple idea that a computation of a finite transducer in standard form defines both a pair of words and an unique edit string which transforms a word into another one.

Let $X$ and $Y$ be input and output alphabets and $E$ be the alphabet of edit operations over $X$ and $Y$. Across this paper we will frequently refer to the following monoid homomorphism:

$$h : E^* \rightarrow X^* \times Y^* ,$$

given by: $h(e/e) = (e, e), h(a/e) = (a, e), h(e/b) = (e, b), h(a/b) = (a, b)$, for all $a \in X$ and $b \in Y$. Due to its importance to our matter, we name this morphism the edit morphism over $X$ and $Y$. It should be clear that for any pair of words $(u, v), h^{-1}([[u, v]])$ consists of all edit strings that transform $u$ into $v$.

Let $T$ be a transducer over $X$ and $Y$, in standard form. By $h^{-1}(T)$ we denote the finite automaton over $E$, obtained from $T$ by replacing each transition label $(x, y)$ with the symbol $(x/y) \in E \cup \{(e/e)\}$. Then $h^{-1}(T)$ will be an $e$-NFA over $E$.

Conversely, given a finite automaton $A$ over $E$, by $h(A)$ we understand the transducer over $X$ and $Y$ obtained from $A$ by replacing each transition label $(x, y)$ with the pair $(x, y) \in X^* \times Y^*$. Then $h(A)$ is in standard form, up to the missing $(e, e)$-loops for each state. For easing the formalism we assume that these loops are present and that $h(A)$ is readily in standard form.

In the previous section we have defined what is meant by a successful computation (and its label) of a transducer $T = (Q, X, Y, \Delta, q_0, F)$. Let

$$c = (q_0, x_1, y_1, q_1), ..., (q_{n-1}, x_n, y_n, q_n)$$

be a successful computation in $T$. If the transducer $T$ is in standard form, then all pairs $(x_i, y_i)$ can be viewed as edit operations, or null operations, and we can define the edit string corresponding to $c$ as $||c|| := (x_1/y_1)_i ..., (x_n/y_n)_i$.

Notice that we have $h(||c||) = |c|$, where $h$ is the edit morphism from $X$ to $Y$. Then the transducer $T$ defines a transduction

$$|T| = \{(u,v)/(u,v) = |c|, \text{ where } c \text{ is a successful computation in } T\} ,$$

and an edit language

$$||T|| = \{e \in E^* / e = ||c||, \text{ where } c \text{ is a successful computation in } T\} ,$$

in other words $||T|| = L(h^{-1}(T))$. In the next definition we use the meaning of $h$ as a monoid morphism.
Definition 2. A transducer $T$ in standard form is saturated if and only if

$$h^{-1}([T]) = ||T||.$$ 

In other words, $T$ is saturated if and only if for any accepted pair of words $(u,v) \in X^* \times Y^*$, and for any edit string $e \in E^*$ which transforms $u$ into $v$ there exists a successful computation $c$ in $T$ such that $||c|| = e$.

Notice that the property of saturation can be generalized to arbitrary transducers. Indeed, let $T$ be an arbitrary transducer. A successful computation of $T$ is said to be admissible if and only if its transitions have labels in $(X \cup \{\epsilon\}) \times (Y \cup \{\epsilon\})$. Then we can define the edit language of $T$ as being

$$||T|| = \{e \in E^*/e = ||c||, \text{ where } c \text{ is an admissible computation in } T\}.$$ 

From here the definition of a saturated transducer is extended naturally to arbitrary transducers. Remark that any saturated transducer is equivalent to a saturated transducer in standard form. Indeed, let $T$ be an arbitrary saturated transducer. It suffices to observe that one can discard all transitions with labels not in $(X \cup \{\epsilon\}) \times (Y \cup \{\epsilon\})$ without changing the transduction realized by $T$.

We say that a transduction over $X$ and $Y$ is saturated if and only if there exists a saturated transducer $T$ such that $\tau = |T|$. We denote by

$$Sat(X^* \times Y^*)$$

the family of saturated transductions. Then clearly $Sat(X^* \times Y^*) \subseteq Rat(X^* \times Y^*)$.

In this section we are interested in basic operations on saturated transducers with the aim of providing constructive proofs for the closure properties of saturated transductions. As it turns out, many known operations on ordinary automata and transducers result in saturated transducers with no extra effort when applied on saturated transducers. For example, the standard product constructions on finite automata, possibly with $\epsilon$ transitions, for union and intersection would result in saturated transducers when applied on saturated transducers. The same happens in the case of the product construction for the composition of transducers.

In the following operations, the operands $A_1$ and $A_2$ are arbitrary finite automata, possibly with $\epsilon$ transitions (unless specified otherwise), and the operands $T_1$ and $T_2$ are arbitrary finite transducers in standard form.

det$(A_1)$: is the automaton obtained by determinization and completion of $A_1$. $\overline{A_1}$, where $A_1$ is a DFA: the DFA that results when we complete $A_1$ and change its non-final states to final, and vice versa. It is well known that $\overline{A_1}$ accepts the complement of the language accepted by $A_1$ and that size$(\overline{A_1}) = O$(size($A_1$)).

$A_1 \times A_2$: is a saturated transducer of size $O$(size($A_1$) · size($A_2$)) such that

$$|A_1 \times A_2| = \mathcal{L}(A_1) \times \mathcal{L}(A_2).$$
The transducer $A_1 \times A_2$ consists of the transitions $((p_1, p_2), x_1, x_2, (q_1, q_2))$ for all transitions $(p_1, x_1, q_1)$ of $A_1$ and $(p_2, x_2, q_2)$ of $A_2$, where we assume that there is always an $\epsilon$ transition from each state to itself - see [Kari et al., 2003] for more details, where the notation $A_1 \cap \mathcal{E} A_2$ is used instead of $A_1 \times A_2$.

$T_1 \cap T_2$: is the transducer in standard form that is obtained when we apply the standard product construction on automata for language intersection on the automata $h^{-1}(T_1)$ and $h^{-1}(T_2)$ over the edit alphabet $\mathcal{E}$. The size of $T_1 \cap T_2$ is $O(\text{size}(T_1) \cdot \text{size}(T_2))$. Obviously, $|T_1 \cap T_2| = |T_1| \cap |T_2|$. The advantage of this construction over $T_1 \cup T_2$ is that the automaton $h^{-1}(T_1 \cup T_2)$ is a DFA when both of $h^{-1}(T_1)$ and $h^{-1}(T_2)$ are DFAs.

$T_1 \cup T_2$: is the transducer in standard form that is obtained when we apply the standard product construction on transducers for transduction composition (see [Mohri, 2003]), hence,

$$|T_2 \cup T_1| = |T_2| \cup |T_1|.$$

Again, the size of $T_2 \cup T_1$ is $O(\text{size}(T_1) \cdot \text{size}(T_2))$. The transducer $T_2 \cup T_1$ consists of the transitions $((p_1, p_2), x, y, (q_1, q_2))$ for all pairs of transitions $(p_1, x, y, q_1)$ in $T_1$ and $(p_2, y, z, q_2)$ in $T_2$ and $y$ in $Y \cup \{\epsilon\}$.

$T_1$: is the transducer $h(\det(h^{-1}(T_1)))$ such that

$$|T_1| = |T_1|.$$

If $h^{-1}(T_1)$ is an NFA then the size of $T_1$ could be exponential with respect to the size of $T_1$. On the other hand, if $h^{-1}(T_1)$ is a DFA then the size of $T_1$ is $O(\text{size}(T_1))$.

**Example 1.** In Fig.1 we are given two automata $A_1$ and $A_2$, $A_1$ accepting all words which in binary have an odd value and $A_2$ accepting all words which have an even length. Following the above construction we obtain a saturated transducer for $A_1 \times A_2$.

**Lemma 1.** If $T_1$ and $T_2$ are saturated transducers then $T_1 \cap T_2$, $T_1 \cup T_2$, $T_1 \cup T_2$, $T_2 \circ T_1$ and $T_1$ are saturated.

**Proof.** We prove only that $T_2 \circ T_1$ is saturated. The rest is left to the reader.
We need to show that for any pair \((x, z)\) in \(|T_2 \circ T_1|\) and for any edit string \(e\) with \(h(e) = (x, z)\), it is the case that \(e\) is in \(|T_2 \circ T_1|\). Suppose that
\[
ed = (x_1/z_1) \cdots (x_n/z_n),
\]
where each \((x_i/z_i)\) is an edit operation. There is a computation \(e'\) of \(T_2 \circ T_1\) such that \(|e'|\) is some edit string \((x'_1/z'_1) \cdots (x'_m/z'_m)\) and \(h(|e'|) = (x, z)\). By the definition of \(T_2 \circ T_1\), there are successful computations \(c'_1\) and \(c'_2\) of \(T_1\) and \(T_2\), respectively, such that the edit strings \(|c'_1|\) and \(|c'_2|\) are of the form \((x'_1/y'_1) \cdots (x'_m/y'_m)\) and \((y'_1/z'_1) \cdots (y'_m/z'_m)\), respectively. Let \(y\) be the word \(y'_1 \cdots y'_m\). We continue by distinguishing two cases.

Firstly, suppose that \(m \leq n\). Let \(y_j = y'_j\) for \(j \leq m\), and \(y_j = e\) for \(j = m + 1, \ldots, n\). Consider the edit strings
\[
e_1 = (x_1/y_1) \cdots (x_n/y_n) \text{ and } e_2 = (y_1/z_1) \cdots (y_n/z_n).
\]
As \(T_1\) and \(T_2\) are saturated, and \(h(e_1) = (x, y)\) and \(h(e_2) = (y, z)\), there are successful computations \(c_1\) and \(c_2\) of \(T_1\) and \(T_2\), respectively, such that \(|c_1| = e_1\) and \(|c_2| = e_2\). Then, by definition of the transducer \(T_2 \circ T_1\), there is a computation \(c\) of this transducer such that \(|c| = e\), as required.

Secondly, suppose that \(m > n\). The proof of this case is similar to the first one and is left to the reader.

\[\square\]

**Example 2.** Let \(\tau_1 \tau_2\) be transductions given by
\[ \tau_1 = \{(u, v) / 2^u \text{ is odd}, v \in \{a\}^*\}; \]
\[ \tau_2 = \{(u, v) / u \in \{a\}^*, 2^v \text{ is even}\} \]

where by \(2^u\) we understand the value of \(u\) as a binary number. The first two saturated transducers in Fig. 2 realize them. Then, using the above construction we obtain a saturated transducer (shown also in Figure 2) which realizes the transduction
\[ \{(u, v) / 2^u \text{ is odd, and } 2^v \text{ is even}\}, \]
which is their composition.

\[
\begin{align*}
\tau_1 & : \\
(0, a), (0, e), (\epsilon, a) & \rightarrow (0, a), (1, a), (\epsilon, a) \\
(1, a), (1, \epsilon) & \rightarrow (1, a), (1, \epsilon)
\end{align*}
\]

\[
\begin{align*}
\tau_2 & : \\
(a, 0), (\epsilon, 0), (a, \epsilon) & \rightarrow (a, 1), (\epsilon, 1), (a, \epsilon) \\
(a, 1), (\epsilon, 1), (a, \epsilon) & \rightarrow (a, 1), (\epsilon, 1), (a, \epsilon)
\end{align*}
\]

\[
\begin{align*}
\tau_2 \circ \tau_1 & : \\
(0, 0), (0, 0), (\epsilon, 0) & \rightarrow (0, 0), (0, 1), (\epsilon, 1), (0, \epsilon) \\
(0, 1) & \rightarrow (0, 1), (\epsilon, 1), (0, \epsilon)
\end{align*}
\]

\[
\begin{align*}
(0, 0), (1, 0), (1, 0) & \rightarrow (0, 0), (1, 0), (1, 1) \\
(1, 0) & \rightarrow (1, 0), (\epsilon, 0), (1, \epsilon)
\end{align*}
\]

\[
\begin{align*}
(1, 0), (1, 1) & \rightarrow (1, 0), (1, 1), (\epsilon, 1) \\
(1, 1) & \rightarrow (1, 1), (\epsilon, 1)
\end{align*}
\]

\[
\begin{align*}
(1, 0), (1, 1), (\epsilon, 1) & \rightarrow (1, 0), (1, 1), (\epsilon, 1)
\end{align*}
\]

**Fig. 2.** Composition of saturated transducers.

A natural question that arises here is whether saturated transductions are closed under the Kleene-star operation and concatenation. The first operation is discussed in the next section. For the second one consider two transducers \(T_1\) and \(T_2\) and the standard construction that connects each final state of \(T_1\) with the start state of \(T_2\) using an \((\epsilon, e)\)-transition, such that the new transducer realizes \(|T_1| \cdot |T_2|\). Unfortunately, however, this transducer is not necessarily saturated when both \(T_1\) and \(T_2\) are saturated. For example, if we connect a saturated transducer for \(\{(a, ab)\}\) with a saturated transducer for \(\{(ab, b)\}\), we obtain a transducer \(T\) such that \(h(T)\) does not accept the edit string \((a/a)(a/b)(b/b)\) –
hence \( T \) is not saturated. A new construction for saturated transducers for the concatenation operation is presented in the following.

For any two edit strings \( f \) and \( g \) of the form

\[
f = (x_1/e) \cdots (x_n/e) \quad \text{and} \quad g = (e/y_1) \cdots (e/y_n),
\]

where each \( x_i \) is in \( X \cup \{e\} \) and each \( y_i \) is in \( Y \cup \{e\} \), we define the left and right merge operations ‘\(<\)’ and ‘\(>\)’ such that

\[
f \triangleleft g = g \triangleright f = (x_1/y_1) \cdots (x_n/y_n).
\]

**Lemma 2.** 1. For any edit strings \( f \) and \( g \) of the form shown above, we have that \( h(f \triangleleft g) = h(g \triangleright f) = h(fg) \). Also, \( (e/e) = (e/e) \triangleleft (e/e) = (e/e) \triangleright (e/e) \).

2. If \( \tau_1 \) and \( \tau_2 \) are transductions and \( e \) is any edit string with \( h(e) \in \tau_1 \cdot \tau_2 \), then \( e \) can be written as \( e_1 e_2 e_3 \) such that \( e_2 \) is of the form \( f_2 \triangleleft g_2 \), or \( f_2 \triangleright g_2 \), and \( h(e_1 f_2) \in \tau_1 \) and \( h(g_2 e_3) \in \tau_2 \).

**Proof.** The first statement follows easily from the definitions of the operations \( \triangleleft \) and \( \triangleright \). For the second statement, first note that there are \((x_1,y_1)\) in \( \tau_1 \) and \((x_2,y_2)\) in \( \tau_2 \) such that \( h(e) = (x_1 x_2, y_1 y_2) \). Notation wise, if \( \alpha = (u,v) \) is a pair of words, then we denote \( \pi_1(\alpha) = u \) and \( \pi_2(\alpha) = v \). We distinguish the following factors of \( e \):

- Let \( e_1 \) be the shortest prefix of \( e \) such that either \( x_1 = \pi_1(h(e_1)) \), or \( y_1 = \pi_2(h(e_1)) \).
- Let \( e_2 \) be the edit string such that \( e_1 e_2 \) is the shortest prefix of \( e \) such that either \( y_1 = \pi_2(h(e_1 e_2)) \), or \( x_1 = \pi_1(h(e_1)) \), respectively.
- Finally, let \( e_3 \) be such that \( e = e_1 e_2 e_3 \).

By looking in detail at the edit operations comprising \( e \), one can verify that there are edit strings \( f_2 \) and \( g_2 \) such that \( e_2 = f_2 \triangleright g_2 \), or \( e_2 = f_2 \triangleleft g_2 \), respectively, and \( h(e_1 f_2) \in \tau_1 \) and \( h(g_2 e_3) \in \tau_2 \), as required.

**Construction of \( T_1 \cdot T_2 \):**

**Input:** Two saturated transducers \( T_1 = (Q_1, X_1, Y_1, \Delta_1, s_1, F_1) \) and \( T_2 = (Q_2, X_2, Y_2, \Delta_2, s_2, F_2) \) in standard form. We shall assume that \( T_1 \) is already trim, that is, each state can be reached from \( s_1 \) and can reach a final state in \( F_1 \).

**Step 1:** Let \( U_{10} \) be the set of states \( p_1 \) in \( Q_1 \) such that there is a successful computation of \( T_1 \), from \( p_1 \), with label \((e,v)\), for some \( v \) in \( Y_1^* \). Let \( U_{01} \) be the set of states \( q_1 \) in \( Q_1 \) such that there is a successful computation of \( T_1 \), from \( q_1 \), with label \((u,e)\), for some \( u \) in \( X_1^* \).

**Step 2:** Define the set \( Q \) consisting of the following states.

- All states \( r_1 \) in \( Q_1 \). Such an \( r_1 \) means that the automaton \( h^{-1}(T_1 \cdot T_2) \) corresponding to the intended transducer \( T_1 \cdot T_2 \) has read an edit string \( e \)
which is also the label of some computation of \( h^{-1}(T_1) \) from \( s_1 \) to \( r_1 \). This implies that, at state \( r_1 \), the machine \( T_1' \cdot T_2 \) has read some label \( (x'_1, y'_1) \) for which there is \((x_1, y_1)\) in \([T_1]\) with \(x'_1\) and \(y'_1\) being prefixes of \(x_1\) and \(y_1\), respectively.

- All states \((q_1, q_2, 01)\) with \(q_1 \in U_{01}\) and \(q_2 \in Q_2\). Such a state means that \( h^{-1}(T_1' \cdot T_2) \) has read an edit string \( e_1 e_2\) such that \( e_2\) is of the form \( f_2 < g_2\) and there is a computation of \( h^{-1}(T_1) \) from \( s_1 \) to \( q_1 \) with label \( e_1 f_2\), and a computation of \( h^{-1}(T_2) \) from \( s_2 \) to \( q_2 \) with label \( g_2\). This implies that, at state \((q_1, q_2, 01)\), \(T_1' \cdot T_2\) has read some label of the form \((x'_1, y'_2)\) for which \(x'_1\) is a prefix of some \(x_1\) with \((x_1, y_1)\) in \([T_1]\) and \(y'_2\) is a prefix of some \(y_2\), with \((x_2, y_2)\) in \([T_2]\) for some \(x_2\). The “flag” \(01\) above reminds us that \(T_1' \cdot T_2\) has completed reading only the second component of \((x_1, y_1)\) and that no part of \(x_2\) can be read before completing \(x_1\).

- All states \((p_1, p_2, 10)\) with \(p_1 \in U_{10}\) and \(p_2 \in Q_2\). Such a state means that \( h^{-1}(T_1' \cdot T_2) \) has read an edit string \( e_1 e_2 e_3\) such that \( e_2\) is of the form \( f_2 \triangleright g_2\), and there is a computation of \( h^{-1}(T_1) \) from \( s_1 \) to \( p_1 \) with label \( e_1 f_2\), and a computation of \( h^{-1}(T_2) \) from \( s_2 \) to \( p_2 \) with label \( g_2\). This implies that, at state \((p_1, p_2, 10)\), \(T_1' \cdot T_2\) has read some label of the form \((x'_2, y'_2)\) for which \(y'_2\) is a prefix of some \(y_1\) with \((x_1, y_1)\) in \([T_1]\) and \(x'_2\) is a prefix of some \(x_2\), with \((x_2, y_2)\) in \([T_2]\) for some \(y_2\).

- All states \( r_2 \) in \(Q_2\). Such an \(r_2\) means that \( h^{-1}(T_1' \cdot T_2) \) has read an edit string \( e_1 e_2 e_3\) such that \( e_2\) is of the form \( f_2 \triangleright g_2\), or \( f_2 \triangleleft g_2\), and there is a computation of \( h^{-1}(T_1) \) from \( s_1 \) to \( F_1\) with label \( e_1 f_2\), and a computation of \( h^{-1}(T_2) \) from \( s_2 \) to \( r_2 \) with label \( g_2 e_3\). This implies that, at state \(r_2\), \(T_1' \cdot T_2\) has read some label of the form \((x_2, x'_2, y_2, y'_2)\) for which \((x_1, y_1)\) is in \([T_1]\) and there is \((x_2, y_2)\) in \([T_2]\) such that \(x'_2\) and \(y'_2\) are prefixes of \(x_2\) and \(y_2\), respectively.

### Step 3: Define the set \( \Delta \) consisting of the transitions of \(T_1' \cdot T_2\) in such a way that the meaning of the states in \(Q\) is preserved. More specifically we have that \( \Delta \) consists of the following transitions.

- All transitions in \( \Delta_1\).
- All transitions of the forms \(\langle p_1, \epsilon, \epsilon, (p_1, s_2, 10) \rangle\), with \(p_1 \in U_{10}\), and \(\langle q_1, \epsilon, \epsilon, (q_1, s_2, 01) \rangle\), with \(q_1 \in U_{01}\).
- All transitions of the form \(\langle (p_1, p_2, 10), a, b, (p'_1, p'_2, 10) \rangle\), with \(p_1, p'_1 \in U_{10}\), \(p_2, p'_2 \in Q_2\), and \(\langle p_1, \epsilon, b, p'_1 \rangle\) in \( \Delta_1\), and \(\langle p_2, a, \epsilon, p'_2 \rangle\) in \( \Delta_2\).
- All transitions of the form \(\langle (q_1, q_2, 01), a, b, (q'_1, q'_2, 01) \rangle\), with \(q_1, q'_1 \in U_{01}\), \(q_2, q'_2 \in Q_2\), and \(\langle q_1, a, \epsilon, q'_1 \rangle\) in \( \Delta_1\), and \(\langle q_2, \epsilon, b, q'_2 \rangle\) in \( \Delta_2\).
- All transitions of the forms \(\langle (p_1, p_2, 10), \epsilon, \epsilon, p_2 \rangle\), with \(p_1 \in F_1\) and \(p_2 \in Q_2\), and \(\langle (q_1, q_2, 01), \epsilon, \epsilon, q_2 \rangle\), with \(q_1 \in F_1\) and \(q_2 \in Q_2\).
- All transitions in \( \Delta_2\).

### Output: The transducer \(T_1' \cdot T_2 = (Q, X_1 \cup X_2, Y_1 \cup Y_2, \Delta, s_1, F_2)\).

### Theorem 1. For any saturated transducers \(T_1\) and \(T_2\), the transducer \(T_1' \cdot T_2\) is saturated and realizes the transduction \([T_1'] \cdot [T_2]\). Moreover, \(\text{size}(T_1' \cdot T_2) = O(\text{size}(T_1) \cdot \text{size}(T_2))\).
Proof. The statement about the size of $T_1 \cdot T_2$ follows easily from its construction. For the first statement, it is sufficient to prove that $|T_1 \cdot T_2| \leq |T_1| \cdot |T_2|$ and that, for any edit string $e$ with $h(e) \in [T_1] \cdot [T_2]$, we have that $e \in h^{-1}(T_1 \cdot T_2)$. Let $(x, y)$ be any element in $[T_1] \cdot [T_2]$. There is a computation of $T_1 \cdot T_2$ with label $(x, y)$ and a corresponding computation of $h^{-1}(T_1 \cdot T_2)$ with some label $e$, with $h(e) = (x, y)$. By the definition of the final states of $T_1 \cdot T_2$, $e$ is of the form $e_1e_2e_3$ with $e_2 = f_2 \cdot g_2$ -- the case $e_2 = f_2 \cdot g_2$ is symmetric -- and $h^{-1}(T_1)$ accepts $e_1f_2$, and $h^{-1}(T_2)$ accepts $g_2e_3$. This implies that

$$(x, y) = h(e_1)h(f_2g_2)h(e_3) = h(e_1)h(g_2e_3) \in [T_1] \cdot [T_2].$$

Now consider any edit string $e$ such that $h(e) \in [T_1] \cdot [T_2]$. We shall use the notation in the preceding construction. The string $e$ can be written as $e_1e_2e_3$ such that $e_2$ is of the form $f_2 \cdot g_2$ -- the case $f_2 \cdot g_2$ is symmetric -- and $h(e_1,f_2) \in [T_1]$ and $h(g_2e_3) \in [T_2]$. This implies that there is a computation of $h^{-1}(T_1)$ from $s_1$ to some $q_1 \in U_{01}$ with label $e_1$, and a computation of $h^{-1}(T_2)$ from $q_1$ to some state $q'_1 \in U_{01}$ with label $f_2$. Moreover there is a computation of $h^{-1}(T_2)$ from $s_2$ to some $q_2 \in Q_2$ with label $g_2$, and a computation of $h^{-1}(T_2)$ from $q_2$ to some state $q'_2 \in F_2$ with label $e_3$. Using the transitions of $T_1 \cdot T_2$ one can verify that there is a successful computation of $h^{-1}(T_1 \cdot T_2)$ with label $e_1e_2e_3$, as required.

We close this section by noting that the construction of $T_1 \cdot T_2$ can be carried out in time $O(\text{size}(T_1) \cdot \text{size}(T_2))$. This is clear in steps 2 and 3. In Step 3, the computation of $U_{00}$ can be done in time $O(\text{size}(T_1))$ as follows. Let $G_1$ be the (directed) graph obtained by adding in the graph of $T_1$ a new state $N$ and $\langle \epsilon, \epsilon \rangle$-transitions from all final states of $T_2$ to $N$. Consider the graph $G_2$ obtained if we keep only the transitions of $G_1$ of the form $\langle \epsilon, a \rangle$ and reverse the direction of these transitions. Then the set $U_{01}$ consists of all the states in $G_2$, other than $N$, that can be reached from the state $N$. This traversal can be performed in time linear with respect to the size of $G_2$. The computation of $U_{01}$ is analogous.

4 Saturation and Recognizability

Let us recall a few facts mentioned in the preliminaries of this paper. We know that a recognizable subset of $X^* \times Y^*$ is rational, therefore there exists a finite transducer which realizes it. The opposite does not hold: there exist quite simple rational transductions which are not recognizable, for example the identity over $X^*$. We also know a characterization of recognizable transductions as finite unions of blocks. There exist another two definitions of recognizable sets in arbitrary monoids: a morphism based definition (see for example [Pin, 1997]) and a definition based on monoid actions on finite sets(for an extensive discussion on the topic, consult [Sakarovitch, 2003, §II.2]). We recall here the later one.

Let $Q$ be a finite set and $(M, \cdot, 1_M)$ an arbitrary monoid. An action of $M$ on $Q$ is a function $f : M \times Q \rightarrow Q$ which satisfies the following two properties: $f(q, 1_M) = q$ and $f(f(q,m),m') = f(q, mm')$, for all $q \in Q$ and $m, m' \in M$. A
subset \(D\) of \(M\) is recognizable if there exists such finite set \(Q\) and action \(f\), and there exists \(F \subseteq Q\) and \(q \in Q\) such that \(D = \{m \in M / f(q, m) \in F\}\).

In this section we give a fourth characterization of recognizable transductions by proving that the appropriate machines which realize them are saturated transducers. We start by giving two useful constructions.

**Construction #1**

**input:** We are given a saturated transducer \(T\), which we put in standard form, if it is not already.

**step 1:** We construct the finite automaton \(h^{-1}(T)\) by interpreting the labels of transitions of \(T\) as edit operation symbols. The automaton \(h^{-1}(T)\) is over the alphabet \(E\) (and has been described in details at the beginning of Section 3).

**step 2:** We determinize and minimize the automaton \(h^{-1}(T)\), obtaining a minimal, complete DFA \(B\). Denote \(B = (Q, E, \delta, q_0, F)\).

**step 3:** For each state \(q\) of \(B\) we construct a corresponding automaton \(C_q\) as following:
- (a) \(C_q\) has the same set of states as \(B\), the same initial state, and it has \(\{q\}\) as the set of final states;
- (b) for each transition in \(B\) of type \((p, (a/e), p')\) with \(a \in X\) we assign a transition \((p, a, p')\) in \(C_q\).

**step 4:** For each state \(q\) of \(B\) we construct a corresponding automaton \(D_q\) as following:
- (a) \(D_q\) has the same set of states as \(B\), the same set of final states, and it has \(q\) as initial state;
- (b) for each transition in \(B\) of type \((p, (\epsilon/b), p')\) with \(b \in Y\) we assign a transition \((p, b, p')\) in \(D_q\).

**output:** Let \(Q' := \{q \in Q / \mathcal{L}(C_q) \neq \emptyset\text{ and } \mathcal{L}(D_q) \neq \emptyset\}\). The algorithm ends by delivering \(\{C_q, D_q\}_{q \in Q'}\).

**Lemma 3.** The above construction ensures the following properties:

(i) \(|T| = \bigcup_{q \in Q'} \mathcal{L}(C_q) \times \mathcal{L}(D_q)\).

(ii) The languages \(\{\mathcal{L}(C_q)\}_{q \in Q'}\) are disjoint. The languages \(\{\mathcal{L}(D_q)\}_{q \in Q'}\) are distinct.

(iii) The transition function of the automaton \(h(B)\) can be extended to a monoid action of \(X^* \times Y^*\) on \(Q\).

(iv) If \(h^{-1}(T)\) is deterministic then

\[
\sum_{q \in Q'} (\text{size}(C_q) + \text{size}(D_q)) = O(\text{size}(T)^2) .
\]
Proof. We analyze each step of the above construction. The automaton \( h^{-1}(T) \) found in step 1 has the following property:

\[
\forall e \in \mathcal{L}(h^{-1}(T)), \forall e' \in E^* : \ h(e') = h(e) \Rightarrow e' \in \mathcal{L}(h^{-1}(T)),
\]

(1)
given by the saturation of \( T \). In other words, if \( h^{-1}(T) \) accepts some edit string \( e \), it will necessarily accept all edit strings which express the same word transformation as \( e \). In algebraic terms, we say that the congruence induced by \( h \) — let us call it \( \equiv_h \) — saturates \( \mathcal{L}(h^{-1}(T)) \).

Since \( B \) found at step 2 is the minimization of \( h^{-1}(T) \), it will preserve the above property. The automaton \( B \) has the following additional property:

\[
\forall e, e' \in E^* : \ h(e) = h(e') \Rightarrow \delta(q_0, e) = \delta(q_0, e'),
\]
in other words:

\[
\equiv_h \subseteq \equiv_{\mathcal{L}(B)},
\]

(2)
where by \( \equiv_{\mathcal{L}(B)} \) we denoted the Myhill-Nerode equivalence of \( \mathcal{L}(B) \). We justify this property as following:
Let \( h(e) = h(e') \) and denote \( p = \delta(q_0, e) \) and \( q = \delta(q_0, e') \). Assume by contradiction that \( p \neq q \). Then, since \( B \) is minimal, it follows that there exists \( e'' \in E^* \) such that \( \delta(p, e'') \) is a final state in \( B \) and \( \delta(q, e'') \) is not. But then, \( ee'' \in L \) and is easy to see that \( h(ee'') = h(e'e'') \). By the property expressed in relation (1) we infer that \( e'e'' \) must be accepted - a contradiction.

Let a pair of words \((u, v) \in X^* \times Y^* \) be accepted by the given transducer \( T \). Consider that \( u = u_1u_2...u_m, v = v_1v_2...v_n, \) with \( u_1, ..., u_m \in X \) and \( v_1, ..., v_n \in Y \). An edit string which transforms \( u \) into \( v \) is

\[
e = (u_1/e)...(u_m/e)(e/v_1)...(e/v_n),
\]
and denote \( e = e_1e_2, \) with \( e_1 = (u_1/e)...(u_m/e). \) Since \((u, v) \in \mathcal{T} \), we have that \( e \in \mathcal{L}(B), \) hence \( \delta(q_0, e_1e_2) \in F \) in \( B \). Denote \( q = \delta(q_0, e_1) \) and observe that \( u \in \mathcal{L}(C_q) \) and \( v \in \mathcal{L}(D_q). \) Since the reciprocal also holds, we have that

\[
(u, v) \in \mathcal{T} \iff u \in C_q \text{ and } v \in D_q \text{ for some } q \in Q,
\]
which proves Property (i) of the lemma.

By the fact that \( B \) is deterministic, it follows that \( \{C_q\}_{q \in Q} \) are disjoint. For the second part of Property (ii), we use yet another property of the automaton \( B \), that is,

\[
\forall q \in Q, \forall e, e' \in E^* \text{ such that } h(e) = h(e') : \ \delta(q, e) \in F \Rightarrow \delta(q, e') \in F,
\]
(3)
which can easily be verified (invoking the saturation of \( T \)). Since \( B \) is minimal, and by the above property, we conclude that \( \mathcal{L}(D_p) \neq \mathcal{L}(D_q) \) for any two distinct states \( p, q \in Q, \) as long as either \( \mathcal{L}(D_p) \) or \( \mathcal{L}(D_q) \) is not empty. This completes the proof of Property (ii).

Let us consider the transducer \( h(B) \), which is obtained from \( B \) by replacing the transition labels(symbols) of the form \( x/y \) with the corresponding pairs...
$(x, y)$. Clearly, $|T| = |h(B)|$. If we denote $f$ to be the transition function of $h(B)$ (it is a partial function due to the determinism of $B$) it is enough to show that we can extend $f$ to $X^* \times Y^*$ such that it verifies the properties of an action. For any $(u, v) \in X^* \times Y^*$, let $e_{u,v}$ be a chosen edit string such that $h(e_{u,v}) = (u, v)$. We set $f(p, (u, v)) := \delta(p, e_{u,v})$ and $f(p, (\epsilon, \epsilon)) := p$, for all states in $Q$. It can readily be checked that the definition is independent of the choice of $e_{u,v}$, that is, $f$ is a function

$$f : (X^* \times Y^*) \times Q \to Q,$$

and that

1. $f(p, (\epsilon, \epsilon)) = p, \forall p \in Q$,
2. $f(f(p_1, (u_1, v_1)), (u_2, v_2)) = f(p, (u_1 u_2, v_1 v_2))$.

Finally we have that $(u, v) \in |T| \Leftrightarrow f(p_0, (u, v))$ is a final state in $h(B)$ (where $p_0$ is the initial state of $h(B)$). $\square$

**Remark 1.** Notice that Property (i) of the above lemma does not depend on the minimality and completeness of $B$. Indeed, if we eliminate step 2 of the above construction, and we consider $h^{-1}(T)$ instead of $B$ in the subsequent steps, we would still obtain Property (i) of the lemma.

**Corollary 1.**

$$\text{Sat}(X^* \times Y^*) \subseteq \text{Rec}(X^* \times Y^*) .$$

**Proof.** By Mezei’s characterization of recognizable transductions, we observe that the transduction realized by a saturated transducer is a finite union of blocks, hence it is recognizable. $\square$

We now turn our attention to a possible reciprocal of the above corollary, and we are aiming, as usual, at a constructive proof.

**Construction #2**

**input:** We have a transduction $\tau \in \text{Rec}(X^* \times Y^*)$ effectively given as a tuple $(A_1, B_1, \ldots, A_n, B_n)$ of finite automata. That is, we know that

$$\tau = \bigcup_{i=1}^{n} \mathcal{L}(A_i) \times \mathcal{L}(B_i) .$$

**step 1:** For each $i \in \{1, \ldots, n\}$ we construct a saturated transducer $T_i$ such that $|T_i| = \mathcal{L}(A_i) \times \mathcal{L}(B_i)$ (the construction has been presented in Section 3).

**step 2:** Since all $T_i$ are saturated, we construct in $n - 1$ iterations the transducer $T^\cup = T_1 \cup \ldots \cup T_n$, which realizes the transduction $|T_1| \cup \ldots \cup |T_n|$ (this construction has also been presented in Section 3).
output: The algorithm delivers $T^\cup$.

**Lemma 4.** The above construction ensures that

$$|T^\cup| = \tau .$$

Moreover, $T^\cup$ is saturated and $\text{size}(T^\cup) = \sum_{i=1}^{n} (\text{size}(A_i) \cdot \text{size}(B_i))$.

**Proof.** The correctness and finiteness of each step has been proven in Lemma 1. \qed

**Remark 2.** The saturation of $T$ is not a trivial property, since there may exist non-saturated transducers which realize recognizable transductions, as the following example shows.

**Example 3.** Consider the transduction $\tau$, over $\{0, 1\}$ and $\{a\}$, which contains all pairs $(u,v)$ with the value of $u$, as a binary word, being odd and $v$ an arbitrary word over $\{a\}^*$. Both transducers in Fig. 3 are in standard form and realize $\tau$; however, only the transducer in Fig. 3 (b) is saturated.

![Diagram](image)

**Fig. 3.** Equivalent non-saturated and saturated transducers.

**Corollary 2.**

$$\text{Rec}(X^* \times Y^*) \subseteq \text{Sat}(X^* \times Y^*) .$$

**Proof.** It is a consequence of the previous lemma. \qed

**Remark 3.** This corollary can also be proven, non-constructively, by using the closure properties of recognizable sets, as following.

**Proof.** Let $\tau$ be a recognizable transduction and consider the edit morphism over $X$ and $Y$,  

$$h : E^* \rightarrow X^* \times Y^* .$$
Since $h$ is a morphism and $\tau$ is recognizable in $X^* \times Y^*$ we have that $h^{-1}(\tau)$ is recognizable in $E^*$ (by the fact that recognizable sets are closed under inverse morphism). Then, by Kleene’s theorem we have that $h^{-1}(\tau)$ is a regular language, hence there exists a finite automaton $A$ over $E$ which accepts $h^{-1}(\tau)$. Assume that $A$ is a complete DFA. It now suffices to observe that the transducer $h(A)$ is saturated, in standard form, and it realizes $\tau$. \qed

Summing up, we have the following characterization of recognizable transductions.

**Theorem 2.** A transduction is recognizable if and only if it is realized by a saturated transducer.

**Proof.** It is a direct consequence of Corollary 1 and Corollary 2. Notice that the previous two constructions give a constructive proof of this theorem. \qed

**Remark 4.** There is an elegant proof for Lemma 2 using Mezei’s theorem. Indeed, if $T_1$ and $T_2$ are saturated transducers, then by the theorem we have that $|T_1|$ and $|T_2|$ are recognizable, hence by Mezei’s theorem we have that

$$|T_1| = \bigcup_{i=1}^{m} A_i \times B_i \text{ and } |T_2| = \bigcup_{j=1}^{n} C_j \times D_j ,$$

where we expressed the transductions as union of blocks. Then it suffices to observe that

$$|T_1 \circ T_2| = \bigcup_{1 \leq i \leq m, 1 \leq j \leq n} G_{i,j} , \text{ with } G_{i,j} = \begin{cases} \emptyset, & \text{if } B_i \cap C_j = \emptyset; \\ A_i \times D_j, & \text{otherwise.} \end{cases}$$

Consequently, $|T_1 \circ T_2|$ is recognizable, therefore realizable by a saturated transducer $T_1 \circ T_2$, which can effectively be constructed. Notice that $T_1 \circ T_2$ may have a structure different than that of $T_2 \circ T_1$ which was proposed in Lemma 1.

**Remark 5.** We have seen in Theorem 1 that given two saturated transducers $T_1$ and $T_2$, one can construct a size $O(\text{size}(T_1) \cdot \text{size}(T_2))$ transducer $T_1 \cdot T_2$ which realizes $|T_1| \cdot |T_2|$. That construction can stand as an alternative proof that recognizable transductions are closed under concatenation (the other proof makes use of Mezei’s theorem).

**Remark 6.** We can now explain why in Section 3 we have not mentioned anything about the “star” operation on a saturated transducer. The reason is that saturated transductions are not closed under iteration, as the following classical example shows: $\{(a, b)\}$ is a saturated transduction, being finite; however, $\{(a, b)\}^*$ is not recognizable, hence can not be realized by a saturated transducer.

**Remark 7.** It is worth noticing that, given a finite transducer $T$ over alphabets with at least two letters, it is undecidable whether there exists a saturated transducer equivalent with $T$. Indeed, this follows from the known fact that is undecidable whether a finite transducer over alphabets with at least two letters realizes a recognizable transduction.
5 Edit Distance and the non-Recognizability of \((L \times L)\neq\)

Edit strings and edit languages constitute natural tools for dealing with problems related to the edit distance between words and languages. In this context, the weight \(\text{weight}(e)\) of an edit string

\[ e = (x_1/y_1) \cdots (x_n/y_n) \]

is the number of edit operations \((x_i/y_i)\) in \(e\) with \(x_i \neq y_i\). For example, the weight of the edit string \(f\) in Section 2 is 2. Then the edit distance between two words \(u\) and \(v\) is the minimum of the weights of the edit strings transforming \(u\) into \(v\), that is,

\[ \text{dist}(u, v) = \min\{\text{weight}(e)/e \in h^{-1}(\{(u, v)\})\}. \]

If we construct automata \(A_u\) and \(A_v\), accepting \(\{u\}\) and \(\{v\}\), respectively, then the saturated transducer \(A_u \times A_v\) accepts all edit strings \(e\) with \(e \in h^{-1}(\{(u, v)\})\). Hence, the quantity \(\text{dist}(u, v)\) is the weight of the smallest-weight path (computation) in the graph corresponding to \(A_u \times A_v\) - here the weights on the transitions are in \(\{0, 1\}\) such that the weight of a transition \((p, (x/y), q)\) is 1 if and only if \(x \neq y\). This simple idea can be generalized for any pair of automata \(A_1\) and \(A_2\) and for more general types of distances - see [Mohri, 2003] and [Kari et al., 2003] for details.

The problem of computing the (inner) edit distance of a language \(L\) is more difficult, however. This quantity is the minimum edit distance between any pair of distinct words of \(L\). Suppose that \(A\) is an automaton accepting \(L\). The difficulty here lies in the fact that the saturated transducer \(A \times A\) accepts edit strings \(e\) corresponding to pairs of equal words. Therefore, one would like to have a saturated transducer for the transduction

\[ (L \times L)_{\neq} = \{(u, v)/u, v \in L\text{ and } u \neq v\}. \]

Although one can construct an ordinary transducer for this transduction, we show next that there is no saturated transducer for this transduction, that is, \((L \times L)_{\neq}\) is not recognizable when \(L\) is infinite. For the sake of completeness we mention that the problem of computing the inner edit distance is solved in [Konstantinidis, 2005] by observing that (i) this quantity is always realized by two words differing at some position bounded by \(j_A\), for some index that depends on the automaton \(A\) accepting \(L\); and (ii) for any index \(j\), there is a transducer \(T_j\) (which turns to be saturated, in our terminology) realizing all pairs of words that differ at position \(j\).

Given an arbitrary set \(P\), by \((P \times P)_{\neq}\) we understand the set of all pairs of different elements of \(P\). In other words, \((P \times P)_{\neq} = (P \times P) \setminus \{d(P)\} P \times P) \setminus \{d(P)\}.

**Proposition 1.** Let \(P\) be an arbitrary, infinite set. The set equation

\[ (P \times P)_{\neq} = \bigcup_{i=1}^{n}\ X_i \times Y_i \]

has no solution \((n, \{X_i, Y_i\})_{i=1}^{n}\).
Proof. Assume, by contradiction, that there exists \((n, \{X_i, Y_i\}_{i=1}^n)\) – a solution of the above equation. Notice first that necessarily \(X_i \cap Y_i = \emptyset\) for all \(i \in \{1, \ldots, n\}\). Since \(P\) is infinite, there exist \(2^{n+1}\) different elements in \(P\). Denote by \(U_1 := \{u_1, \ldots, u_{2^{n+1}}\}\) a set of such elements.

Consider the triplet \(U_1, X_1\) and \(Y_1\). We can write

\[
U_1 = (U_1 \cap X_1) \cup (U_1 \cap Y_1) \cup (U_1 \setminus (X_1 \cup Y_1)),
\]

since \(X_1\) and \(Y_1\) are disjoint. Let us assume, without loss of generality that \(|U_1 \cap X_1| \geq |U_1 \cap Y_1|\), and let us denote \(U_2 := U_1 \setminus Y_1\).

We first prove that \(U_2\) has at least \(2^n\) elements. We have that \(|U_1 \cap X_1| + |U_1 \cap Y_1| \leq 2^{n+1}\) and that \(|U_1 \cap X_1| \geq |U_1 \cap Y_1|\). This implies that \(|U_1 \cap Y_1| \leq 2^n\), by the fact that \(U_1 \cap X_1\) and \(U_1 \cap Y_1\) are disjoint. Then clearly \(|U_2 \setminus Y_1| \geq 2^n\), hence \(|U_2| \geq 2^n\). We may also observe that the pairs of different elements in \(U_2\) can not appear in \(X_1 \times Y_1\). Indeed, we can not have \((u, v) \in X_1 \times Y_1\) and \(u, v \in U_2\), since \(U_2 = U_1 \setminus Y_1\).

We repeat the above argument for the triplet \(U_2, X_2\) and \(Y_2\). We obtain a set \(U_3 \subseteq U_1\) with \(|U_3| \geq 2^{n-1}\) and no pair of elements in \(U_3\) can be found in \(X_2 \times Y_2\).

Then, we repeat this argument till we obtain \(U_{n+1} \subseteq U_2\) with \(|U_{n+1}| \geq 2\) and no pair of elements in \(U_{n+1}\) can be found in \(X_n \times Y_n\).

Take two different elements \(u, v \in U_{n+1}\). Since we have \(U_{n+1} \subseteq U_n \subseteq \ldots \subseteq U_1\), we conclude that the pair \((u, v)\) does not belong to any \(X_i \times Y_i\), for \(1 \leq i \leq n\).

But this contradicts the fact that \(U_1 \subseteq P\). \(\square\)

Corollary 3. Let \(L \in X^*\) be an infinite regular language. The transduction \((L \times L)_\tau\) can not be realized by a saturated transducer over \(X\).

Proof. In order to have a saturated transducer for \((L \times L)_\tau\), this set must be recognizable, by Theorem 2. However, Proposition 1 shows that it can not be written as a finite union of blocks, hence it is not recognizable, by Mezei’s characterization. \(\square\)

6 Final Comments and Future Work

In this paper we have achieved the following. We have revealed the relation between edit languages, recognizable transductions and saturated transducers. We have shown that operations with saturated transducers can efficiently be implemented, and we outlined methods to construct and manipulate saturated transducers. We have shown how one can use saturated transducers for computing the edit distance between words and languages. Finally, we have studied situations when our framework can not be used, due to the non-recognizability of various rational relations.

It is worth noticing that our entire framework still holds when is restricted to the use of only two edit operations: insertion and deletion (for this case,
one defines “restricted saturated transducers”). This restriction may be of importance in applications where only these two edit operations are of interest ([Levenshtein, 1966]).

Left for further analysis are a few matters which have not been tackled yet. For example, it is worth investigating algorithms to efficiently compute saturated transducers for given finite transductions; in particular, for finite identities.

It is interesting to notice that the notion of minimal saturated transducer for a recognizable transduction makes sense, since it is given by the minimal corresponding DFA over the edit alphabet. Size-complexity matters may be investigated in this aspect.

Finally, we have left for study the comparison of two representations (characterizations) of recognizable transductions: one using saturated transducers the other using tuples of automata.

References


