Some Remarks on Regular Factorizations

STAVROS KONSTANTINIDIS
Department of Mathematics and Computing Science
Saint Mary's University
Halifax, Nova Scotia, B3H 3C3
CANADA
s.konstantinidis@stmarys.ca

Abstract: - The concept of word factorization constitutes an important tool in various areas of formal languages such as theory of codes and word combinatorics. Recently, a paper by Karhumäki, Plandowski, and Rytter [4] was devoted entirely to algorithmic problems of word factorizations. In the present paper we define certain automata corresponding to regular factorizations that constitute the main tools for answering algorithmic questions related to such factorizations. In particular, we revisit some of the problems solved in [4] and give an answer to a question raised in [4]. Moreover, we consider a few additional problems related to factorizations. Our presentation provides *explicit* constructions and time complexity requirements in the proposed algorithms.

Key-Words: - automata, transducers, word factorizations, formal languages.

Research partially supported by Grant OGP220259 of NSERC Canada

1. Introduction

The concept of word factorization constitutes an important tool in various areas of formal languages such as theory of codes and word combinatorics [2], [6]. Recently, the paper [4] was devoted entirely to algorithmic problems of word factorizations. In the present paper we define certain automata corresponding to regular factorizations that constitute the main tools for answering algorithmic questions related to such factorizations. In particular, we revisit some of the problems solved in [4] and give an answer to a question raised in [4]. Moreover, we consider a few additional problems related to factorizations. Our presentation provides *explicit* constructions and time complexity requirements in the proposed algorithms.

The paper is organized as follows. In the next section we recall the definition of generalized factorization and give a notation about finite automata and transducers. In Section 3 we define certain automata associated to a given regular factorization. In Section 4 we show how to answer some of the problems considered in [4] using the automata defined in Section 3. In particular, we give an answer to the problem of computing minimal and maximal factorizations of a given word in polynomial time. Finally, in Section 5 we consider the problem of computing labeled factorizations of a word as well as factorizations of a word satisfying certain conditions on the number of the permitted factors.

2. Basic Notions and Notation

An *alphabet* X is a finite and nonempty set of symbols. A *word* (over X) is a finite sequence of symbols from X. A word w can be written as $a_1 \cdots a_n$, where a_i is in X for all i. In this case, n is the length of the word w, and is denoted by $|w|$. The set of all words is denoted by X^*. This set includes the empty word λ whose length is zero. If w_1 and w_2 are two words then $w_1 w_2$ denotes the concatenation of w_1 and w_2. We refer the reader to [7] for general information on formal languages.

A *(finite) automaton with empty transitions* (a λ-*NFA* for short) is a quintuple $\mathcal{B} = (X, Q, s, F, E)$ 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, s is the initial state in Q, and F 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* $\mathcal{B}$ is a word ω of the form $p_0 x_1 p_1 \cdots x_n p_n$ such that each factor $p_{i-1} x_i p_i$ of ω is in E. The language accepted by the automaton $\mathcal{B}$ is denoted by $L(\mathcal{B})$.

The automaton is called *trim* if, for every state q in Q, there is a path from s to q and a path from q to a final state in F (when F is not empty). The size of $\mathcal{B}$, denoted by $|\mathcal{B}|$, is equal to $|Q| + |E|$. If $\mathcal{B}$ is trim then $|Q| \le |E| + 1$ and, therefore, $|\mathcal{B}| = \Theta(|E|)$. The automaton $\mathcal{B}$ 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'', $q' = q''$.

A *(finite) transducer* $\mathcal{T}$ is a sextuple (X, Y, Q, s, F, E) such that Y is the output alphabet and the components X, Q, s, F, 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. For a word w, $\mathcal{T}(w)$ is the set of all output words when w is used as input to the transducer. The transducer is called *trim* if, for every state $q \in Q$, there is a path from s to q and a path from q to a final state (when $F \neq \emptyset$).

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 continue with a few concepts related to factorizations from [4]. A *(generalized) factorization* $\mathcal{F}$ is a tuple $(L_0, L_1, \ldots, L_k)$ of languages, with $k \ge 1$. We write $I_{\mathcal{F}}$ for the set of indices $\{1, \ldots, k\}$. The languages $L_1, \ldots, L_k$ are subsets of Σ^*, where Σ is an alphabet, and L_0 is a subset of $I_{\mathcal{F}}^*$ – the set $I_{\mathcal{F}}$ is considered to be an alphabet. In the sequel, we assume that Σ is fixed, but arbitrary, and that Σ is disjoint from $I_{\mathcal{F}} \cup \{(,)\}$. An *$\mathcal{F}$-factorization of a word* w is a word φ in $(\Sigma \cup \{(,)\})^*$ of the form $(w_1) \cdots (w_n)$ such that each w_i is in L_{t_i}, for some $t_i \in I_{\mathcal{F}}$, and $t_1 \cdots t_n$ is in L_0. In this case, each w_i is called a *factor of* φ. Note that a word w might have infinitely many $\mathcal{F}$-factorizations. This can happen for instance when one of the languages L_i contains the empty word. For example, let $\mathcal{F} = (1^*2, a^*, b^*)$ be a factorization, and let $w = bb$. Then $(\lambda)^n(bb)$ is an $\mathcal{F}$-factorization of w for all $n \ge 0$. An $\mathcal{F}$-factorization φ of a word is *λ-free* if every factor of φ is nonempty. A *labeled $\mathcal{F}$-factorization of* w is a word ψ of the form $t_1 w_1 t_1 \cdots t_n w_n t_n$ such that each w_i is in L_{t_i} and $t_1 \cdots t_n$ is in L_0.

The factorization $\mathcal{F}$ is *regular* if every language L_i is a regular language. In this case, $\mathcal{F}$ can be represented by a tuple $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k)$ of NFAs. In the sequel, we assume that $\mathcal{A}$ is fixed, but arbitrary, and we write $\mathcal{F}_{\mathcal{A}}$ for the regular factorization represented by $\mathcal{A}$. Moreover, we write $I_{\mathcal{A}}$ instead of $I_{\mathcal{F}_{\mathcal{A}}}$.

3. Automata for Factorizations

For each regular factorization $\mathcal{F}_{\mathcal{A}}$ we shall define three finite automata that can be used to answer questions about $\mathcal{F}_{\mathcal{A}}$. Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k)$ be an NFA representation of a factorization.

The automaton $\alpha_{\mathrm{f}}(\mathcal{A})$ includes the states of the automaton $\mathcal{A}_0$ and a copy of the automaton $\mathcal{A}_i$ for every transition qiq' of $\mathcal{A}_0$. The state q is connected to the start state s_i of $\mathcal{A}_i$ with label (, and every final state of $\mathcal{A}_i$ is connected to q' with label). Let E_0 be the set of transitions of $\mathcal{A}_0$. Then, for each transition $qiq' \in E_0$, the automaton $\alpha_{\mathrm{f}}(\mathcal{A})$ includes the transitions $q(s_i$ and $f)q'$, for every final state f of $\mathcal{A}_i$, and new copies of all the transitions of $\mathcal{A}_i$. By this construction, it follows that the language accepted by $\alpha_{\mathrm{f}}(\mathcal{A})$ consists of all words of the form $(w_1)\cdots(w_n)$ such that each w_i is in $L(\mathcal{A}_{t_i})$, for some index $t_i \in I_{\mathcal{A}}$, and $t_1 t_2 \cdots t_n \in L(\mathcal{A}_0)$. The size of $\alpha_{\mathrm{f}}(\mathcal{A})$ is of order $\sum_{qiq' \in E_0} |\mathcal{A}_i|$, which is bounded by $|\mathcal{A}_0||\mathcal{A}_{\max}|$, where $\mathcal{A}_{\max}$ is an automaton of $\mathcal{A}$ with maximal size.

The automaton $\alpha_\ell(\mathcal{A})$ differs from $\alpha_{\mathrm{f}}(\mathcal{A})$ in that it represents all the labeled factorizations of words as permitted by $\mathcal{F}_{\mathcal{A}}$. As before, $\alpha_\ell(\mathcal{A})$ includes the states of the automaton $\mathcal{A}_0$ and a copy of the automaton $\mathcal{A}_i$ for every transition qiq' of $\mathcal{A}_0$, such that q is connected to the start state of $\mathcal{A}_i$ with label i, and every final state of $\mathcal{A}_i$ is connected to q' with label i. By this construction, it follows that the language accepted by $\alpha_\ell(\mathcal{A})$ consists of all words of the form $t_1 w_1 t_1 \cdots t_n w_n t_n$ such that each t_i is in $I_{\mathcal{A}}$, and each w_i is in $L(\mathcal{A}_{t_i})$, and $t_1 t_2 \cdots t_n \in L(\mathcal{A}_0)$. It follows that $|\alpha_\ell(\mathcal{A})| = |\alpha_{\mathrm{f}}(\mathcal{A})|$.

Finally, the automaton $\alpha(\mathcal{A})$ obtains from $\alpha_{\mathrm{f}}(\mathcal{A})$ by replacing each transition of the form $q(q'$ or $q)q'$ with $q\lambda q'$. The language accepted by $\alpha(\mathcal{A})$ consists of all words $w_1 \cdots w_n$ such that each w_i is in $L(\mathcal{A}_{t_i})$, for some $t_i \in I_{\mathcal{A}}$, and $t_1 t_2 \cdots t_n \in L(\mathcal{A}_0)$. In other words, $L(\alpha(\mathcal{A}))$ consists of all the words that admit at least one $\mathcal{F}_{\mathcal{A}}$-factorization. It follows again that $|\alpha(\mathcal{A})| = |\alpha_{\mathrm{f}}(\mathcal{A})|$.

4. Results Pertaining to [4]

We address now the following problems using the tools defined in the previous section.

1. Given a regular factorization $\mathcal{F}_{\mathcal{A}}$ and a regular language L, compute the set of all words in L that admit at least one $\mathcal{F}_{\mathcal{A}}$-factorization.
2. Given a regular factorization $\mathcal{F}_{\mathcal{A}}$ and a regular language L, compute the set of all $\mathcal{F}_{\mathcal{A}}$-factorizations of the words in L. In case L consists of a single word w the problem becomes the same as part (1) of Theorem 4.4 in [4].
3. Using the same techniques we can compute the minimal and maximal $\mathcal{F}_{\mathcal{A}}$-facto-rizations of a given word in polynomial time – see the open problems section in [4].
4. We show a polynomial time algorithm to decide whether a given $\mathcal{F}_{\mathcal{A}}$-factorization possesses the uniqueness property, using a result about transducer functionality – see also Theorem 4.3 of [4].

Before we discuss the above problems, we need the following product construction on transducers and automata [5]. If $\mathcal{T}$ is a trim transducer in standard form and $\mathcal{B}$ is a trim λ-NFA, then one can construct in time $O(|\mathcal{T}||\mathcal{B}|)$ the trim transducer $\mathcal{T} \downarrow \mathcal{B}$ such that, for all words w and z, $z \in (\mathcal{T} \downarrow \mathcal{B})(w)$ if and only if $z \in \mathcal{T}(w)$ and $w \in L(\mathcal{B})$. In other words, $\mathcal{T} \downarrow \mathcal{B}$ accepts as input only the words in $L(\mathcal{B})$ and, for each of those words, the possible outputs are exactly the same as those of the transducer $\mathcal{T}$. Note that if we replace every transition qx/yq' of $\mathcal{T} \downarrow \mathcal{B}$ with the transition qyq', we obtain a trim λ-NFA that accepts all words z such that $z \in \mathcal{T}(w)$ for some word w in $L(\mathcal{B})$. We denote this λ-NFA by $\mathcal{B}/\mathcal{T}$. Obviously, the size of $\mathcal{B}/\mathcal{T}$ is $O(|\mathcal{B}||\mathcal{T}|)$. The above construction can be used to define, for any two λ-NFAs $\mathcal{B}$ and $\mathcal{C}$, a trim λ-NFA $\mathcal{B} \cap \mathcal{C}$ such that $|\mathcal{B} \cap \mathcal{C}| = |\mathcal{C}||\mathcal{B}|$ and $L(\mathcal{B} \cap \mathcal{C}) = L(\mathcal{B}) \cap L(\mathcal{C})$. We note that the automata $\mathcal{B}/\mathcal{T}$ and $\mathcal{B} \cap \mathcal{C}$ are not new, but they are normally defined for the case where $\mathcal{B}$ and $\mathcal{C}$ are deterministic [8]. Finally we note that if $\mathcal{B}$ is an NFA and $\mathcal{T}$ contains only transitions of the form qx/yq' with $y \neq \lambda$ then the automaton $\mathcal{B}/\mathcal{T}$ is also an NFA.

For the first problem, assume that L is given by an NFA $\mathcal{B}$. Then, the required language is given by the automaton $\alpha(\mathcal{A}) \cap \mathcal{B}$ that can be constructed in time $O(|\alpha(\mathcal{A})||\mathcal{B}|)$.

For the second problem, assume again that L is given by $\mathcal{B}$. Construct a transducer $\mathcal{T}_{(,)}$ that takes as input a word w over Σ and outputs all possible factorizations of w; that is, $\mathcal{T}(w)$ consists of all words of the form $(w_1)\cdots(w_n)$ such that $w = w_1 \cdots w_n$. This transducer has two states s and q, where s is the start and the only final state, and $2+|\Sigma|$ transitions: $s\lambda/(q,\ q\lambda/)s$, and qa/aq for all a in Σ. Obviously, $|\mathcal{T}_{(,)}| = O(1)$ – recall Σ is assumed to be fixed. Then, construct the NFA $\alpha_{\mathrm{f}}(\mathcal{A}) \cap \mathcal{B}/\mathcal{T}_{(,)}$ in time $O(|\mathcal{B}||\alpha_{\mathrm{f}}(\mathcal{A})|)$. Obviously, this automaton accepts all the $\mathcal{F}_{\mathcal{A}}$-factorizations of the words in L. In case $L = \{w\}$, the automaton $\mathcal{B}_w$ can be constructed from w in time $O(|w|)$, and the question can be answered in time $O(|w||\alpha_{\mathrm{f}}(\mathcal{A})|)$.

For the third problem, consider first computing a minimal $\mathcal{F}_{\mathcal{A}}$-factorization of a given word w as well as an automaton accepting all minimal $\mathcal{F}_{\mathcal{A}}$-factorizations of w. As before, we construct the automaton $\alpha_{\mathrm{f}}(\mathcal{A}) \cap \mathcal{B}_w/\mathcal{T}_{(,)}$, which contains no empty transitions – recall each $\mathcal{A}_i$ is an NFA. Then a shortest path of the automaton from the start state to a final state would give a minimal $\mathcal{F}_{\mathcal{A}}$-factorization. Such a path can be computed in linear time, as the graph is unweighted. Let n be the length of a minimal $\mathcal{F}_{\mathcal{A}}$-factorization of w and let $\mathcal{C}_n$ be an automaton of size $O(n)$ accepting all words of length n over $\Sigma \cup \{(,)\}$. Then the automaton $(\alpha_{\mathrm{f}}(\mathcal{A}) \cap \mathcal{B}_w/\mathcal{T}_{(,)}) \cap \mathcal{C}_n$ accepts all minimal $\mathcal{F}_{\mathcal{A}}$-factorizations of w. As n cannot exceed the number of states in the automaton $\alpha_{\mathrm{f}}(\mathcal{A}) \cap \mathcal{B}_w/\mathcal{T}_{(,)}$, it follows that $n = O(|\alpha_{\mathrm{f}}(\mathcal{A})||w|)$.

We can improve the above upper bound on n to $O(|\mathcal{A}_0||w|)$ as follows. Let

$$(\lambda)^{x_0}(y_1)(\lambda)^{x_1}\cdots(y_r)(\lambda)^{x_r}$$

be a minimal $\mathcal{F}_{\mathcal{A}}$-factorization of w of length n such that each factor y_i is nonempty and each integer x_j is nonnegative. Then, $r \leq |w|$. Assume there is ℓ such that $x_\ell \geq |\mathcal{A}_0|$. Then the factors $(\lambda)^{x_\ell}$ correspond to a computation, say $p_0 t_1 p_1 \cdots t_{x_\ell} p_{x_\ell}$, of the automaton $\mathcal{A}_0$ such that λ is in $L(\mathcal{A}_{t_j})$ for all $j = 1, \ldots, x_\ell$. As $x_\ell + 1$ exceeds the number of states in $\mathcal{A}_0$, at least one state is repeated in the computation, which implies that there is a shorter sequence of indices, say $t_1 \cdots t_j t_{j+c} \cdots t_{x_\ell}$ for some $c > 1$, in $L(\mathcal{A}_0)$. It follows then that a shorter factorization of w obtains if we replace $(\lambda)^{x_\ell}$ with $(\lambda)^{x_\ell - c + 1}$; a contradiction. Hence, $x_\ell < |\mathcal{A}_0|$ for all $\ell = 0, \ldots r$ and, therefore, $n = O(|w||\mathcal{A}_0|)$. We can also show that this bound is tight. For example, let $\mathcal{F}_{\mathcal{A}} = ((1^m 2)^*, a^*, \{b\})$ and let $w = b^r$ for some $m, r > 0$. Then the minimal factorization of w is $\varphi = (\lambda)^m(b)\cdots(\lambda)^m(b)$ whose length is $r(2m+3)$. Moreover, as $\mathcal{A}_0$ can be chosen to have $m+2$ states, it follows that the length of φ is $\Theta(|\mathcal{A}_0||w|)$. Hence, we have shown the following.

Theorem 4.1: Given a regular factorization $\mathcal{F}_{\mathcal{A}}$ and a word w, the problem of computing an NFA accepting all minimal $\mathcal{F}_{\mathcal{A}}$-factorizations of w can be computed in time $O(|\alpha_{\mathrm{f}}(\mathcal{A})||w|^2|\mathcal{A}_0|)$.

We note that the number of minimal $\mathcal{F}_{\mathcal{A}}$-factorizations of w could be exponential with respect to $|\mathcal{A}_0|$ and $|w|$, even when none of the languages $L(\mathcal{A}_i)$ contains the empty word. For example, let $\mathcal{F}_{\mathcal{A}} = (1^n 1^*, \Sigma^+)$ be the regular factorization such that a word w admits such a factorization if $w = w_1 \cdots w_r$ with $r \geq n$ and $w_j \neq \lambda$ for all j. Obviously, a minimal $\mathcal{F}_{\mathcal{A}}$-factorization of w consists of exactly n factors. As there are $|w|$ possible positions in w where a symbol (can be placed, it follows that there are $\binom{|w|}{n}$ ways of factoring w in n factors.

Now we turn to computing maximal $\mathcal{F}_{\mathcal{A}}$-factorizations of the given word w. Let N be the total number of maximal $\mathcal{F}_{\mathcal{A}}$-factorizations of w. First we test whether N is infinite. For this, it is sufficient to test whether the trim part of the automaton $\alpha_{\rm f}(\mathcal{A}) \cap \mathcal{B}_w/\mathcal{T}_{(,)}$ contains a cycle – this is possible as the automaton contains no empty transitions. This can be performed in linear time using the well-known topological sorting algorithm. If N is infinite we are done. Otherwise, N can be obtained if we compute a longest path in the automaton $\alpha_{\rm f}(\mathcal{A}) \cap \mathcal{B}_w/\mathcal{T}_{(,)}$ from the start state to a final state. Then, as before, the automaton $(\alpha_{\rm f}(\mathcal{A}) \cap \mathcal{B}_w/\mathcal{T}_{(,)}) \cap \mathcal{C}_N$ accepts all the maximal $\mathcal{F}_\mathcal{A}$-factorizations of w. Moreover, as in the case of minimal factorizations, it follows that $N = O(|\mathcal{A}_0||w|)$.

In case we need to compute the λ-free factorizations of w, we can simply use the above methods, after eliminating the empty word from each of the languages $L(\mathcal{A}_i)$. This can be done in linear time by constructing the automata $\mathcal{A}'_i = \mathcal{A}_i \cap \mathcal{C}_+$, where $\mathcal{C}_+$ is an automaton of size $O(1)$ accepting the language Σ^+.

For the fourth problem, we need the concept of transducer functionality. A tarnsducer $\mathcal{T}$ is said to be *functional* if $z_1, z_2 \in \mathcal{T}(w)$ implies that $z_1 = z_2$. If the transducer $\mathcal{T}$ is in standard form then the problem of whether the transducer $\mathcal{T}$ is functional can be decided in time $O(|\mathcal{T}|^2 \log |\mathcal{T}|)$ [3]. If $\mathcal{T}$ is real time then the same problem is decidable in time $O(|\mathcal{T}|^2)$ [1]. We also need the inverse of a transducer $\mathcal{T}$, denoted by $\mathcal{T}^{-1}$, which is defined such that $w \in \mathcal{T}^{-1}(z)$ if and only if $z \in \mathcal{T}(w)$. The transducer $\mathcal{T}^{-1}$ can be obtained from $\mathcal{T}$ in linear time by simply replacing each transition qx/yq' of $\mathcal{T}$ with the transition qy/xq' [8]. Now let $\mathcal{T}_1$ be a transducer of size $O(1)$, in standard form, that erases every input symbol in $\{(,)\}$ and preserves every input symbol in Σ. Then, the transducer $\mathcal{T}_1 \downarrow \alpha_{\rm f}(\mathcal{A})$ is of size $O(|\alpha_{\rm f}(\mathcal{A})|)$ and such that $w \in (\mathcal{T}_1 \downarrow \alpha_{\rm f}(\mathcal{A}))(\varphi)$ if and only if φ is an $\mathcal{F}_\mathcal{A}$-factorization of w. Then it follows that a given factorization $\mathcal{F}_\mathcal{A}$ has the uniqueness property if and only if the transducer $(\mathcal{T}_1 \downarrow \alpha_{\rm f}(\mathcal{A}))^{-1}$ is functional. Hence, the uniqueness property of a given factorization $\mathcal{F}_\mathcal{A}$ can be decided in time $O(|\alpha_{\rm f}(\mathcal{A})|^2 \log |\alpha_{\rm f}(\mathcal{A})|)$.

5. Further Results

In this section we consider the problem of computing all $\mathcal{F}_\mathcal{A}$-factorizations of the words in a given regular language such that each such factorization contains at most m (or at least m, or exactly m) factors, for some given nonnegative integer m. Then we discuss how to modify our methods about $\mathcal{F}_\mathcal{A}$-factorizations of words to obtain analogous results about labeled factorizations.

For a given nonnegative integer m we define the transducers $\mathcal{T}^{\geq m}$, $\mathcal{T}^{\leq m}$, and $\mathcal{T}^m$ as follows.

The input alphabet of the transducer $\mathcal{T}^{\geq m}$ is Σ, the output alphabet is $\Sigma \cup \{(,)\}$, the state set is $\{q_0, p_0, \ldots, q_m, p_m\}$, the start state is q_0, the final state is q_m, and the set of transitions consists of $q_i\lambda/(p_i$ and p_ix/xp_i, where $x \in \Sigma$, for $i = 0, \ldots, m$, and the transitions $p_m\lambda/)q_m$ and $p_j\lambda/)q_{j+1}$, for all $j = 0, \ldots, m-1$. It follows that $|\mathcal{T}^{\geq m}| = O(m)$ and that φ is in $\mathcal{T}^{\geq m}(w)$ if and only if $\varphi = (w_1)\cdots(w_r)$ for some $r \geq m$, and $w_1 \cdots w_r = w$. Obviously, when $m = 0$ the transducer $\mathcal{T}^{\geq m}$ is equal to the transducer $\mathcal{T}_{(,)}$ introduced in the previous section.

The transducer $\mathcal{T}^{\leq m}$ is defined similarly. The differences from $\mathcal{T}^{\geq m}$ are as follows. The set of states is $\{q_0, p_0, \ldots, q_{m-1}, p_{m-1}, q_m\}$, the set of final states is $\{q_0, \ldots, q_m\}$, and the set of transitions consists of $q_i\lambda/(p_i$, p_ix/xp_i, and $p_i\lambda/)q_{i+1}$, for all $i = 0, \ldots, m-1$. It follows that the size of $\mathcal{T}^{\leq m}$ is $O(m)$ and that $\varphi \in \mathcal{T}^{\leq m}(w)$ if and only if $\varphi = (w_1)\cdots(w_r)$ for some $r \leq m$, and $w_1 \cdots w_r = w$.

The transducer $\mathcal{T}^m$ differs from $\mathcal{T}^{\geq m}$ only in that q_m is the only final state. Then, $\varphi \in \mathcal{T}^m(w)$ if and only if $\varphi = (w_1)\cdots(w_m)$ and $w_1 \cdots w_m = w$.

Now consider, for any given NFA $\mathcal{B}$, the NFA $\alpha_{\rm f}(\mathcal{A}) \cap \mathcal{B}/\mathcal{T}^{\leq m}$ that accepts all $\mathcal{F}_\mathcal{A}$ factorizations φ of the words in $L(\mathcal{B})$ such that φ contains at most m factors. Using this observation, we obtain the following.

Theorem 5.1: Given a regular factrorization $\mathcal{F}_\mathcal{A}$, an NFA $\mathcal{B}$, and a nonnegative integer m, the length of a shortest word in $L(\mathcal{B})$ such that the word has an $\mathcal{F}_\mathcal{A}$-factorization with at most (or at least) m factors can be computed (if it exists) in time

$$O(m|\alpha_{\mathrm{f}}(\mathcal{A})||\mathcal{B}|\log(m|\alpha_{\mathrm{f}}(\mathcal{A})||\mathcal{B}|)).$$

Proof: Consider again the transducer $\mathcal{T}_1$ of the previous section that erases the sumbols (and), and preserves every symbol in Σ of the input word. Then the λ-NFA $\mathcal{C} = (\alpha_{\mathrm{f}}(\mathcal{A}) \cap \mathcal{B}/\mathcal{T}^{\leq m})/\mathcal{T}_1$ accepts all words that have an $\mathcal{F}_{\mathcal{A}}$-factrorization with at most m factors. Replace now each transition $q\lambda q'$ of $\mathcal{C}$ with the weighted edge $q0q'$, and each transition qxq' where $x \in \Sigma$ with the weighted edge $q1q'$. Then $\mathcal{C}$ becomes a weighted graph for which the length of a shortest path can be computed in time $O(|\mathcal{C}|\log|\mathcal{C}|)$ using Dijkstra's algorithm. $\square$

In case we want to compute the length of a shortest word in $L(\mathcal{B})$ that has an $\mathcal{F}_{\mathcal{A}}$ factorization with exactly m factors, we can compute a shortest unweighted path in the automaton $\alpha_{\mathrm{f}}(\mathcal{A}) \cap \mathcal{B}/\mathcal{T}^m$. In this case, the problem is solved in time $O(m|\alpha_{\mathrm{f}}(\mathcal{A})||\mathcal{B}|)$.

For the case of labeled factorizations, our results can be revised as follows. First we use the automaton $\alpha_\ell(\mathcal{A})$ instead of $\alpha_{\mathrm{f}}(\mathcal{A})$. As this automaton contains transitions of the form qtq' with $t \in I_{\mathcal{A}}$, each of the transducers $\mathcal{T}^{\leq m}$, $\mathcal{T}^{\geq m}$, $\mathcal{T}^m$, and $\mathcal{T}_1$ must be modified. For example, the new transducer $\mathcal{T}^{\geq m}$ should have the following transitions: $q_i\lambda/tp_i$ with $t \in I_{\mathcal{A}}$ and p_ix/xp_i with $x \in \Sigma$, for all $i = 0, \ldots, m$, and the transitions $p_m\lambda/tq_m$ and $p_j\lambda/tq_{j+1}$ with $t \in I_{\mathcal{A}}$, for all $j = 0, \ldots, m-1$. Obvioulsy now the size of each of the new transducers is increased by the factor $|I_{\mathcal{A}}|$, which should be accounted in the time complexities of the modified algorithms.

References:

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

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

[3] T. Head and A. Weber. Deciding code related properties by means of finite transducers. In *Sequences II, Methods in Communication, Security, and Computer Science*, pages 260–272, 1993.

[4] J. Karhumäki, W. Plandowski, and W. Rytter. Generalized factorizations of words and their algorithmic properties. *Theoretical Computer Science*, 218:123–133, 1999.

[5] S. Konstantinidis. Transducers and the properties of error-correction, error-detection and finite-delay decodability. *J. Universal Computer Science*, 2002. To appear.

[6] M. Lothaire. *Combinatorics on Words.* Addison-Wesley, Reading, MA, 1983.

[7] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume vol. I, Berlin, 1997. Springer-Verlag.

[8] S. Yu. Regular languages. In Rozenberg and Salomaa [7], pages 41–110.