Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks

José Oncina, Pedro García, and Enrique Vidal

Abstract—**The "interpretation" framework in pattern recognition (PR) arises in the many cases in which the more classical paradigm of "classification" is not properly applicable generally because the number of classes is rather large or simply because the concept of "class" does not hold. A very general way of representing the results of interpretations of given objects or data is in terms of sentences of a "semantic language" in which the actions to be performed for each different object or datum are described. Interpretation can therefore be conveniently formalized through the concept of formal transduction, giving rise to the central PR problem of how to automatically learn a transducer from a training set of examples of the desired input–output behavior. This paper presents a formalization of the stated transducer learning problem, as well as an effective and efficient method for the inductive learning of an important class of transducers, namely, the class of subsequential transducers. The capabilities of subsequential transductions are illustrated through a series of experiments that also show the high effectiveness of the proposed learning method in obtaining very accurate and compact transducers for the corresponding tasks.**

Index Terms—**Formal languages, inductive inference, learning, rational transducers, subsequential functions, syntactic pattern recognition.**

I. Introduction

SYNTACTIC pattern recognition (SPR) is currently seen as a very appealing approach to many pattern recognition (PR) problems for which the most traditional decision–theoretic or statistical approaches fail to be effective [1]–[3]. One of the most interesting features of SPR consists of the ability to deal with highly structured (representations of) objects and to uncover the underlying structure by means of parsing. In fact, it is often claimed that such a distinctive feature is indeed what would enable SPR to go beyond the capabilities of other traditional approaches to PR. However, though parsing is perhaps the most fundamental and widely used tool of SPR, it is often used simply to determine the (likelihood of) membership of the test objects to a set of classes, each of which is modeled by a different syntactic model. Any possible structural byproduct of the parsing process is therefore discarded, thus considerably wasting the high potential of SPR.

Nevertheless, membership determination is in fact all that is needed if the problem considered is simply one of classification. However, SPR can be seen as particularly useful for many other interesting PR problems for which the classification paradigm is not properly applicable—simply because the concept of "class" does not hold or because the number of underlying classes is very large or infinite. In order to properly deal with these problems, one should move from the traditional, rather comfortable classification point of view to the most general paradigm of *interpretation* (see e.g., [4] and [5]).

Within this framework, a PR system is assumed to accept (convenient representations of) a certain class of objects as input and must produce, as output, adequate interpretations of these objects that are consistent with the *a priori* knowledge (model) with which the system's universe is embodied. Without loss of generality, interpretation results can be seen as "*semantic messages*" (SM) out of a (possibly infinite) set, which constitutes the "*semantic universe*" (SU) of the problem considered [4]. The complexity of the interpretation task is therefore determined by the complexity (size) of this universe. If the SU is "small" (i.e., a small number of different SM's is expected), interpretation can be viewed as the traditional process of classification by letting the output consist simply of a label specifying the obtained SM (class). On the other hand, if the SU is "large," a simple label can be inadequate to completely specify an SM, and the output should now consist of some type of *structured information*. In this case, rather than speaking of (pattern) "recognition," the term "*understanding*" is usually adopted instead [4].

One convenient and general way of representing SM's is by means of sentences from a "semantic language," which specifies the actions to be carried out as a response to the interpretation of a given object. For instance, the sequence of operations to be performed by a robotized machine tool in order to correctly position and mechanize a hypothesized (interpreted) input piece can be properly specified by a sentence of the command language of the machine.

Clearly, this calls for applying formal language techniques and then quite naturally places interpretation within the SPR framework. However, one should realize that the interpretation paradigm no longer supports the traditional assumption of one (syntactical) model per class; rather, all the system's *a priori* knowledge should now be represented as a whole into a single global model. In the theory of formal languages, such a representation can be properly accomplished through the concept of *transduction*. A transducer is a formal device that inputs structural representations of objects (strings) from a given input set (language) and outputs strings from another (usually different) output language. Many interesting results on properties of rational or finite-state transducers are well known

Manuscript received April 15, 1991; revised June 22, 1992. This work was supported by the Spanish CICYT under grant TIC-0448189. Recommended for acceptance by Associate Editor M. Nagao.

The authors are with the Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain.

IEEE Log Number 9208009.

from the classical theory of formal languages (see e.g., [6]), and some examples of their application to SPR are often given in the textbooks of SPR [1], [2]. However, for the transduction framework to be really fruitful, it is absolutely necessary to first solve the important problem of *how to learn or "train" these models from known instances of input–output strings.*

This problem, which is central from a PR point of view, has very rarely been dealt with in the literature. Apart from the particular treatment involved in Gold's study of the complexity of automata identification [7], only very limited and/or heuristic work seems to have been carried out so far. Perhaps one of the earliest works is that of Velenturf [8], in which a (nonincremental) algorithm is proposed to obtain a (reduced) Mealy machine, which is compatible with the given training data. If the source transducer is a (canonical) n-state Mealy machine and the training data contains all the input–output pairs of lengths smaller than $2n - 1$, then the algorithm is shown to identify a machine that is exactly equivalent to the source transducer. Another related work is that of Luneau *et al.* [9], in which a heuristic incremental procedure is proposed to infer Moore machines. Apart from the "heuristic" nature of this method, an important drawback is that the obtained transducers are not guaranteed to be compatible with the training data. One still more recent reference to the transducer inference problem is given in [10] in relation to a context-free grammar inference technique. The main drawback of all these works, however, is that they are strictly restricted to the inference of *sequential* transductions that, as will be discussed below, introduce strong limitations that are inadmissible in many cases of interest. Finally, Vidal *et al.* [11] proposed a (heuristic) methodology for the inference of transducers, which was reminiscent of the "morphic generator grammatical inference methodology" (MGGI) [20] for the inference of general regular languages. Although this work was based on rather sound theoretical arguments and produced interesting empirical results, no properties of identifiability were given, and the (heuristic) application of the methodology to each particular task required considerable skills on the part of the designer.

In this paper, we present a first step towards a formalization of the transducer learning (TL) problem from the point of view of its applicability within the interpretation paradigm. We have adopted a framework that is somewhat similar to that of grammatical inference (GI) [12]–[15], though one should realize that many important differences between GI and TL exist. In particular, we have tried to develop our work in the same direction as that of the so-called "characterizable" methods of GI [14], [16]–[17], that is, we have identified a (characterized) class of transductions and have developed a TL method that, under suitable conditions and using only positive presentations of input–output pairs, can be proved to correctly identify such a class.

This class is the class of *subsequential transductions* that is a subclass of the most general *rational or finite-state transductions* and properly contains the class of *sequential transductions* [6]. A sequential transduction is one that preserves the increasing length prefixes of input–output strings. Although this can be considered as a rather "natural property" of tranductions, there are many real-world situations in which such a strict sequentiality is clearly inadmissible. The class of subsequential transductions makes this restriction milder, therefore allowing application in quite interesting practical situations.

Although a detailed theoretical study of the subsequential TL problem has been carried out [18], [22], only those key formal issues needed for a convenient description of the proposed approach will be dealt with here. Instead, greater attention will be devoted to presenting the results in such a way that their possible application to pattern recognition problems can be easily understood. For this purpose, examples are used throughout the paper to clarify concepts and results. In addition, a number of experiments that illustrate both the usefulness of subsequential transduction and the ability of our lerning methods to actually obtaining the corresponding tranducers from positive training data will be presented.

It should be emphasized that both the methods and the experiments presented in the forthcoming sections are mainly aimed at *illustrating* the great potential of the proposed approach to the interpretation problems in PR. Obviously (rather direct), further developments of the methods introduced here, such as stochastic and/or error correcting extensions, are required before these methods can be properly applied to the kind of noisy and distorted data usually encountered in real PR situations. Nevertheless, we consider that the contents of this paper constitute, in fact, a required first step toward a new framework that would eventually launch SPR beyond the boundaries of the classification paradigm.

II. Mathematical Background and Notation

Let X be a finite set or *alphabet* and X^* the *free monoid* over X. For any string $x \in X^*$, $|x|$ denotes the length of x, and λ is the symbol for the string of length zero. For every $x, y \in X^*$, xy is the *concatenation* of x and y, with $|xy| = |x| + |y|$. If v is a string in X^*, then X^*v denotes the set of all strings of X^* that end with v; i.e., $X^*v = \{x \in X^* : uv = x, u \in X^*\}$. Similarly, $uX^* = \{x \in X^* : uv = x, v \in X^*\}$. On the other hand, $\forall x \in X^*$. $Pr(x)$ denotes the set of *prefixes* or *left factors* of x, that is $Pr(x) = \{u \in X^* : uv = x, v \in X^*\}$. Given $u, v \in X^*$, with $u \in Pr(v)$, the *right quotient* $u^{-1}v$ is the suffix of v that results after eliminating its prefix u, that is, $u^{-1}v = w \Leftrightarrow v = uw$. Given a set $L \subseteq X^*$, the *longest common prefix* of L is denoted as $lcp(L)$, where

$$lcp(L) = w \text{ iff } w \in \cap_{x \in L} Pr(x) \wedge \forall w' \\ (w' \in \cap_{x \in L} Pr(x) \Rightarrow |w'| \leq |w|).$$

In general, a *transduction* from X^* to Y^* is a relation $t \subseteq (X^* \times Y^*)$. In what follows, only those transductions that are (partial) *functions* from X^* to Y^* will be considered. For a partial function $t : X^* \rightarrow Y^*$, $Dom(t)$ denotes the subset of X^*, where t is defined. In order to simplify the forthcoming presentation and without loss of generality, all the functions t will be assumed to be defined on λ, with $t(\lambda) = \lambda$. A function t is finite if $Dom(t)$ is finite.

A *finite state* or *rational transducer* τ is defined as a six-tuple $\tau = (Q, X, Y, q_0, Q_F, E)$, where Q is a finite set of

states, $q_0 \in Q$ is the *initial state*, $Q_F \subseteq Q$ is a set of *final states*, and E is a finite subset of $(Q \times X^* \times Y^* \times Q)$ whose elements are called *edges*. Associated with an edge (q', x, y, q), there is a transition $(q', x, q) \in (Q \times X^* \times Q)$, which corresponds to the conventional concept in automata.

A *sequential transducer* is a rational transducer in which $E \subset (Q \times X \times Y^* \times Q), Q_F = Q$ and $(q, a, u, r), (q, a, v, s) \in E \Rightarrow (u = v \wedge r = s)$ (*determinism condition*) [6]. Correspondingly, Q_F can be omitted, and a sequential transducer is completely specified as a five-tuple $\tau = (Q, X, Y, q_0, E)$. Sequential transducers are also called "*generalized sequential machines*" (GSM's), from which Mealy and Moore machines are restricted instances.

A *path* in a sequential transducer τ is a sequence of edges $\pi = (q_0, x_1, y_1, q_1)(q_1, x_2, y_2, q_2) \cdots (q_{n-1}, x_n, y_n, q_n), q_i \in Q, x_i \in X, y_i \in Y^*, 1 \leq i \leq n$. Whenever the intermediate states involved in a path are not of particular concern, a path will be written as $\pi = (q_0, x_1 x_2 \cdots x_n, y_1 y_2 \cdots y_n, q_n)$. Let Π_τ be the set of all possible paths over τ. The transduction realized by a sequential transducer τ is the partial function $t : X^* \rightarrow Y^*$ defined as $t(x) = y$ iff $\exists (q_0, x, y, q) \in \Pi_\tau$.

Sequential transduction has the property of preserving left factors, that is, $t(\lambda) = \lambda$ and if $t(uv)$ exists, then $t(uv) \in t(u)Y^*$. Such a seemingly "natural" property (a restriction, in fact), however, can be quite inadmissible in many cases of interest. For instance, there are finite functions that are not sequential. In order to make this restriction milder, the following extension is introduced [6].

A *subsequential transducer* is defined as a six-tuple $\tau = (Q, X, Y, q_0, E, \sigma)$, where $\tau' = (Q, X, Y, q_0, E)$ is a sequential transducer, and $\sigma : Q \rightarrow Y^*$ is a (partial) function that assigns output strings to the states of τ'. The partial function $t : X^* \rightarrow Y^*$ that is realized by τ is defined as $t(x) = y\sigma(q)$ iff $\sigma(q)$ is defined, and $(q_0, x, y, q) \in \Pi_\tau$. The class of subsequential functions properly contains the class of sequential functions. In the same way as for conventional finite transducers, a subsequential transducer can be represented as a graph with the only difference that nodes are labeled as pairs $(q, \sigma(q))$, where $q \in Q$.

By using an additional input symbol "#," not in X, to mark the end of the input strings, any subsequential transduction $t : X^* \rightarrow Y^*$ can be obtained as a restriction to $X^*\#$ of a sequential transduction $t' : (X \cup \{\#\})^* \rightarrow Y^*$ so that $\forall x \in X^*, t(x) = t'(x\#)$ [6]. If $\tau = (Q, X, Y, q_0, E, \sigma)$ is a subsequential transducer that realizes t, then t' can be realized by the sequential transducer $\tau' = (Q, X \cup \{\#\}, Y, q_0, E')$, where $E' = E \cup \{(q, \#, \sigma(q), q_0) : q \in Q\}$. Note that any given subsequential transducer admits several different sequential representations. For instance, an alternative to the above sequential representation of τ can be obtained by adding an extra state $\hat{q}$ for every state q of Q and replacing every $(q, \#, \sigma(q), q_0)$ in E' for $(q, \#, \sigma(q), \hat{q})$. In what follows, the term "subsequential transducer" will be used, at convenience, either to denote a subsequential transducer as defined by $(Q, X, Y, q_0, E, \sigma)$ or a sequential representation thereof. Fig. 1 shows an example of a subsequential transducer and a sequential representation.

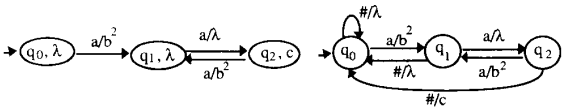


Fig. 1. Subsequential transducer of Example 1 and a sequential representation.

Example 1: The function $t : \{a\}^* \rightarrow \{b, c\}^*$ defined by

$$t(a^n) = \begin{cases} \lambda & n = 0 \\ b^{n+1} & n \text{ odd} \\ b^n c & n \text{ even} \end{cases}$$

is subsequential, but not sequential, since it does not preserve left factors. The function t is realized by the subsequential transducer $\tau = (Q, X, Y, q_0, E, \sigma)$ in which $Q = \{q_0, q_1, q_2\}, \blacktriangledown X = \{a\}, \blacktriangledown Y = \{b, c\}, \blacktriangledown E = \{(q_0, a, b^2, q_1), (q_1, a, \lambda, q_2), (q_2, a, b^2, q_1)\}$, and $\sigma(q_0) = \lambda, \sigma(q_1) = \lambda, \sigma(q_2) = c$. The graph of τ and that of a sequential representation of τ are shown in Fig. 1. □

Following, e.g., [12] and [14], an appropriate framework for the TL problem can be established as follows: Let $f : X^* \rightarrow Y^*$ be a partial recursive function, a TL algorithm $\mathcal{A}$ is said to identify f in the limit if, for any (positive) presentation of the input–output pairs (graph) of f, $\mathcal{A}$ converges to a transducer τ that realizes a function $g : X^* \rightarrow Y^*$ such that $\forall x \in Dom(f), g(x) = f(x)$.

III. Onward Subsequential Transducers

The TL algorithm to be presented in the next section requires some additional new concepts to be introduced. An *onward subsequential transducer* (OST) is one in which the output strings are assigned to the edges in such a way that they are as "close" to the initial state as they can be. Formally, a subsequential transducer $\tau = (Q, X, Y, q_0, E, \sigma)$ is an OST if

$$\forall p \in Q - \{q_0\} lcp(\{y \in Y^* | (p, a, y, q) \in E\} \cup \{\sigma(p)\}) = \lambda.$$

Much in the same way as happens in regular languages, any subsequential transduction admits a characterization in terms of a *canonical transducer in onward form*; in other words, for any subsequential transduction t, there exists an OST that has a minimum number of states and is unique up to isomorphism. Let $\tau(t)$ denote such a transducer. A construction for $\tau(t)$ can be developed using the concept of set of tails:

Let $t : X^* \rightarrow Y^*$ be a partial function, and let $x \in X^*$. The *set of tails* of x in $t, T_t(x)$ is defined as

$$T_t(x) = \{(y, v) | t(xy) = uv, u = lcp(t(xX^*))\}$$

with $T_t(x) = \emptyset$ if $x \notin Pr(Dom(t))$.

It can be seen that if t is subsequential, the number of different sets of tails in $\{T_t(x) | x \in Pr(Dom(t))\}$ is finite [18]. Therefore, these sets can be used to build a subsequential transducer $\tau(t) = (Q, X, Y, q_0, E, \sigma)$ as follows:

$$Q = \{T_t(x) | x \in Pr(Dom(t))\}: q_0 = T_t(\lambda);$$
$$E = \{(T_t(x), a, lcp(t(xX^*))^{-1} lcp(t(xaX^*)), T_t(xa)) | T_t(x), T_t(xa) \in Q\}$$

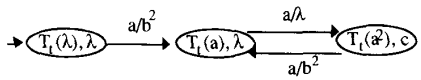


Fig. 2. Canonical transducer for the subsequential function of Example 2.

$$\sigma(T_t(x)) = \begin{cases} lcp(t(xX^*))^{-1}t(x) & \text{if } x \in Dom(t) \\ \emptyset & \text{otherwise.} \end{cases}$$

It can be easily seen [18] that such a transducer is onward and that any other OST for t either has a greater number of states or is isomorphic to $\tau(t)$.

Example 2: Let $t : \{a\}^* \rightarrow \{b, c\}^*$ be a function defined as in Example 1. The different sets of tails are:

$$\begin{aligned} T_t(\lambda) &= \{(\lambda, \lambda)\} \cup \{(a^{2n+1}, b^{2n+2}) | n \geq 0\} \\ &\quad \cup \{(a^{2n}, b^{2n}c) | n \geq 1\} \\ T_t(a) &= \{(\lambda, \lambda)\} \cup \{(a^{2n+1}, b^{2n}c) | n \geq 0\} \\ &\quad \cup \{(a^{2n}, b^{2n}) | n \geq 1\} \\ T_t(a^2) &= \{(\lambda, c)\} \cup \{(a^{2n+1}, b^{2n+2}) | n \geq 0\} \\ &\quad \cup \{(a^{2n}, b^{2n}c) | n \geq 1\} \\ T_t(a^{2n-1}) &= T_t(a), n \geq 1 \\ T_t(a^{2n}) &= T_t(a^2), n > 1 \end{aligned}$$

and the canonical OST that realizes t, which is shown in Fig. 2, is isomorphic with the corresponding transducer shown in Fig. 1. □

The transduction learning algorithm to be presented in the next section requires a finite sample of input–output pairs $T \subset (X^* \times Y^*)$, which is assumed to be nonambiguous or *single-valued*, i.e., $(x, y), (x, y') \in T \Rightarrow y = y'$. Such a sample can be properly represented by a "tree subsequential transducer" (TST) $\tau = (Q, X, Y, q_0, E, \sigma)$, with

$$\begin{aligned} Q &= \cup_{(u,v)\in T} Pr(u), q_0 = \lambda, \\ E &= \{(w, a, \lambda, wa) | w, wa \in Q\}, \\ \sigma(u) &= \begin{cases} v \text{ if } (u, v) \in T \\ \text{undefined otherwise.} \end{cases} \end{aligned}$$

A possible sequential representation τ' of this *TST* can be obtained by eliminating σ of τ and extending its set of states and edges as follows:

$$\begin{aligned} Q' &= Q \cup \{u\# | (u, v) \in T\}; \\ E' &= E \cup \{(u, \#, v, u\#) | (u, v) \in T\}. \end{aligned}$$

Whenever such a sequential representation is used, the training pairs (u, v) will be written as $(u\#, v)$.

Given T, an onward tree subsequential transducer (OTST) representing T, $OTST(T)$ can be obtained by building an OST equivalent to the TST of T. Let $\tau' = (Q', X \cup \{\#\}, Y, q_0, E')$ be a given (sequential representation) of a TST. In order to obtain the OTST equivalent to τ', the states of τ' must be considered in an orderly fashion, starting from the leaves and in each state $q \neq q_0$:

if $w = lcp\{v \in Y^* | (q, a, v, r) \in E'\} \wedge w \neq \lambda$ **then**

1) substitute every outgoing edge of q: (q, a, wz, r) for (q, a, z, r)

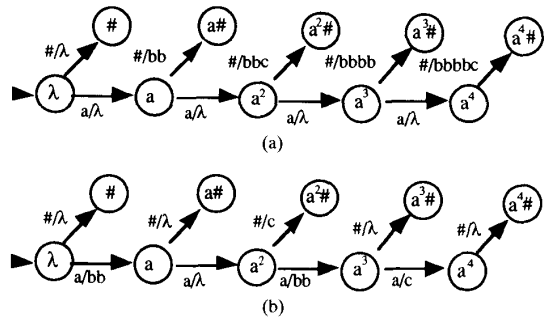


Fig. 3. Tree subsequential transducer (TST): (a) Onward tree subsequential transducer OTST(T); (b) representing the sample $T = \{(\#, \lambda), (a\#, bb), (aa\#, bbc), (aaa\#, bbbb), (aaaa\#, bbbbc)\}$.

2) substitute the ingoing edge of q: (p, b, y, q) for (p, b, yw, q).

Example 3: Let $T = \{(\#, \lambda), (a\#, bb), (aa\#, bbc), (aaa\#, bbbb), (aaaa\#, bbbbc)\}$. This sample has been drawn from the transduction in Example 1. Sequential representations of the TST of T and $OTST(T)$ are as shown in Fig. 3. □

IV. The Transducer Learning Algorithm

Let $T \subset (X^*\# \times Y^*)$ be a finite single-valued training set. The proposed algorithm consists of a nonincremental procedure that starts building the $OTST(T), \tau = (Q, X \cup \{\#\}, Y, q_0, E)$ and then proceeds by orderly trying the merge of states of τ. This state merging may result in transducers that are not subsequential as defined in Section II. In particular, they may often violate the condition of *determinism*: $(q, a, u, r), (q, a, v, s) \in E \Rightarrow (u = v \wedge r = s)$. In these cases, we may still insist on preserving the subsequential nature of the resulting transducers. For this to be possible, some output (sub)strings associated with the edges of τ often need to be "*pushed back*" towards the leaves of τ in a process that, to a limited extent, reverses the forward (sub)string displacements carried out to transform the initial TST associated with T into τ. The test as to whether a transducer τ is subsequential, as defined in Section II, is assumed to be supplied by a (hypothetical) procedure "*subseq*" that takes a transducer and outputs *true* if such a property holds or *false* otherwise. One should realize, however, that when embedded in the algorithm to be described below, such a test can be carried out at no cost as a byproduct of the remaining operations of the core of the algorithm.

The merging process mentioned above requires the states of τ to be successively taken into account in a lexicographic order of the names given to these states through the TST construction (Section III) [18]. Let "$<$" be such an order on Q, with $first(\tau)$ and $last(\tau)$ being the first and last states with respect to "$<$," and $\forall q \in Q - \{last(\tau)\}$, and let $next(\tau, q)$ denote the state that immediately follows q in the order "$<$." The merging of any two states $q', q \in Q$, with $q' < q$, results in a new transducer $\tau' = (Q', X \cup \{\#\}, Y, q_0, E')$ in which the state q no longer exists in Q', and all the outgoing edges of q in E are assigned to q' in E'. Let $\tau' = merge(\tau, q'q)$

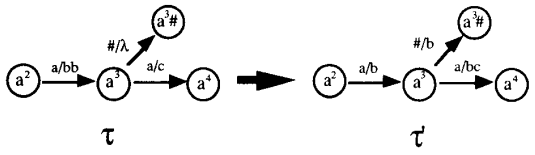


Fig. 4. Example of a push-back operation.

```
INPUT: Single-valued finite set of input-output pairs T⊂ (X*# x Y*)
OUTPUT: Onward Subsequential Transducer τ consistent with T
τ := OTST(T)
q:=first(τ )
while q < last(τ ) do
   q:= next(τ,q);
   p:= first(τ );
   while p < q do
        τ':= τ
        τ := merge(τ,p,q)
        while ¬subseq(τ ) do
           let (r,a,v,s), (r,a,w,t) be two edges of τ
             that violate the subseq condition, with s < t
           if ((v≠w) and (a=#)) or (s<q and v∉ Pr(w)) then exit while
           u:= lcp(v,w)
           τ := push_back(τ, u⁻¹v,(r,a,v,s))
           τ := push_back(τ, u⁻¹w,(r,a,w,t))
           τ := merge(τ,s,t)
        end while //¬subseq(τ )//
        if ¬subseq(τ ) then τ := τ' else exit while
        p:= next(τ ,p)
   end while // p < q //
   if ¬subseq(τ ) then τ := τ'
end while // q < last(τ)//
end //OSTIA//
```

Fig. 5. Transducer inference algorithm.

represent this merging operation that is quite similar to the merging of states in finite-state automata [16].

The process of "pushing back" output substrings in a transducer $\tau = (Q, X \cup \{\#\}, Y, q_0, E)$ requires more explanation. Let $q \in Q$ be a state of τ and $(q', a, w, q) \in E$ (one of) its ingoing edge(s). If $u \in Y^*$ is a prefix of w, then the suffix $v = u^{-1}w$ can be "pushed back" to behind q and distributed throughout all the outgoing edges of q to produce another transducer $\tau' = (Q, X \cup \{\#\}, Y, q_0, E')$ as follows (see Fig. 4):

$$E' = (E - \{(q', a, uv, q)\}) \cup \{(q', a, u, q)\} \cup \{(q, b, vz, r) : (q, b, z, r) \in E\}.$$

Let $\tau' = \textit{push\_back}\,(\tau, v, (q', a, w, q))$ denote this operation for which the following equivalence property can be easily established [18]:

If $v = \lambda$ or if (q', a, w, q) is the only edge that enters q and $a \neq \#$, then transductions are preserved, and the transducer τ' is equivalent to τ.

The algorithm that performs the above outlined procedures is called the "*onward subsequential transducer inference algorithm* (OSTIA)" and is formally presented in Fig. 5.

The computational cost of this algorithm can easily be shown to be polynomial. Let $n = \sum_{(x,y)\in T} |x|$, $m = \max_{(x,y)\in T} |y|$ and $k = |X|$ be parameters measuring the size of the input training set T. The initial TST of T has a number of states that is linear with n. The initial $OTST(T)$ has the same number of states, and its construction requires computing time that grows linearly with $n \cdot m \cdot k$.

The two outer **while** loops of OSTIA entail, in principle, a total number of iterations that is quadratic with the number of states of the initial transducer $\tau = OTST(T)$—hence, with n. This could be considered exceedingly pessimistic since many states could rapidly be eliminated from τ by successive *merge* operations. Nevertheless, a worst case can be identified in which, after having carried out all possible *merge* operations of the inner **while** loop and having exhausted all the remaining states, this loop is **exited** without success (**if** condition). In this case, the inner loop would perform, at most, $O(n)$ iterations, resulting in an $O(n^3)$ total number of executions of the core of the algorithm. Except for lcp and *push_back*, all the other operations involved in the OSTIA can be easily implemented to run with unit cost, that is, computing time independent of n, m, and k. The lcp operation can also be easily implemented to work in $O(m)$ time, whereas *push_back* may require $O(k)$ steps if string indexes rather than actual (sub)strings are used to represent the input–output substrings of the edges of τ. This amounts to a total cost in $O(m + k)$ for the core operations and a total computing time that can be bounded as $O(n^3(m + k) + nmk)$. Obviously, in many cases of interest, this bound is, in fact, pessimistic. Real empirical timing results will be shown in Section VI.

The correctness of this algorithm is also clear. The proof starts by realizing that the property $q < t$ results in being a loop invariant of the algorithm. Given the tree structure of the initial OTST, along with the order in which states are considered for merging, this invariant implies that only one edge can enter the state t. Correspondingly, from the above *equivalence property*, the *push_back* operation involving t is always transduction preserving. On the other hand, the other *push_back* operation, which involves the state s, can only be executed in two very precise situations: 1) $s < q$, in which case $v \in Pr(w)$ and thus $u^{-1}v = \lambda$ or 2) $q < s$, which implies that only one edge can enter s. Therefore, the equivalence property guarantees that, in both cases, the operation also results in being transduction preserving. Since the initial OTST is obviously consistent with T, proceeding by induction, we see that every iteration yields a transducer τ that is *subsequential* and *consistent* with T.

Similar arguments support the assertion that the resulting transducer is *onward*. The initial OTST is obviously onward, which is a property that is clearly preserved by the successive executions of the operation "$merge(\tau, p, q)$." However, the two *push_back* operations (which aim at trying to recover the subseq condition after its possible violation by the merge of two states) may result in losing the onward property. However, in this case, the subsequent "merge(τ, s, t)" operation directly restores the property, yielding again an onward transducer.

Following this discussion, it can be seen that the OSTIA produces a subsequential transducer that is a state-merging-based generalization of the initial subsequential tree transducer

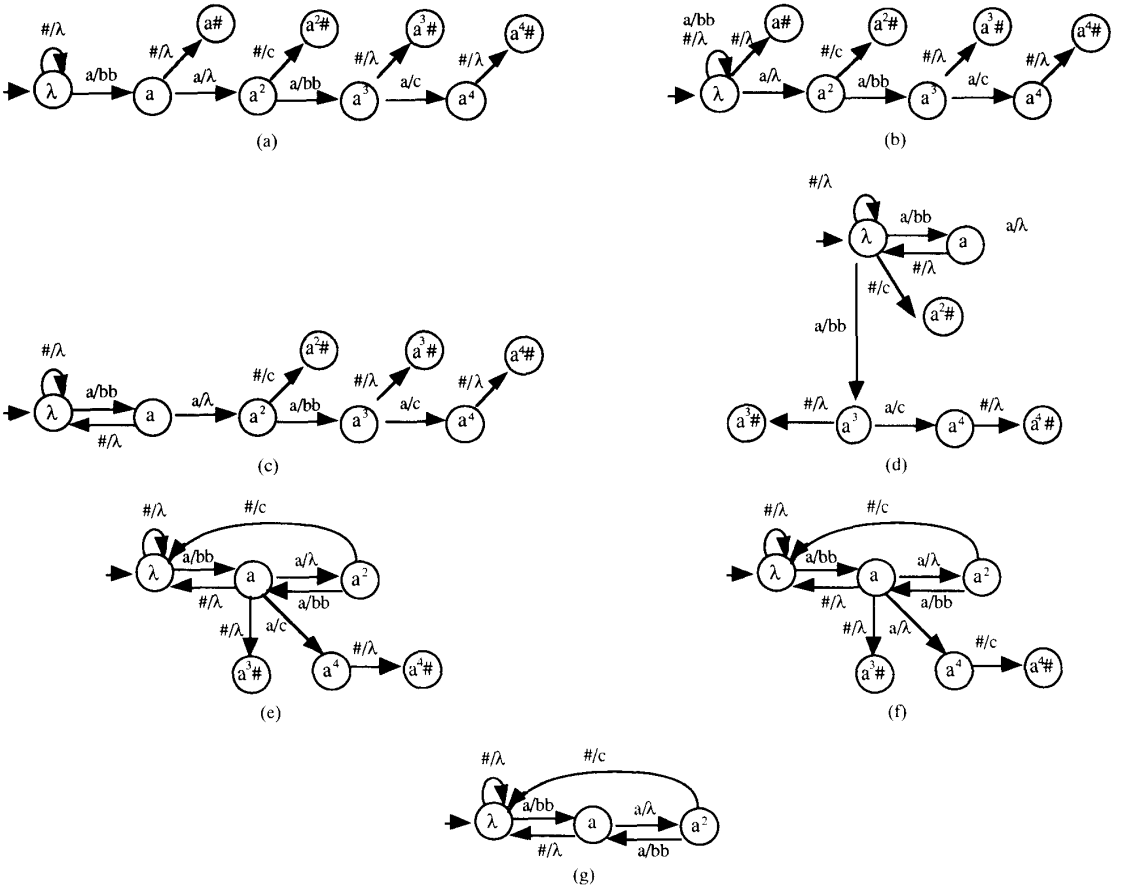


Fig. 6. Some key steps of the OSTIA as applied to the OTST in Fig. 3(b).

representing the given sample T. This generalization, however, tends to progressively shrink as T gets larger and includes more training pairs that are "representative" of the unknown (subsequential) transduction from which T is assumed to have been drawn. As will be discussed in the next section, this leads to the important result that *using the OSTIA, the class of subsequential functions can be identified in the limit.*

Using the training pairs of Example 3, Fig. 6 illustrates some key steps of the OSTIA. Starting from the OTST of Fig. 3(b), the algorithm attempts the merging of states λ and #. Fig. 6(a) shows the resulting transducer. Then, OSTIA goes on trying the merging of states a with λ (Fig. 6(b)). This transducer has the two edges $(\lambda, a, bb, \lambda)$ and $(\lambda, a, \lambda, aa)$ that violate the subseq condition. Given that $\lambda < a$ and, $bb \notin Pr(\lambda)$ the innermost loop of OSTIA is exited, and states λ and $a\#$ are merged in the previous transducer (Fig. 6(a)), resulting in the transducer of Fig. 6(c). The next step tries the merging of the states λ and aa. The existence of edges $(\lambda, \#, \lambda, \lambda)$ and $(\lambda, \#, c, aa\#)$ (Fig. 6(d)) causes the exit of the innermost loop of OSTIA. Similarly, the attempt of merging the states a and aa is unsuccessful. After merging the states λ with $aa\#$ and given the imposibility of merging λ with aaa, Fig. 6(e) shows the result of the merging of the states aaa with a. The edges (a, a, λ, aa) and $(a, a, c, aaaa)$ of the resulting transducer violate the subseq condition; nevertheless $lcp(\{\lambda, c\}) = \lambda$, and the operations $push\_back(\tau, \lambda, (a, a, \lambda, aa))$ and $push\_back(\tau, c, (a, a, c, aaaa))$ produce the transducer shown in Fig. 6(f). The rest of the work carried out by the innermost loop of OSTIA finally leads to the subsequential transducer of Fig. 6(g) that is the inferred result from the training sample of Example 3.

V. Identification of Subsequential Transductions in the Limit

In this section, we will show how the OSTIA identifies the class of subsequential functions from positive data (input–output pairs) in the limit. To this end, we start defining a finite representative sample T of a total subsequential function t. We will show that the transducer $OTST(T)$ contains a subset of states (referred to as kernel), which induces a subtree in $OTST(T)$ that contains representations of every and all the edges of the canonical OST of t, $\tau(t)$. This result will allow us to establish that if a representative sample is input to OSTIA, then the output will be $\tau(t)$. Given that any positive presentation of t will eventually include a representative sample, the property of the identification in the limit will result.

In what follows, only the main arguments of the proofs will be sketched. A full account of the complete results can be found in [18].

Definition 1: Let $t : X^* \rightarrow Y^*$ be a subsequential function. The string w is a short prefix of t iff $w \in Pr(Dom(t))$, and $\forall v \in X^*(T_t(w) = T_t(v) \Rightarrow v \geq w)$. The set of short prefixes of t is denoted $SP(t)$.

The set $SP(t)$ includes a short prefix for every different set of tails in $\{T_t(w)|w \in Pr(Dom(t))\}$. Given that the sets of tails are just the states of the canonical OST of t (Section III), the set of short prefixes constitutes a minimal set of shortest strings that allow reaching every state of $\tau(t)$.

Definition 2: Let $t : X^* \rightarrow Y^*$ be a subsequential function. The kernel of t is the set $K(t) = (SP(t)X \cap Pr(Dom(t))) \cup \{\lambda\}$.

In other words, $K(t)$ contains the empty string along with the extensions of all of the strings in $SP(t)$ with all the symbols in X, as long as these extensions are prefixes of strings in the domain of t.

Definition 3: Given a *total* subsequential function $t : X^* \rightarrow Y^*$, a *sample* T of t is said to be *representative* of t iff

1) $\forall u \in K(t) \quad \exists (v, w) \in T | u \in Pr(v)$
2) $\forall u \in SP(t) \quad \forall v \in K(t) \quad (T_t(u) \neq T_t(v) \Rightarrow \exists (uw, u'w'), (vw, v'w'') \in T |$
   $(w, w') \in T_t(u), (w, w'') \in T_t(v), w' \neq w'')$
3) $\forall u \in K(t) \quad \exists (uv, u'v'), (uw, u'w') \in T |$
   $(v, v'), (w, w') \in T_t(u), lcp(\{v', w'\}) = \lambda$.

Example 4: The set of short prefixes of the transduction t of Example 1 is $SP(t) = \{\lambda, a, aa\}$ (c.f., Example 2), and the kernel is $K(t) = \{\lambda, a, aa, aaa\}$. An example of a representative sample is $T = \{(\lambda, \lambda), (a, bb), (aa, bbc), aaa, bbbb), (aaaa, bbbbc)\}$ (c.f., Example 3). □

In the theorems given below, condition 1) of Definition 3 will guarantee that all the states and transitions of the canonical OST $\tau(t)$ are represented in $OTST(T)$. Condition 2) will distinguish those states of $OTST(T)$ for which the corresponding states of $\tau(t)$ are different. Finally, condition 3) will make it possible that for every state of $OTST(T)$ that is reachable by a string in the kernel of t, the edges will be identical to the corresponding edges in $\tau(t)$.

In the forthcoming theorems the following notation will be used: $\tau_T = (Q_T, X, Y, \lambda, E_T, \sigma_T)$, and $\tau(t) = (Q_t, X, Y, q_0, E_t, \sigma_t)$ will denote the $OTST(T)$ and the canonical OST of t, respectively; $\tau_i = (Q_i, X \cup \{\#\}, Y, \lambda, E_i)$ will denote the OST obtained after the ith iteration of the outermost while loop of the OSTIA with input $T \subset (X^*\# \times Y^*)$.

Theorem 1: Let T be a representative sample of $t : X^* \rightarrow Y^*$. A function $\varphi : Q_T \rightarrow Q_t$ exists such that $\forall u, ua \in K(t)$ $((u, a, v, ua) \in E_T \Leftrightarrow (\varphi(u), a, v, \varphi(ua)) \in E_t)$, and $\forall u \in K(t)$ $\sigma_T(u) = \sigma_t(\varphi(u))$.

Proof: If $u \in K(t)$ and $t' : X^* \rightarrow Y^*$ is the finite function realized by τ_T, the third condition for a representative sample (Definition 3) allows us to establish that $lcp(t'(uX^*)) = lcp(t(uX^*))$. Then, by defining $\varphi : Q_T \rightarrow Q_t$ as $\varphi(u) = T_t(u), u \in Q_T$ and using the first condition of representative sample (Definition 3), the theorem follows. □

Theorem 2: Let $q \in Q_i$ be the state of τ_i considered in the $i + 1$ iteration of the outermost while loop of OSTIA, let $Q'_i = \{q' \in Q_i | q' < q\}$, and let $\tau'_i = (Q'_i, X \cup \{\#\}, Y, \lambda, E'_i)$ be the subtransducer induced by Q'_i. If T is a representative sample of a total subsequential function $t : X^* \rightarrow Y^*$, then τ'_i is isomorphic with a sequential representation of a subtransducer of $\tau(t)$.

Proof: Let $\varphi : Q'_i \rightarrow Q_t$ be defined as $\varphi(u) = T_t(u), u \in Q'_i$. By induction in the number of iterations and given that the second condition of the definition of representative sample guarantees us that OSTIA will perform the merge of two states only if both have the same image with φ, we can establish that

a) $(u, a, v, ua) \in E'_i \Rightarrow (\varphi(u), a, v, \varphi(ua)) \in E_t$
b) $(u, \#, v, u\#) \in E'_i \Rightarrow \sigma_t(\varphi(u)) = v$

and, using an appropriate sequential representation, the theorem follows. □

Corolary 1: If the input to OSTIA is a representative sample of a total function t, then the output is an OST that is isomorphic with $\tau(t)$.

Theorem 3: Using the OSTIA, the class of subsequential functions can be identified in the limit with positive presentation.

Proof: If $t : X^* \rightarrow Y^*$ is a total function, then any positive presentation of t will eventually include a representative sample. Hence, OSTIA will converge to a subsequential transducer that is isomorphic with $\tau(t)$. If t is not total, the second condition of the definition of representative sample is not guaranteed. Correspondingly, OSTIA may merge states that are associated to states of $\tau(t)$ that are different from each other. However, in this case, no finite information sequence can effectively distinguish such states. Thus, OSTIA would converge to a subsequential transducer τ that realizes a function t' such that $\forall x \in Dom(t)$ $t'(x) = t(x)$, which is precisely the condition assessing the identification of t in the limit (Section II). □

VI. Experiments

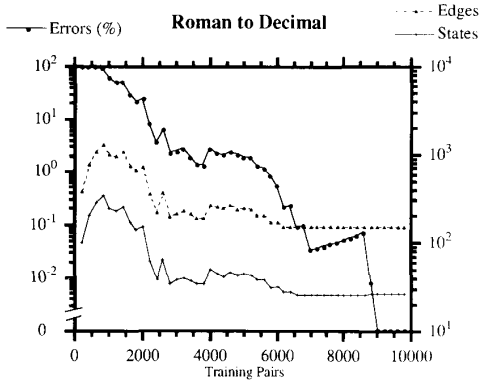
Theoretical properties assessing the adequate behavior in the limit (convergence) of subsequential transduction identification using the OSTIA have been discussed in the previous section. However, from PR viewpoint, results concerning finite data behavior seem to be of greater interest. In order to obtain such results and at the same time illustrate the capabilities of subsequential transduction and their OSTIA learning, a number of experiments have been carried out. Some of these experiments and their results will be presented in this section.

For the first group of experiments, the task of translating roman numerals into their decimal representation has been chosen. It should be taken into account that such a task can by no means be supported by pure sequential transduction since many input–output pairs exist in which identical input prefixes must lead to different output substrings. As will be seen below, subsequential transduction easily overcomes this difficulty, allowing the task to be properly accomplished.

For the first experiment, a series of 50 increasing-size random training sets, each including the previous ones, were

(III#, 3)	(LXXIV#,74)
(IV#, 4)	(XCV#, 95)
(VI#, 6)	(CXV#, 115)
(IX#, 9)	(CDII#, 402)
(XI#, 11)	(CMLXXXIX#,989)
(XIX#, 19)	(MCXI#, 1111)
(XLII#, 42)	(MMMMMMMMDCCCLXXXVIII#, 8888)

Fig. 7. Sample of selected training pairs for the roman to decimal translation task.



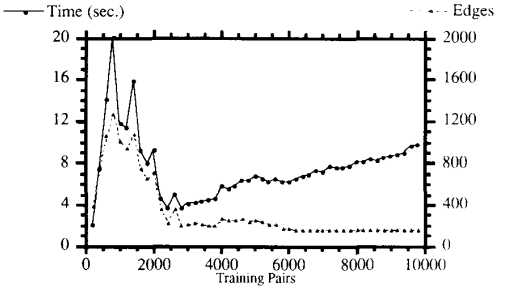


Fig. 8. Behavior of the OSTIA in the roman to decimal translation task.

drawn from a uniform distribution in the range of (roman and decimal) numbers from 1 to 10^4-1. The random procedure was prevented from generating repeated samples, and the test set consisted of all the roman numerals from 1 to 10^4-1 that were not included in the corresponding training sets. Some selected training pairs are shown in Fig. 7.

Every training set was submitted, in turn, to the OSTIA, and each resulting subsequential transducer was used to translate all the roman-numeral strings of the test set. The results of this experiment appear in Fig. 8. The items shown are a) the test-set error rates, b) the sizes of the learnt transducers, and c) the computing time required for each execution of the OSTIA, on a $\sim$ 25 mips conventional RISC computer (HP 9000/835).

It is worth noting that with small training sets, the inferred transducers tend to be rather large and error prone, whereas both sizes and errors reduce dramatically as enough source structure is made available through the training data. More specifically, learning can be considered almost completely

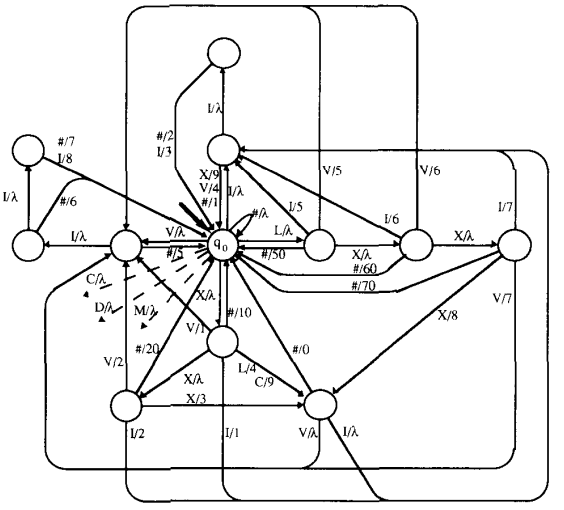


Fig. 9. Part of the subsequential transducer that was learned for the roman to decimal translation task. The part shown corectly translates the roman numbers from 1 to 99. The initial state is marked as "$q0$." and the dashed lines correspond to edges of the remaining parts (not shown) of the whole transducer.

accomplished (97% performance) after having seen approximately 3000 random training pairs.

It should be taken into account that these results were obtained without concern as to how "relevant" the (random) training data were for the considered learning task. The existence of a "minimal" relevant training set (representative sample) has been proven in the last section; however, without *a priori* knowledge of the source transduction, theory does not seem to provide us with adequate means to identify such a set in practice. In order to empirically investigate how small the training set could become if appropriately selected, an additional experiment was carried out involving the following *greedy procedure*. First, starting with a transducer learned from a first training pair "(I,1)" the test data was submitted in numerical order to transduction until one roman-numeral string that was incorrectly translated appeared. Then, this string, along with its correct decimal transduction, was used as a training pair. The first phase of this procedure stopped when all the strings were correctly translated. In the second phase, the roman numeral (training) pairs that were selected in the first phase were considered, in turn, to see whether they could be discarded without change in the inferred transducer. After this second phase, a representative sample of 210 roman-decimal pairs that led to the correct identification of an exact subsequential transducer that was identical to the one obtained in the previous experiment with a random training set of 9000 pairs was identified. A portion of this transducer is shown in Fig. 9.

A second group of experiments was concerned with the task of translating english numbers in the $0 \ldots 10^6$-1 range into their conventional decimal representation. Although this task is rather better suited to sequential transduction than that of the roman-decimal task, there are also in this case situations that prevent pure sequentiality, e.g., "ninety" $\rightarrow$ "90," "nineteen"

(nine#, 9)
(ninety#,90)
(nineteen#, 19)
(ninehundred#, 900)
(nineteenthousandandeight#, 19008)
(ninehundredthousandandeighteen#, 900018)
(twohundredandthirteenthousandandtwelve#, 213012)
(fourhundredandonethousandeighthundred#, 401800)
(eighthundredandeightthousandeighthundredandseven#, 808807)
(sixhundredandseventeenthousandsixhundredandsixteen#, 617616)
(threehundredandninetyfivethousandonehundredandsixtythree#, 395163)
(sevenhundredandeighteenthousandsevenhundredandseventeen#, 718717)

Fig. 10. Sample of selected training pairs for the english to decimal translation task.

→ "19," etc. The experimental framework was similar to that of the previous experiments, except that in this case, the numbers were randomly drawn from a nonuniform distribution in which the lengths of the decimal strings were (approximately) equiprobable. In this case, 60 training sets of sizes increasing from 200 up to 12 000 were used, and the test set consisted of the whole range of 10^6 different english-number strings except those used for training. In order to approach the conditions of phonemic recognition, these strings were written with no separating blanks between the different words, which considerably increased the difficulty of the task. Fig. 10 shows some training pairs that have been selected from the training material. The results appear in Fig. 11, which shows the same items as Fig. 8, with similar behavior. In this case, learning can be considered almost completely accomplished (97% performance) after having seen approximately 2500 random training pairs. In addition, a greedy procedure similar to the one described above was carried out, ending with a representative sample of 260 input–output pairs with which a perfect transducer is obtained that is exactly the same as that obtained from 11 600 random training pairs.

Apart from these experiments, many other similar experiments involving these and many other tasks were carried out. For instance, number translation from english to spanish and from spanish to decimal were successfully, learned from english–spanish pairs and spanish–decimal pairs, respectively, even though spanish number rules entail different nonsequentialities and are more involved than the corresponding rules in english [21]. In addition, transducers that implement integer division of a decimal number by an arbitrary divisor and (reversed) multiplication by an arbitrary factor were easily learned with the OSTIA from data-result examples [19]. The behavior of the OSTIA, which is observed in all these experiments, was rather similar to that shown in the experiments described above. The presentation of these experiments has been omitted here for the sake of brevity.

The last group of experiments was aimed at establishing some empirical worst-case results on the computational requirements of the OSTIA. Although a worst-case polynomial behavior was easily deduced from the properties of the OSTIA (Section IV), a tighter upper bound seems rather difficult to establish theoretically. Nevertheless, it should be clear that worst-case behavior would be exhibited for training data

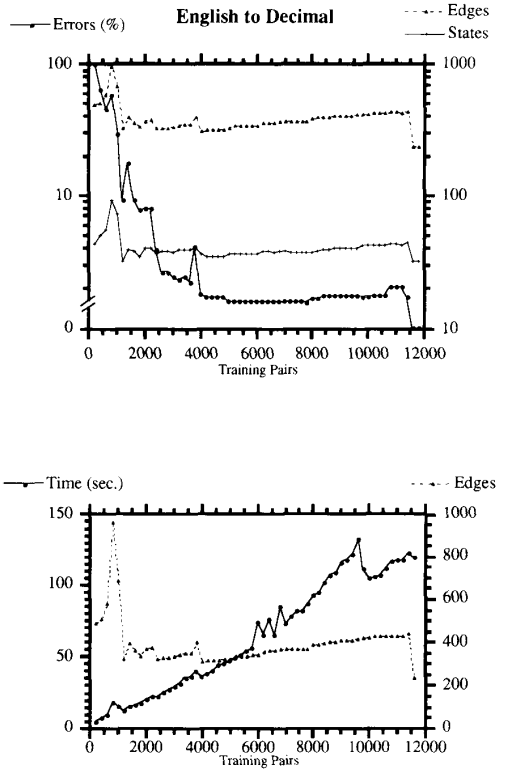


Fig. 11. Behavior of the OSTIA for the english to decimal translation task.

corresponding to nonsubsequential transduction. In other case, when enough source transducer structure is represented in the training data, the state merging process (quickly) reduces the (large) number of states of the initial OTST, leading to the quasi-linear, low-computing-time growing rates shown in Figs. 8 and 11 for sufficiently large training sets. Therefore, in order to empirically investigate the OSTIA computing-time growth, an experiment involving nonsubsequential transduction was carried out.

The chosen task was that of reversing strings (representing decimal numbers in the 10^4 range), which by no means might properly be accomplished by subsequential transduction. Using a similar experimental setup as in the previous experiments, the OSTIA was presented with 20 training sets of sizes increasing from 200 up to 4000. Moreover, in order to (as much as possible) approach an absolute worst-case data conditioning for the OSTIA, the training data were presented in strict number ordering within each training set. Obviously, in this case, no appropriate transducer was ever obtained, leading to a 100% error rate for those strings not used for training. The computing times required by the OSTIA in this case are presented in Fig. 12 along with the corresponding sizes of the inferred transducers. A second-degree polynomial that was least-squared fitted to the experimental values is also shown. It is worth noting that although a first-degree fit would clearly be inappropriate, the quadratic coefficient of the fitted polynomial is relatively small, and for the range of sizes considered, a cubic coefficient would have not improved the fit.

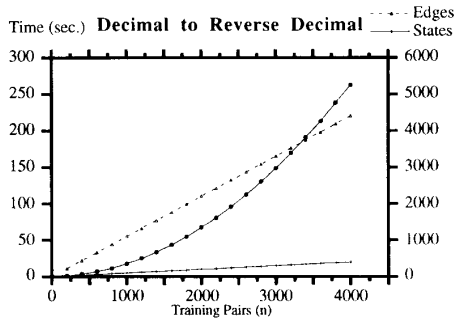


Fig. 12. Worst-case behavior of the OSTIA as applied to a nonsubsequential transduction learning task. A second-degree polynomial: Time $= 1.6 \cdot 10^{-5} n^2 + 0.00013n + 0.2595$ is least squared fitted to the computing times.

VII. Discussion and Conclusion

The results of the experiments described in the last section clearly indicate both the versatility of subsequential transduction and the effectiveness of the OSTIA to learn subsequential transducers from training input–output examples. For those (small) training sets that do not convey enough structure of the unknown source transduction, the transducers produced by OSTIA tend to be rather large and error prone; however, very compact and accurate solutions are always obtained once the training data contain a very small number of relevant input–output pairs. The existence of such small sets of relevant training data was shown through some of the experiments reported in the last section, and some theoretical results regarding what a "relevant" set of input–output training pairs is (*representative sample*) were presented in Section V. However, the only practical hint these results seem to suggest is that such training data should contain the "simplest" (usually also the shortest) transduction examples, and how to actually choose adequate and small sets of training data remains an open issue of practical concern. In any case, by relying on chance alone, good results (i.e., low error rates) tend to be obtained with reasonably small sets of training pairs.

Another important issue is related to the *partial-function* nature of the transductions dealt with by OSTIA. Since no (positive) sample can effectively help teaching how to "correctly" translate ("incorrect") strings that are not in the domain of a function of this type, any learning device is granted the freedom of conveniently generalizing the training data by means of allowing arbitrary translation of these strings. For instance, the OSTIA-learned transducer of Fig. 9 outputs the string "84" in response to the input string "VIIIIV," which is not a correct roman numeral and therefore is not expected to be submitted for transduction in the test phase. Obviously, since this string is not in the domain of the transduction to be learned, it could never have the chance to appear in any (positive) presentation, thus impeding learning to yield just the string "*error*" as a response. Apart from the rather obvious use of such a type of "negative" input–output examples (with output = "*error*"), we may try to overcome this problem by attempting an identification (or at lest an adequate restriction) of the function domain, either by using a priori knowledge or by means of more conventional grammatical interference methods. Although we have not yet fully pursued this approach, we have reason to believe that any available knowledge of the transduction domain could significantly help not only to produce more "natural" transduction results in undefined cases but also to make the learning task being considered even easier.

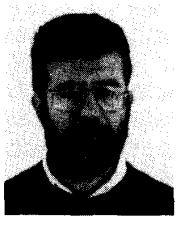
Apart from training-data selection and domain definition, two other interesting (practical) issues remain to be investigated. First, since the OSTIA is essentially a nonincremental learning algorithm, the possibility of small *incremental adaptation* to new training data of a transducer that was already fairly well established from previous (nonincremental) OSTIA learning should be worth studying. Second, the possibility of incorporating an appropriate (also learned) *error model* into the learned transducers, as well as making the resulting transducers *stochastic*, needs be investigated if dealing with real and natural data such as speech or images is required.

In any case, we think that the work presented in this paper constitutes a required step that would eventually allow many real *interpretation tasks* to be dealt with under the *transduction framework* of syntactic pattern recognition.

References

[1] R. C. González and M. G. Thomason, *Syntactic Pattern Recognition, An Introduction*. MA: Addison-Wesley, 1978.
[2] K. S. Fu, *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[3] L. Miclet, *Structural Methods in Pattern Recognition*. Oxford, UK: North-Oxford Academic, 1986.
[4] R. de Mori (chairman) *et al.*, "Report of the working group B: Waveform and speech recognition," in *Proc. NATO Adv. Res. Workshop Syntactic Structural Patt. Recogn.*, (Barcelona), Oct. 1986; in *Syntactical and Structural Pattern Recognition*. (G. Ferrate, T. Pavlidis, A. San Feliu, and H. Bunke, Eds.). New York: Springer-Verlag, 1988, pp. 446–451.
[5] J. C. Simon, E. Backer, and J. Sallantin, "A structural approach to pattern recognition," *Signal Processing*, no. 2, pp. 5–22, 1980.
[6] J. Berstel, *Transductions and Context-Free Languages*. Stuttgart: Teubner, 1979.
[7] E. M. Gold, "Complexity of automation identification from given data" *Inform. Contr.*, vol. 37, pp. 302–320, 1978.
[8] L. P. J. Velenturf, "Inference of sequential machines from sample computation," *IEEE Trans Comput.*, vol. 27, pp. 167–170, 1978.
[9] P. Luneau, M. Richetin, and C. Cayla, "Sequential learning from input–output behavior," *Robotica*, vol. 1, pp. 151–159, 1984.
[10] Y. Takada, "Grammatical inference for even linear languages based on control sets," *Inform. Processing Lett.*, vol. 28, no. 4, pp. 193–199, 1988.
[11] E. Vidal, P. García, and E. Segarra, "Inductive learning of finite-state transducers for the interpretation of unidimensional objects," in *Structural Pattern Analysis*, (R. Mohr, T. Pavlidis, and A. San Feliu, Eds.). New York: World Scientific, 1990, pp. 17–35.
[12] E. M. Gold, "Language identification in the limit," *Inform. Contr.*, vol. 10, pp. 447–474, 1967.
[13] K. S. Fu and T. L. Booth, "Grammatical inference: Introduction and survey," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-5, pp. 95–111; 409–423, pt. 1, 2 1975.
[14] D. Angluin and C. H. Smith, "Inductive inference: Theory and methods," *Comput. Surveys*, vol. 15, no. 3, pp. 237–269, 1983.
[15] L. Miclet, "Grammatical inference," in *Syntactic and Structural Pattern Recognition* (H. Bunke and A. San Feliu, Eds.). New York: World Scientific, 1990, pp. 237–290.
[16] D. Angluin, "Inference of reversible languages," *J. ACM*, vol. 29, no. 3, pp. 741–765, 1982.
[17] P. García and E. Vidal, "Inference of k-testable languages in the strict sense and application to syntactic pattern recognition" *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 12, no. 9, pp. 920–925, 1990.
[18] J. Oncina and P. García, "Inductive learning of subsequential functions," Univ. Politécnia de Valencia, Tech. Rep. DSIC II-34, 1991.

[19] A. M. Corbí. "Estudio de un algoritmo de inferencia de transductores subsecuenciales," Facultad de Informática, Univ. Politécnica de Valencia, Proyecto fin de carrera, 1991.
[20] P. García, E. Vidal, and F. Casacuberta. "Local languages, the successor method, and a step towards a general methodology for the inference of regular grammars," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9 no. 6, pp. 841–845, 1987.
[21] J. Oncina, P. García, and E. Vidal, "Transducer learning in pattern recognition," in *Proc 11th IAPR Int. Conf. Patt. Recogn.*, 1992.
[22] J. Oncina, "Aprendizaje de lenguajes regulares y funciones subsecuenciales," Ph.D. dissertation, Univ. Politécnica de Valencia, 1991.



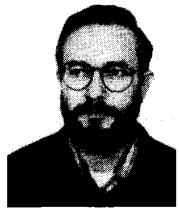
José Oncina received the Licenciatura in Ciencias Físicas degree in 1986 from the University Complutense of Madrid, Spain, and the Doctor of computer science degree in 1991 from the Polytechnic University of Valencia (UPV), Spain.

Since 1987, he has been with the Department of Informatic Systems and Computation (DSIC) at the UPV and has been teaching in the Computer Science College of the University. He is also a member of the Pattern Recognition and Artificial Intelligence Research group in the DSIC, where he is working on several scientific projects. He is co-author of a number of scientific publications in journals and congresses on the subjects of algorithmic learning and grammatical inference.

Pedro García received the Licenciado in Ciencias Físicas degree in 1976 from the University of Valencia and the Doctor of computer science degree in 1988 from the Polytechnic University of Valencia (UPV).

In 1987, he joined the Department of Informatic Systems and Computation (DSIC) of the UPV and has been teaching in the Computer Science Faculty of the University since that time. He is a member of the Pattern Recognition and Artificial Intelligence research group in the DSIC, where he is working on several scientific projects. His current fields of interest include computational learning theory and their application to syntactic pattern recognition.

Dr. García is a member of the ACM, the EATCS, and the Spanish Association for Pattern Recognition and Image Analysis (AERFAI), which is an affiliate society of the IAPR.



Enrique Vidal received the Licenciado en Ciencias Físicas degree in 1978 and the Doctor en Ciencias Físicas degree in 1985, both from the University of Valencia.

From 1972 to 1978, he was with several companies, working in electronics and computer engineering. In 1978, he joined the Computer Center of the University of Valencia, where he served as a systems analyst, and in 1981, he also joined the Department of Electronics and Informatics of the same university as an honorary collaborator. Since then, he coordinated, in both centers, a research group in the field of automatic speech recognition. In 1986, he joined the Department of Informatics Systems and Computation (DSIC) of the Polytechnic University of Valencia and has been chair professor in the Computer Science Faculty of the University. His current fields of interest include statistical and syntactic pattern recognition and their applications to automatic speech recognition, where he is especially concerned with grammatical inference and, in general, with automatic learning methodologies.

Dr. Vidal is a member of the International Association for Pattern Recognition (IAPR) and the Spanish Association for Artificial Intelligence (AEPIA). He also serves as a member of the governing board of the Spanish Association for Pattern Recognition and Image Analysis (AERFAI), which is an affiliate society of the IAPR.