ELSEVIER

Theoretical Computer Science 292 (2003) 131–143

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Minimizing subsequential transducers: a survey

Christian Choffrut

*LIAFA, Université Paris 7, 2, Pl. Jussieu, 75 251 Paris Cedex 05, France*

*Keywords:* Sequential machines; Finite automata; Minimization

## 1. Introduction

This paper deals with the notion of subsequential transducer, i.e., of finite deterministic automaton whose transitions are provided with an "output". Its purpose is twofold. First, it is meant to give better access to the result saying, in loose terms, that it is possible to define a notion of morphism on subsequential transducers in such a way that the ordinary theory of deterministic automata carries over to subsequential transducers. In particular, given a function realized by some subsequential transducer, there exists a "minimal" subsequential transducer realizing it such that all trimmed subsequential transducers map onto it. This is a critical departure from the more general case of rational functions where the existence of such a minimal object is still unsettled, cf. [13]. Thus, though more complex than finite deterministic automata, subsequential transducers enjoy the same "syntactical" property and the techniques for proving this fact are not substantially different. This appeared in my doctorial thesis and in the proceedings of the ICALP'79 conference. To be more precise, this was actually stated in a slightly larger context since the function was supposed to map the free monoid into the free group. In spite of being more general and easily "downgradable" to free monoids, this formalism has probably confused some readers and obscured the fact that it was dealing with free monoids also.

The second objective is to give an account of further works concerned with the actual construction of minimal transducers. All the constructions are based on the observation, which I made in my ICALP paper, that it is possible to preserve the structure of the underlying automaton and pull the output labels "upstream" in such a way that the longest common prefix of the labels of all the final paths leaving a given state is the empty word. A transducer enjoying this property can be minimized

*E-mail address:* cc@liafa.jussieu.fr (C. Choffrut).


PII: S0304-3975(01)00219-5

by considering the underlying finite automaton only. As a consequence, the problem reduces to estimating the cost of computing these longest common prefixes for each state. This approach is adopted in [12, p. 70] where it is observed that the computation is polynomial in time but no estimate of the exponent of the polynomial is given. The first serious attempt dates back to [8], see also [10], where a worst case time complexity in $O((L+1)m)$ is claimed, with $m$ the number of transitions and $L$ the maximum length, ranging over the states, of these longest common prefixes. Although this algorithm works correctly on most randomly given transducers, it is based on a wrong implicit assumption which makes it fail in some "pathological" cases, an example of which is given in the last section. Recently, Béal and Carton established this claim rigorously [3]. Their proof shows that the result requires a much more sophisticated artillery than that employed by Mohri. Later on, Breslauer tackled the problem in a different way by reducing it to a problem of single source shortest paths in a graph and by exhibiting an algorithm whose complexity is essentially proportional to the sum of the lengths of the output labels of the transducer. It starts with observing that substituting lengths for the actual outputs gives an upper bound on the length of the longest common prefixes. The clever idea consists in imposing an extra condition which, combined with the length condition, gives the correct result. It is difficult to compare the two complexities since they are expressed in terms of different parameters. It can, however, be said roughly, that Breslauer's algorithm is slightly less efficient than the corrected version of Mohri's algorithm for "randomly" given transducers.

There is a growing interest for subsequential functions outside the community of theoretical computer scientists. Indeed, it was long observed that they are relevant to lexical analysis, text processing, coding theory, computer arithmetics etc. Nowadays, transducers are commonly used by people involved with speech recognition [10] and machine learning [11]. Actually, there is ongoing research for extending these results to the much more difficult case where the outputs of the transducers are subsets of words, cf., e.g. [9]. So the topic is not yet exhausted. Subsequential functions can actually be viewed as a special case of rational relations. We refer the reader to Jean Berstel's textbook on the topic for a general exposition of transducers and their relation to traditional language theory [4].

## 2. Preliminaries

The present theory deals largely with partial functions. Indeed, the predecessors of the subsequential transducers were the *generalized sequential machines* of Ginsburg and Rose [7], all the states of which were final (equivalently which defined total functions). Soon some authors introduced the distinction between final and non-final states but Schützenberger added an initial and final processing of each input word which preserved the main feature of the model, that of being completely deterministic, while suppressing the artificial conditions of the original definition. That this model is relevant

is proven by the nice functional equations characterizing subsequential functions [14] or the metric characterization "à la Ginsburg and Rose" which they enjoy [6]. However, the price to pay is having to speak of partial rather than total functions. In order to avoid tedious case studies as much as possible, we augment any set $X$ with an extra element representing the "undefined value" denoted by 0 and we view each partial function $f: X \to Y$ as a total function mapping $X \cup \{0\}$ into the set $Y \cup \{0\}$ where $f(x) = 0$ whenever $f(x)$ is undefined and with the convention $f(0) = 0$. Observe that the new "undefined value" is common to all sets considered, be it a free monoid, a set of states or whatever. We may think of $f$ as assigning the singleton $\{f(x)\}$ to the singleton $\{x\}$ whenever $f(x)$ is defined and the empty set otherwise. Equivalently, the 0 represents the empty set. The *domain* of the partial function $f$ is the set of $x \in X$ such that $f(x) \neq 0$ and is denoted as usual by $\mathrm{Dom}(f)$.

### 2.1. Notations

The free monoid generated by a set (or *alphabet*) $A$ is denoted by $A^*$. Its elements are *words*, its unit is the *empty word* and is denoted by $\varepsilon$. We set $M(A) = A^* \cup \{0\}$. By making the element 0 act as a zero on $M(A)$ ($u0 = 0u = 0$ for all $u$) $M(A)$ is a monoid. The concatenation product between elements is extended in the standard way to subsets: if $X, Y$ are two subsets then $XY = \{xy \mid x \in X,\ y \in Y\}$. In particular, if $X = \emptyset$ then $XY = \emptyset$.

Given two elements $u, v \in M(A)$, we say $u$ is a *prefix* of $v$ whenever there exists an element $w$ such that $v = uw$ holds and we write $u \leqslant v$. Observe that this defines a partial ordering whose greatest element is 0. Given a subset $X \subseteq M(A)$ we denote by $\bigwedge X$ the longest common prefix of the words in $X$ if $X \neq \emptyset$. We pose $\bigwedge X = 0$ if $X$ contains no word, i.e., $X = \emptyset$ or $X = \{0\}$. When $X$ contains two, not necessarily different elements $u, v$, we simply write $u \wedge v$ instead of $\wedge\{u, v\}$. It is clear that if $X \cap A^* \neq \emptyset$, then there exist two words $u$ and $v$ such that $\bigwedge X = u \wedge v$.

Given two elements $u, v \in M(A)$, we pose $u^{-1}v = w$ if $v = uw$ holds for some unique element $w$ and $u^{-1}v = 0$ otherwise, e.g. $(ab)^{-1}abaa = aa$, $(ab)^{-1}baa = 0$ (there is no element $w$), $(ab)^{-1}0 = 0$, and $0^{-1}0 = 0$ (there are infinitely many elements $w$). The following holds:

$$\forall u, v, w \in M(A), \quad (uv)^{-1}w = v^{-1}(u^{-1}w), \tag{1}$$

$$\forall u \in M(A), \quad u^{-1}0 = 0^{-1}u = 0, \tag{2}$$

$$\forall u \in A^*, \quad u^{-1}u = \varepsilon. \tag{3}$$

Of course, if $x$ is an element and $X, Y$ are two subsets of $M(A)$ we have

$$xX \subseteq Y \ \Rightarrow\ \bigwedge Y \leqslant x \bigwedge X, \tag{4}$$

$$\bigwedge xX = x \bigwedge X. \tag{5}$$

### 2.2. Subsequential transducers and subsequential functions

We recall the basics on the notion of subsequential transducers introduced by Schützenberger in [14]. It is a deterministic device provided with a finite memory that performs a mapping from a free monoid into another.

A *subsequential* transducer is a construct $\mathscr{T} = (Q, A, B, q_-, \mathtt{t}, \mathtt{i})$ where

(i) $A$ and $B$ are the *input* and *output* alphabets.
(ii) $Q$ is the finite set of *states*.
(iii) $q_-$ is a subset of $Q$ containing at most one element in which case we identify this element with the subset and we call it the *initial* state.
(iv) $\mathtt{t} : Q \to B^*$ is the *termination* partial function.
(v) $\mathtt{i} \in M(B)$ is the *initialization value* with $\mathtt{i} \in B^*$ if $q_-$ is a singleton and $\mathtt{i} = 0$ if $q_- = \emptyset$.

We assume that a *transition* partial function is given which associates with each pair $(q, a) \in Q \times A$ of its domain the *next state* $q \cdot a \in Q$. Observe that the function defines an action on the monoid $M(A)$ by the usual induction on the length of the words and by setting $q \cdot 0 = 0$ for all $q \in Q$ and $0 \cdot a = 0$ for all $a \in A \cup \{0\}$. Also a *production function* associates with each pair $(q, a) \in Q \times A$ of its domain the *output word* $q * a \in B^*$. We denote by $Q_+$ the domain of $\mathtt{t}$ and call it the subset of *final* states. A *path* is a finite sequence of quadruples in $Q \times A \times B^* \times Q \cup \{(0,0,0,0)\}$.

$$(q_0, a_0, y_0, q_1), (q_1, a_1, y_1, q_2), \ldots, (q_{n-1}, a_{n-1}, y_{n-1}, q_n)$$

with $q_i \cdot a_i = q_{i+1}$ and $q_i * a_i = y_i$ for all $i = 0, \ldots, n-1$. We say the path *starts* in $q_0$ and *ends* in $q_n$. A path is *final* if $q_n \in Q_+$ and it is *successful* if furthermore $q_0 = q_-$. In particular, when $q_- = 0$ then there is no successful path. The elements $a_0 a_1 \ldots a_{n-1} \in M(A)$ and $y_0 y_1 \ldots y_{n-1} \in M(B)$ are, respectively, the *input* and the *output labels* of the path. We consider an empty path $(n = 0)$ for each state $q$ and we view it as starting and ending in $q$ and labelled by $\varepsilon$. A transducer is *trimmed* if each state $q \in Q$ lies on a successful path.

The function *realized* by the subsequential transducer is also called subsequential and is defined for all $u \in M(A)$ by the condition

$$f(u) = \mathtt{i}(q_- * u)\mathtt{t}(q_- \cdot u).$$

It is clear from the definition that there is no loss of generality in assuming that the transition and the production functions have the same domain of definition.

For example, the function realized by the transducer of Fig.1 has the initial state (here 0) indicated by an incoming arrow with no source and labeled by $\mathtt{i} = \varepsilon$. The final states $q$ (here 1 and 3) are indicated by outcoming arrows with no target and labelled by $\mathtt{t}(q)$ (here *ba* and *a*, respectively).

*Unless otherwise stated, all transducers considered in this paper are assumed trimmed.* Observe that there exists a unique trimmed transducer realizing the function which is nowhere defined.

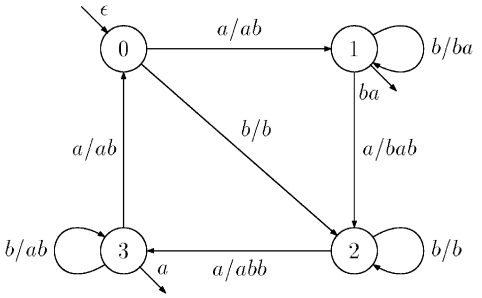


Fig. 1. A subsequential transducer.

## 3. Syntactic congruence

Given a function $f : A^* \to B^*$, with each $u \in A^*$ we set $F(u) = \bigwedge_{w \in A^*} f(uw) \in M(B)$. Observe that by equality (4) this function $F$ is prefix preserving, i.e., $u \leqslant v$ implies $F(u) \leqslant F(v)$. For all $u, v \in A^*$ we define $u \underset{f}{\sim} v$ is and only if the following is satisfied:

$$\text{for all } w \in A^*, \quad F(u)^{-1} f(uw) = F(v)^{-1} f(vw). \tag{6}$$

This equivalence relation is finer than the right equivalence associated with the underlying automaton (the "regular" equivalence in the literature) since the following holds:

$$u \underset{f}{\sim} v \text{ implies for all } w \in A^*\text{: } uw \in \mathrm{Dom}(f) \;\Leftrightarrow\; vw \in \mathrm{Dom}(f). \tag{7}$$

In order to check whether it is right invariant we denote by $g(w)$ the common value of both sides of Eq. (6). Observe that $F(u) = 0$ holds if and only if $F(v) = 0$. In this case we have $F(ua) = F(va) = 0$ and we are done. Assume now $F(ua) \neq 0$ and $F(va) \neq 0$ and set $x = \bigwedge_{w \in A^*} g(aw) \in B^*$. By equality (5), we get

$$F(ua) = \bigwedge_{w \in A^*} f(uaw) = \bigwedge_{w \in A^*} F(u) g(aw) = F(u) \bigwedge_{w \in A^*} g(aw) = F(u)x$$

and thus

$$F(ua)^{-1} f(uaw) = (F(u)x)^{-1} F(u) g(aw) = x^{-1} g(aw).$$

The last equality follows from $(F(u)x)^{-1} F(u) = x^{-1}(F(u)^{-1} F(u)) = x^{-1}$ by virtue of identities (1). In the same way, we would prove that $F(va)^{-1} f(vaw) = x^{-1} g(aw)$ holds.

We call $\underset{f}{\sim}$ the *syntactic* congruence of the function.

We may associate with each partial function $f : A^* \to B^*$ a (not necessarily finite) subsequential transducer

$$\mathscr{T}_f = (Q_f, A, B, q_f, \mathtt{t}_f, \mathtt{i}_f), \tag{8}$$

where $Q_f=\{[u]\mid F(u)\neq 0\}$, $q_f=\{[\varepsilon]\}$ if $F(\varepsilon)\neq 0$ and $q_f=\emptyset$ otherwise, $\mathtt{i}_f=F(\varepsilon)$ and $\mathtt{t}_f([u])=F(u)^{-1}f(u)$ for all $u\in A^*$ which is well-defined by equality (6). For the transition function we set $[u]\cdot a=0$ if $F(ua)=0$, else $[u]\cdot a=[u\cdot a]$ which is well-defined since the equivalence is right invariant. The production function is defined by $[u]*a=F(u)^{-1}F(ua)$. In order to show that this is also well-defined consider $u\underset{f}{\sim}v$. We have $F(ua)=0$ is and only if $F(va)=0$. Assume thus that $F(ua)\neq 0$ and choose an arbitrary word $w$ such that $uaw, vaw\in \mathrm{Dom}(f)$ (such a word exists by (7)). Then applying the definition of the syntactic equivalence to $u\underset{f}{\sim}v$ there exists $x\in B^*$ such that $f(uaw)=F(u)x$ and $f(vaw)=F(v)x$. Also, the condition $ua\underset{f}{\sim}va$ implies that there exists $y\in B^*$ such that $f(uaw)=F(ua)y$ and $f(vaw)=F(va)y$ holds. Now, $F$ is prefix preserving and $f(uaw)=F(u)x=F(ua)y$ which shows that for some $z\in B^*$ we have $x=zy$. Then we obtain $z=F(u)^{-1}F(ua)=F(v)^{-1}F(va)$ as claimed.

Now, we verify that the transducer thus defined computes the function $f$. Let us calculate the image of $u=a_1a_2\ldots a_n\in A^*$ in the subsequential transducer. Pose $q_i=q_f\cdot a_1\ldots a_i$ for all $i=0,\ldots,n$. Compute

$$\begin{aligned}F(\varepsilon)(q_f*u)\mathtt{t}_f(q_f\cdot u) &= F(\varepsilon)(q_f*a_1)(q_1*a_2)\ldots(q_{n-1}*a_n)\mathtt{t}_f(q_n)\\ &= F(\varepsilon)(F(\varepsilon)^{-1}F(a_1))(F(a_1)^{-1}F(a_1a_2))\ldots\\ &\qquad\times(F(a_1\ldots a_{n-1})^{-1}F(a_1\ldots a_n))F(u)^{-1}f(u)\\ &= f(u).\end{aligned}$$

Indeed, if all $F(a_1\ldots a_i)$'s are different from 0, then everything cancels out except $f(u)$. Otherwise, both sides are equal to 0.

**Theorem 1.** *A function $f:A^*\to B^*$ is subsequential if and only if its syntactic congruence has finite index.*

**Proof.** Clearly, the condition is necessary. Indeed, let $\mathscr{T}$ be a subsequential transducer realizing $f$ and consider the equivalence on $A^*$ defined by $u\sim v$ if and only if $q_-\cdot u=q_-\cdot v$. If $q_-\cdot u=q_-\cdot v=0$ then $F(u0)=F(v)=0$ and condition (6) is met. Otherwise, if $q=q_-\cdot u=q_-\cdot v\neq 0$ holds for all $w\in A^*$, denote by $h(w)$ the value $(q*w)\mathtt{t}(q\cdot w)$. Finally, let $y=\bigwedge_{w\in A^*}h(w)$. Then we have $F(u)=\mathtt{i}(q_-*u)y$, $F(v)=\mathtt{i}(q_-*v)y$ and thus

$$F(u)^{-1}f(uw)=y^{-1}h(w)=F(v)^{-1}f(vw).$$

Since the relation $\sim$ is finer than the syntactic congruence, the latter has finite index.

The converse is a consequence of the construction previous to the present theorem. □

## 4. Morphisms of subsequential transducers

The purpose of this section is to show that it is posible to define a notion of morphisms on the set of trimmed subsequential transducers. Consider two such transducers.

$$\mathscr{T}^{(1)} = (Q^{(1)}, A, B, q_-^{(1)}, \mathtt{t}^{(1)}, \mathtt{i}^{(1)}) \quad \text{and} \quad \mathscr{T}^{(2)} = (Q^{(2)}, A, B, q_-^{(2)}, \mathtt{t}^{(2)}, \mathtt{i}^{(2)}).$$

We are given a partial mapping $h : Q^{(1)} \to Q^{(2)}$ and a total mapping $\ell : Q^{(1)} \in B^* \cup (B^*)^{-1}$, where $(B^*)^{-1}$ is the set of formal inverses of elements of $B^*$. We denote by $Q_+^{(1)}$, and $Q_+^{(2)}$ the sets of final states of $\mathscr{T}^{(1)}$ and $\mathscr{T}^{(2)}$, respectively.

The pair $(h,\ell)$ is a *morphism* of $\mathscr{T}^{(1)}$ onto $\mathscr{T}^{(2)}$ if the following conditions hold:

$$h(q_-^{(1)}) = q_-^{(2)} \quad \text{and} \quad Q_+^{(1)} = h^{-1}(Q_+^{(2)}), \tag{9}$$

$$\text{for all } q \in Q \text{ and } a \in A\text{: } h(q) \cdot_2 a = h(q \cdot_1 a), \tag{10}$$

$$\mathtt{i}^{(2)} = \mathtt{i}^{(1)}\ell(q_-), \tag{11}$$

$$\text{for all } q \in Q\text{: } \mathtt{t}^{(2)}(h(q)) = \ell(q)^{-1}\mathtt{t}^{(1)}(q), \tag{12}$$

$$\text{for all } q \in Q \text{ and } a \in A\text{: } h(q) *_2 a = \ell(q)^{-1}(q *_1 a)\ell(q \cdot_1 a). \tag{13}$$

The mapping $\ell$, meant as defining a "lag" on the output labels, requires some comment. Observe that the labels of the transitions leaving state 1 of Fig. 1 all start with the prefix $b$. Since 1 is not a final state, the behaviour of the transducer is not modified if this prefix is stripped of these labels and assigned to the right of the incoming transitions. By a formal inverse of $u = b_1 \dots b_n$ we mean the sequence $u = b_n^{-1} \dots b_1^{-1}$. The formal inverse of the empty word $\varepsilon$ is $\varepsilon$ itself. If $u \in B^*$ the notation $u^{-1}v$ is to be understood as in Section 2.1. If $u \in (B^*)^{-1}$ then $u$ is the formal inverse of $u' \in B^*$ and by definition $u^{-1}v$ is equal to $u'v$.

For example, the transducer of Fig. 1 is mapped onto the following transducer via the morphism $(h,\ell)$ where $h(0) = h(2) = \bar{0}$, $h(1) = h(3) = \bar{1}$ $\ell(0) = \varepsilon$, $\ell(1) = ba$, $\ell(2) = \varepsilon$, $\ell(3) = a$ (see Fig. 2).

**Proposition 1.** *If $(h,\ell)$ is a morphism from $\mathscr{T}^{(1)}$ onto $\mathscr{T}^{(2)}$ then the two transducers realize the same subsequential functions.*

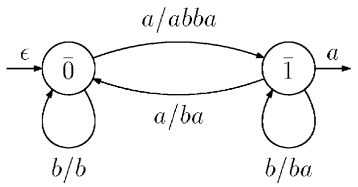


Fig. 2. A morphic image of the subsequential transducer of Fig. 1.

**Proof.** Indeed, consider a word $u=a_1a_2\dots a_n$ and its computation in the transducer $\mathscr{T}^{(1)}$. For all $0\leqslant i\leqslant n$ set $q_-\cdot a_1\dots a_i=q_i$. Then the image of $u$ in $\mathscr{T}^{(1)}$ is the word

$$\mathtt{i}^{(1)}(q_0 *_1 a_1)(q_1 *_1 a_2)\dots(q_{n-1} *_1 a_n)\mathtt{t}^{(1)}(q_n).$$

By the definition of the morphism, and by setting $h(q_i)=q_i^{(2)}$ for $i=0,\dots,n$, the image of the same word in the transducer $\mathscr{T}^{(2)}$ is

$$\begin{aligned}&\mathtt{i}^{(2)}(q_0^{(2)} *_2 a_1)(q_1^{(2)} *_2 a_2)\dots(q_{n-1}^{(2)} *_2 a_n)\mathtt{t}^{(2)}(q_n)\\&\quad=\mathtt{i}^{(1)}\ell(q_0)(\ell(q_0)^{-1}(q_0 *_1 a_1)\ell(q_1))\dots\\&\qquad(\ell(q_{n-1})^{-1}(q_{n-1} *_1 a_1)\ell(q_n))\ell(q_n)^{-1}\mathtt{t}^{(1)}(q_n)\\&\quad=\mathtt{i}^{(1)}(q_0 *_1 a_1)(q_1 *_1 a_2)\dots(q_{n-1} *_1 a_n)\mathtt{t}^{(1)}(q_n).\end{aligned}$$

Because of the hypotheses on $h$, the converse is also true. □

**Proposition 2.** *For all trimmed subsequential transducers $\mathscr{T}$ realizing a subsequential function $f$, there exists a unique morphism from $\mathscr{T}$ onto $\mathscr{T}_f$.*

**Proof.** Given a transducer $\mathscr{T}=(Q,A,B,q_-,\mathtt{t},\mathtt{i})$ and the transducer (8), we define a mapping $(h,\ell)$ from the former onto the latter and then we verify whether it satisfies conditions (9)–(13). For all $q\in Q$ we set

$$\begin{aligned}&h(q)=[u]\text{ for some arbitrary }u\text{ with }q_-\cdot u=q,\\&\ell(q)=\bigwedge_{q\cdot w\in Q_+}(q*w)\mathtt{t}(q\cdot w).\end{aligned}$$

Clearly $h$ is well-defined since $q_-\cdot u=q_-\cdot v$ implies $[u]=[v]$.

It is straightforward to see that condition (9) is satisfied. Since $\underset{f}{\sim}$ is right invariant we have $h(q\cdot a)=[ua]=[u]\cdot a=h(q)\cdot a$ proving that condition (10) also holds. Observe that the function $\ell$ satisfies, for all $u\in A^*$ and $q_-\cdot u=q$,

$$\mathtt{i}(q_-*u)\ell(q)=F(u).\tag{14}$$

Indeed, we have

$$\begin{aligned}F(u)&=\bigwedge_{w\in A^*}\mathtt{i}(q_-*u)(q*w)\mathtt{t}(q\cdot w)=\mathtt{i}(q_-*u)\bigwedge_{w\in A^*}(q*w)\mathtt{t}(q\cdot w)\\&=\mathtt{i}(q_-*u)\ell(q).\end{aligned}$$

As a consequence, we get $F(\varepsilon)=\mathtt{i}\ell(q_-)$ which is condition (11). For all $q_-\cdot u=q\in Q_+$ the following holds:

$$t_f(h(q))=F(u)^{-1}f(u)=F(u)^{-1}\mathtt{i}(q_-*u)\mathtt{t}(q)=\ell(q)^{-1}\mathtt{t}(q),$$

which establishes condition (12).

Applying equality (14) to $ua$, we have $\mathtt{i}(q_- * ua)\ell(q \cdot a) = F(ua)$. This entails

$$\ell(q)^{-1}(q * a)\ell(q \cdot a) = F(u)^{-1}\mathtt{i}(q_- * u)(q * a)(q_- * ua)^{-1}\mathtt{i}^{-1}F(ua)$$

$$= F(u)^{-1}F(ua)$$

which completes the verification of condition (13). □

## 5. Complexity considerations

Given a subsequential transducer $\mathscr{T}$ and a state $q$, denote by $\lambda_{\mathscr{T}}(q)$ the longest common prefix of all the output labels of the final paths starting from $q$, i.e.,

$$\lambda_{\mathscr{T}}(q) = \bigwedge_{u \in A^*} (q * u)\mathtt{t}(q \cdot u).$$

The construction of the minimal transducer realizing a given subsequential function is based on the simple property first observed in [6, p. 96] that there exists a transducer $\mathscr{T}'$ realizing the same function as $\mathscr{T}$ having the same underlying automaton and such that $\lambda_{\mathscr{T}'}(q) = \varepsilon$ holds for all states $q \in Q$. Indeed, we can define a total mapping $\lambda : Q \to B^*$ by setting

$$\lambda(q) = \bigwedge\{(q * u)\mathtt{t}(q.u) \mid q \cdot u \in Q_+\}. \tag{15}$$

Define a new production function by setting $q \circ a = \lambda(q)^{-1}(q * a)\lambda(q \cdot a)$. That $q \circ a \in B^*$ holds results from equality (4) and the following inclusion:

$$(q * a)\{((q \cdot a) * u)(\mathtt{t}(q \cdot au)) \mid a \in A, u \in A^*\}$$

$$= \{(q * au)(\mathtt{t}(q \cdot au)) \mid a \in A, u \in A^*\} \subseteq \{(q * u)(\mathtt{t}(q \cdot u)) \mid u \in A^*\}.$$

The resulting transducer has the right property. As a consequence of Proposition 2, minimizing the transducer $\mathscr{T}'$ can be achieved by minimizing its underlying finite automaton since the function $\ell$ in the definition of a morphism is then necessarily constant and equal to the empty word, e.g. the transducer of Fig. 1 can be transformed into the transducer shown in Fig. 3 which enjoys the right property.

Minimizing an automaton can be achieved in worst case time complexity $O(n \log n)$, where $n$ is the number of states, cf. [1], while computing all the longest prefixes has complexity $O((L+1)m)$, where $m$ is the number of transitions and $L$ is the maximum of the lengths of $\lambda_{\mathscr{T}}(q)$ for $q \in Q$ as is briefly reported now.

Several authors have investigated the complexity of constructing the minimal transducer [8, 5, 3]. They use different methods but they all start by observing that the input labels are irrelevant and thus can be ignored. In order to account for the functions

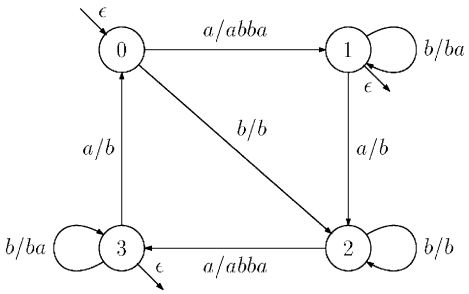


Fig. 3. A subsequential transducer.

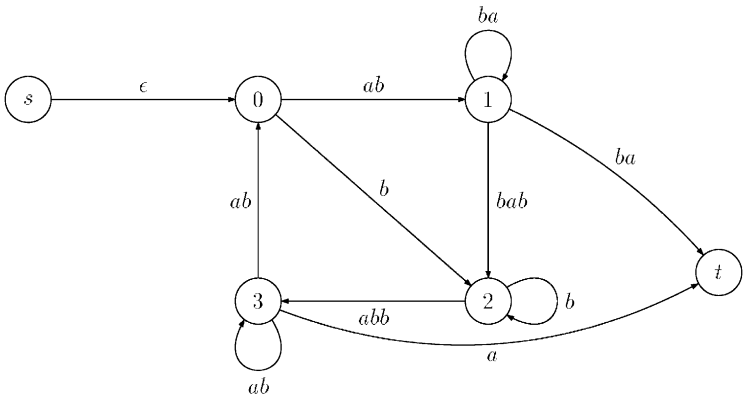


Fig. 4. The graph associated with Fig. 1.

t and i, two new vertices, a *source* $s$ and a *target* $t$, are added. The resulting object is a finite graph $G=(Q,E)$ whose vertices are the states of the transducer and whose set $E$ of edges, labelled by words in $B^*$, are its transitions. In order to make this more evident, we adopt the terminology of graphs and speak of nodes rather than states, of edges rather than transitions. The label born by the edge $(q,p)$ is denoted by $\mathtt{label}(q,p)$ (with $\mathtt{label}(q,p)=0$ if $(q,p)\notin E$). The question arises as to how to compute, for all nodes $q$ of the graph, the longest common prefix $\lambda(q)$ of all the paths starting in $q$ and ending in $t$, e.g. Fig. 1 is transformed into a graph as shown in Fig. 4.

The maximum length of $\lambda(q)$ when $q$ ranges over $Q$ is denoted by $L$. Mohri was the first to propose an algorithm with claimed worst case time complexity in $O((L+1)m)$, where $m$ is the number of edges. However, it does not work correctly in all cases since it is based on the wrong (though not explicitly stated) assumption that in order to guarantee the "global" condition $\lambda(q)=\varepsilon$ for all $q\in Q$, it suffices to ensure the "local" condition: $\bigwedge_{(q,p)\in E}\mathtt{label}(q,p)=\varepsilon$ for all $q\in Q$. Fig. 5 shows that this need not be the case. In fact, it is tempting to consider the $Q$-tuple $(\lambda(q))_{q\in Q}$ as the solution of a system of equations over the monoid $M(B)$.

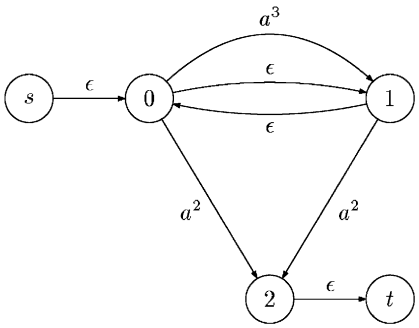


Fig. 5. A system with several solutions.

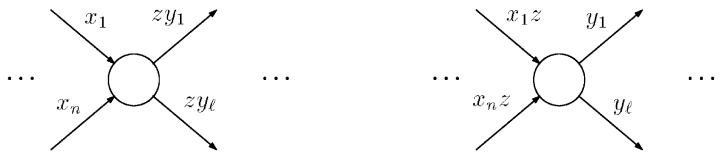


Fig. 6. The common prefix migrates "upstream".

With the above graph, e.g. we are led to associate a system which when reduced to the unknowns $\lambda(0)$, $\lambda(1)$ and $\lambda(2)$ is as follows:

$$\lambda(0) = a^3\lambda(1) \wedge \lambda(1) \wedge a^2\lambda(2),$$

$$\lambda(1) = \lambda(0) \wedge a^2\lambda(2),$$

$$\lambda(2) = \varepsilon. \tag{16}$$

Clearly, this system has three different solutions in the variables $\lambda(0)$ and $\lambda(1)$, to wit, $\lambda(0) = \lambda(1) = \varepsilon$, $\lambda(0) = \lambda(1) = a$, and $\lambda(0) = \lambda(1) = a^2$. The longest common prefixes are actually given by the last solution, which is the greatest fixed point of the decreasing function defined in (16). The solution is thus obtained by setting all three unknowns to 0 and by iteratively applying function (16). Mohri's algorithm, however, would compute the first solution. It would determine the states where the outputs of the leaving edges have a common prefix different from the empty word. For such states, the maximum common prefix would be deleted from the labels of the leaving edges and it would be added to the right of the labels of the entering edges (see Fig. 6) Here it would stop without doing anything.

More precisely, Mohri considers the classical decomposition of the graph into its largest strongly connected components (SCC) and visits the components in a reverse topological ordering. Each component $C$ is treated successively. Every node in $C$ is given a status: `dead`, `live`, `sleeping`. A node $q$ is `dead` whenever there exist two leaving arcs labelled by two non-empty words starting with different letters: $\texttt{label}(q, p) = au$ and $\texttt{label}(q, r) = bv$ for $p, r \in Q$, $a \neq b \in A$, $u, v \in A^*$. Clearly, in this

case, we have $\lambda(q)=\varepsilon$ and the node needs no further treatment. A node $q$ is `live` if the labels of all leaving edges have a non-empty common prefix, say $w$. Then $w$ may migrate from leaving to entering edges. Finally, a node $q$ is `sleeping` in all other cases, i.e., it has a leaving arc labelled by $\varepsilon$. The `sleeping` nodes are not discarded since they may change status (the empty label may receive some non-empty word from a neighbouring `live` node). The algorithm investigates the live nodes only and when it stops it guarantees that all nodes satisfy the condition on the labels "locally" but not globally, i.e.,

$$\forall q \in Q\colon \bigwedge_{p\in Q} \texttt{label}(q,p) = \varepsilon. \tag{17}$$

Béal and Carton have corrected this error in [3]. More precisely, they have denoted by $L_G$ the maximum length of the strings of the form $\lambda(q)$ where $q\in Q$. A depth-first search on the subgraph $G'$ obtained by deleting all non-empty labels is run in order to determine its maximum strongly connected components and compute the first common letter of all the outgoing labels, if such a letter exists at all. A second depth-first search on the entire graph $G$ performs the migration of this letter resulting in a new graph $G_1$. At this point, we have $L_{G_1}=L_G-1$. It suffices to proceed in this manner $L_G$ times in order to obtain the desired graph. The overall cost is in $O((L+1)m)$ as claimed.

Breslauer proceeds in a different manner, by reduction to a problem of shortest-path in a graph. Indeed, the label of each edge is replaced by its length and viewed as a "distance" between the two ends of the edge. The shortest distance of each node $q$ to some final node is an upper bound on the length of $\lambda(q)$. In general, it is strictly greater. In order to enforce equality the author uses the following approximation of $\lambda(q)$. Define an arbitrary covering forest of the graph where the final nodes are the roots of the trees composing the forest. With every node $q$ associate the label $L(q)$ of the unique path leading from $q$ to some root and set

$$C(q) = \bigwedge_{(q,p)\in E} \texttt{label}(q,p)L(p). \tag{18}$$

Add a vertex $\top$ and connect each root to this node with an edge of distance 0. Also connect each node $q$ with an edge of distance $|C(q)|$. The author proves that $\lambda(q)$ is the prefix of $C(q)$ whose length is the shortest distance from $q$ to $\top$. These values can be computed efficiently by any variant of Dijkstra's algorithm (cf. [2]). The overall complexity is dominated by the cost of solving (18). Using the structure of "suffix tree", the author shows that the complexity is in $O(n+m+s|B|)$, where $s$ is the sum of the lengths of the labels of the graph and $|B|$ the cardinality of the alphabet.

## References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Analysis, Addison-Wesley, Reading, MA, 1974.

[2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[3] M.P. Béal, Olivier Carton, Computing the prefix of an automaton, RAIRO, Inf. Théor. Appl., to appear.
[4] J. Berstel, Transductions and Context-Free Languages, Teubner, Stuttgart, 1979.
[5] D. Breslauer, The suffix tree of a tree and minimizing sequential transducers, Theoret. Comput. Sci. 191 (1998) 131–144.
[6] C. Choffrut, A combinatorial problem in the theory of free monoids, in: H.A. Maurer (Ed.), Proc. 6th ICALP Conference, Lecture Notes in Computer Science, vol. 71, Springer, Berlin, 1979, pp. 88–103.
[7] S. Ginsburg, G.F. Rose, A characterization of machine mappings, Canad. J. Math. 18 (1966) 381–388.
[8] M. Mohri, Minimisation of Sequential Transducers, Lecture Notes in Computer Science, vol. 807, Springer, Berlin, 1994, pp. 151–163.
[9] M. Mohri, Finite-state transducers in language and speech processing, Comput. Linguistics 23 (1997) 269–311.
[10] M. Mohri, Minimization algorithms for sequential transducers, Theoret. Comput. Sci. 234 (2000) 177–201.
[11] J. Oncina, P. Garcia, E. Vidal, Learning subsequential transducers for pattern recognition and interpretation tasks, IEEE Trans. Pattern Anal. Mach. Intell. 15 (1993) 448–458.
[12] C. Reutenauer, Subsequential functions: characterizations, minimization, examples, International Meeting of Young Computer Scientists, Lecture Notes in Computer Science, vol. 464, Springer, Berlin, 1990, pp. 62–79.
[13] C. Reutenauer, M.P. Schüzenberger, Minimization of rational word functions, Siam J. Comput. 30 (4) (1991) 669–685.
[14] M.P. Schüzenberger, Sur une variante des fonctions séquentielles, Theoret. Comput. Sci. 11 (1977) 47–57.