Using domain information during the learning of a subsequential transducer*

Jose Oncina and Miguel Angel Varó**

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante, E-03071 Alicante (Spain)
e-mail: oncina@dlsi.ua.es

Abstract. The recently proposed OSTI algorithm allows for the identification of subsequential functions from input–output pairs. However, if the target is a partial function the convergence is not guaranteed. In this work, we extend the algorithm in order to allow for the identification of any partial subsequential function provided that either a negative sample or a description of the domain by means of a deterministic finite automaton is available.

1 Introduction

The problem of transducer learning has seldom been studied in the literature. Except for the particular restricted approach in Gold's paper [Gold,78], only very limited or heuristic work seems to have been carried out so far [Vel,78] [LRC,84] [Tak,88] [VGS,90].

In 1991 Oncina and collaborators [OGV,93] [OG,91] [Onc,91] presented a first step towards a formalization of the *transducer learning* (TL) problem from the point of view of its applicability within the *interpretation paradigm*. The framework used was similar to that of *grammatical inference* (GI) [Gold,67] [AS,83], though there are important differences between GI and TL. In particular, the work showed important analogies with the so-called *characterizable* methods in GI [AS,83]. [Ang,82] [GV,90].

In this case, the characterizable class is the set of the *subsequential transductions* which is a subclass of the most general *rational* or *finite-state transductions* but properly contains the class of *sequential transductions* [Ber,79]. A *sequential transduction* is defined as the one that preserves the prefixes of input–output strings. Although this can be considered as a rather natural property of transductions, there are many real-world situations where such a strict sequentiality is inadmissible. The class of *subsequential transductions* makes this restriction milder and may therefore be applied in a wider range of practical situations.

* This work has been partially supported by the Spanish CICYT under contract TIC93-0633-C02 & TIC95-0984-C02-01

** Miguel A. Varó is supported by a postgraduate grant from the *Generalitat Valenciana* (local government)

In [OGV,93] only subsequential transductions defined by total functions are considered in order to show the identifiability in the limit. It is not difficult to prove that identification in the limit of partial functions from positive samples is equivalent to identification in the limit of regular languages from only positive data, which is not generally possible. Thus, some additional information is necessary in order to achieve identification. In the following we will explore the use of negative information about the input language (*domain*). Another possibility is that a description of the domain is available. The last approach does not seem to be very useful from a practical point of view but sometimes, a good description of the domain language can be obtained by inferring an automaton from the input part of the training set [GOV,90]. In such a case, we do not have an exact description of the domain as required for identification in the limit but, as shown by some experimental results [Varó,94] [OCVJ,94] [CGV,94], it accelerates learning and, more importantly, it reduces the error rate. Therefore, this may also be seen as a way to introduce heuristic information in the inference algorithm.

2 Mathematical background and notation

Let X be a finite set or *alphabet*, and X^* the set of all strings over X. The *set of prefixes* of a string x is denoted as $\mathrm{Pr}(x)$, and can be extended as $\mathrm{Pr}(L) = \bigcup_{x \in L} \mathrm{Pr}(L)$. The *right quotient* of u and v is $u^{-1}v = w$ such that $v = uw$. The symbol $\oplus$ denotes symmetric difference. A *language* L is any subset of X^*. The *longest common prefix* of a language L is denoted as $\mathrm{lcp}(L)$.

A *finite automaton* (FA) A is defined by a 5-tuple (Q, X, δ, q_0, F) where Q is a finite set of *states*, q_0 is the *initial* state, $F \subseteq Q$ is the set of *accepting* states and $\delta : Q \times X \to 2^Q$ is a partial function. A FA is deterministic (DFA) if $\forall q \in Q$ and $\forall a \in X, \delta(q, a)$ has at most one element. With a DFA this function can be extended as $\delta : Q \times X^* \to Q$, where $\forall q \in Q, \delta(q, \lambda) = q$ and $\delta(q, xa) = \delta(\delta(q, x), a)$.

A *transduction* from X^* to Y^* is a relation $t \subseteq (X^* \times Y^*)$. In what follows, only those transductions which are (partial) *functions* from X^* to Y^* will be considered. It will be assumed that $t(\lambda) = \lambda$. For a partial function $t : X^* \to Y^*$, the *domain* of the function is $\mathrm{Dom}(t) = \{x | (x, y) \in t\}$. The domain function can be extended as $\forall u \in X^*, \mathrm{Dom}_t(u) = \{x | (ux, y) \in t\}$, then $\mathrm{Dom}(t) = \mathrm{Dom}_t(\lambda)$.

A *sequential transducer* is defined as a 5-tuple $\tau = (Q, X, Y, q_0, E)$, where Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, X is the input alphabet, Y is the output alphabet and E is a finite subset of $(Q \times X \times Y^* \times Q)$ whose elements are called *edges*.

A *path* in a sequential transducer τ is a sequence of edges $\pi = (q_0, x_1, y_1, q_1)$ $(q_1, x_2, y_2, q_2) \ldots (q_{n-1}, x_n, y_n, q_n), q_i \in Q, x_i \in X, y_i \in Y^*, 1 \le i \le n$, denoted also as $\pi = (q_0, x_1 x_2 \ldots x_n,\ y_1 y_2 \ldots y_n, q_n)$. Let $\prod_\tau$ be the set of all possible paths over τ. The transduction realized by a sequential transducer τ is the partial function $t : X^* \to Y^*$ defined as $t(x) = y$ iff $\exists (q_0, x, y, q) \in \prod_\tau$.

A *subsequential transducer* is defined as a 6-tuple $\tau = (Q, X, Y, q_0, E, \sigma)$, where $\tau' = (Q, X, Y, q_0, E)$ is a sequential transducer and $\sigma : Q \to Y^*$ is a (partial) function assigning output strings to the states of τ'. The partial function $t : X^* \to Y^*$ that is realized by τ is defined as $t(x) = y\sigma(q)$ iff $\sigma(q)$ is defined and $(q_0, x, y, q) \in \prod_{\tau'}$. The class of subsequential functions properly contains the class of sequential functions. It can be easily shown that if t is a subsequential function then $\mathrm{Dom}(t)$ is a regular language.

By using an additional input symbol $\# \notin X$ to mark the end of the input strings, any subsequential transduction $t : X^* \to Y^*$ can be obtained as a restriction to $X^*\#$ of a sequential transduction $t' : (X \cup \{\#\})^* \to Y^*$, so that, $\forall x \in X^*\ t(x) = t'(x\#)$ [Ber,79]. 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.

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 possible. Formally, a subsequential transducer $\tau' = (Q, X, Y, q_0, E, \sigma)$ is an OST if $\forall p \in Q - \{q_0\}\ \mathrm{lcp}(\{y \in Y^* | (p, a, y, q) \in E\} \cup \{\sigma(p)\}) = \lambda$. For any subsequential transduction there exists a *canonical OST* which has a minimum number of states and is unique up to isomorphism [OG,91]. This transducer is obtained through the following construction:

Let $t : X^* \to Y^*$ be a partial function and let $x \in \mathrm{Pr}(\mathrm{Dom}(t))$. The *set of tails* of x in t is $T_t(x) = \{(y, v) | t(xy) = uv, u = \mathrm{lcp}(t(xX^*))\}$ with $T_t(x) = \emptyset$ if $x \notin \mathrm{Pr}(\mathrm{Dom}(t))$.

It can be seen [OG,91] that, if t is subsequential, the number of different sets of tails in t is finite. Therefore, these sets can be used to build a subsequential transducer $\tau(t) = (Q, X, Y, q_0, E, \sigma)$ as follows:

$$Q = \{T_t(x) | x \in \mathrm{Pr}(\mathrm{Dom}(t))\}, \quad q_0 = T_t(\lambda)$$
$$E = \{(T_t(x), a, \mathrm{lcp}(t(xX^*))^{-1}\mathrm{lcp}(t(xaX^*)), T_t(xa)) | T_t(x), T_t(xa) \in Q\}$$
$$\sigma(T_t(x)) = \mathrm{lcp}(t(xX^*))^{-1} t(x) \text{ if } x \in \mathrm{Dom}(t), \text{undefined otherwise}$$

Following, [Gold,67] and [AS,83], an appropriate framework for the transducer learning problem can be established as follows: Let $f : X^* \to Y^*$ be a partial recursive function; a transducer learning algorithm A is said to *identify f in the limit* if, for any (positive) presentation of the input–output pairs of f, A converges to a transducer τ that realizes a function $g : X^* \to Y^*$ such that $\forall x \in \mathrm{Dom}(f), g(x) = f(x)$.

3 The transducer learning algorithm

The OSTI algorithm requires a finite sample of input–output pairs $T \subset (X^* \times Y^*)$, which is assumed to be non-ambiguous 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 $Q = \cup_{(u,v)\in T}\mathrm{Pr}(u)$, $q_0 = \lambda$, $E = \{(w, a, \lambda, wa) | w, wa \in Q\}$, $\sigma(u) = v$ if $(u, v) \in T$ and being undefined otherwise. This kind of TST is said to be *consistent* with T.

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

$$Q' = Q \cup \{u\#|(u,v) \in T\}; E' = E \cup \{(u, \#, v, u\#)|(u,v) \in T\}$$

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* representing T, $\mathrm{OTST}(T)$, can be obtained by building an OST equivalent to the TST of T. This can be accomplished through the following construction (directly given for a sequentially represented TST $\tau' = (Q', X \cup \{\#\}, Y, q_0, E')$): $\forall q \in Q', \mathbf{if}\ w = \mathrm{lcp}\{v \in Y^*|(q,a,v,r) \in E'\} \wedge w \neq \lambda$ **then** : **1:** substitute every outgoing edge of q: (q, a, wz, r) by (q, a, z, r); **2:** substitute every ingoing edge of q: (q, b, y, q) by (p, b, yw, q).

The OSTI algorithm consists of a non-incremental procedure that starts building the $\mathrm{OTST}(T)$, $\tau = (Q, X \cup \{\#\}, Y, q_0, E)$, and then proceeds by orderly trying to merge states of τ. This state merging may result in transducers which are not subsequential. 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 to preserve the sequential nature of the resulting transducers. For this to be possible, some output (sub)strings associated to the edges of τ often need to be *pushed back* towards the leaves of τ in a process which, to a limited extent, reverses the forward (sub)string displacements carried out to transform the initial TST associated to T into τ. The test as to whether a transducer τ is subsequential 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.

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. Let "$<$" be such an order on Q, with $\mathrm{first}(\tau)$ and $\mathrm{last}(\tau)$ being the first and last states with respect to "$<$", and $\forall q \in Q - \{\mathrm{last}(\tau)\}$ let $\mathrm{next}(\tau, q)$ denote the state which 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 $\tau' = \mathrm{merge}(\tau, q', q)$ represent this merging operation which is quite similar to the merging of states in finite-state automata [Ang,82].

The process of *pushing back* output substrings in a transducer $\tau = (Q, X \cup \{\#\}, Y, q_0, E)$ requires some more detailed 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 a state 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:

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

Let $\tau' = \mathrm{push\_back}(\tau, v, (q', a, uv, q))$ denote this operation, for which the following *equivalence property* can be easily established [OG,91]: if $v = \lambda$ or if

(q', a, w, q) is the *only* edge that enters q, then transduction are preserved and the transducer τ' is *equivalent* to τ.

```
Algorithm OSTIA // Onward subsequential transducer algorithm
      INPUT: Single-valued finite set of input–output pairs T ⊂ (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 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^{-1}v, (r, a, v, s))
               τ := push_back(τ, u^{-1}v, (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 //
      end while // q <last(τ) //
end // OSTIA //
```

Fig. 1. *The transducer inference algorithm*

The algorithm that performs the procedures outlined above is called *Onward Subsequential Inference Algorithm* (OSTIA) and is formally presented in Figure 1.

4 Identification of partial subsequential functions

The proof that OSTIA can identify the class of the *total* subsequential functions from positive data (input–output pairs) in the limit starts defining a *finite representative sample* T of a total subsequential function t. It can be shown that the transducer $\mathrm{OTST}(T)$ contains a subset of states, referred to as *kernel*, which induces a subtree in $\mathrm{OTST}(T)$ that contains representations of all the edges of the canonical OST of t, $\tau(t)$, and only these edges. This result allows us to establish that, if a representative sample is contained in the input to OSTIA, then the output will be the canonical OST of the function t. Given that any positive presentation of t will eventually include a representative sample, the property of the *identification in the limit* will result [OG,91].

Definition 1. Let $t : X^* \to Y^*$ be a subsequential function. The string w is a *short prefix* of t iff $w \in \mathrm{Pr}(\mathrm{Dom}(t))$ and $\forall v \in X^*(T_t(w) = T_t(v) \Rightarrow v \geq w)$. The *set of short prefixes* of t, $SP(t)$, includes a short prefix for every different tail in $\{T_t(w) | w \in \mathrm{Pr}(\mathrm{Dom}(t))\}$.

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

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

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

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

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

For total functions, a representative set can always be found. In particular, a set that fulfills condition 2 can be found because $\forall u \in X^*$, $\mathrm{Dom}(u) = X^*$ and then, if $T_t(u) \neq T_t(v)$ is because $\exists w \in X^*$ such that $(w, w') \in T_t(u), (w, w'') \in T_t(v)$ but $w' \neq w''$.

For partial functions, a set that fulfills conditions 1 and 3 can always be found. However, there exist transducers for which it is impossible to find a set that fulfills condition 2). For a partial function, $(T_t(u))$ can be different from $T_t(v)$ but $\forall w \in \mathrm{Dom}(u) \cap \mathrm{Dom}(v) (w, w') \in T_t(u) \Leftrightarrow (w, w') \in T_t(v)$, just because $\exists w \in \mathrm{Dom}(u)$ such that $w \notin \mathrm{Dom}(v)$ or vice versa. In such a case, we will say that the two states are not *distinguishable by any transduction*.

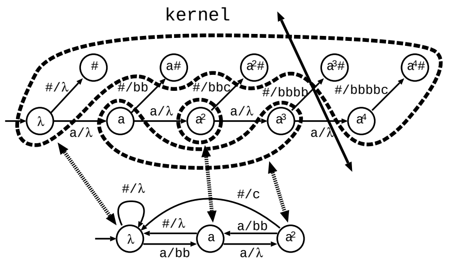

Fig. 2. *Relation between the states of the canonical OST and the OTST of the sample of the example 1.*

Example 1: Let $t : \{a\}^* \to \{b, c\}^*$ be a total subsequential function defined by: $t(a^n) = \lambda$ if $n = 0$, b^{n+1} if n is odd and $b^n c$ if n is even, and let $T = (\lambda, \lambda), (a, bb), (aa, bbc), (aaa, bbbb), (aaaa, bbbbc)$ be a sample drawn from t.

Then, all the different states of the kernel are distinguishable by the training T (fig. 2). In this case the kernel is $K(t) = \{\lambda, a, aa, aaa\}$, as $\{(\lambda, \lambda), (a, bb)\} \subset T_{\mathrm{OTST}}(\lambda)$, $\{(a, c)\} \subset T_{\mathrm{OTST}}(a)$ and $\{(\lambda, c), (a, bb)\} \subset T_{\mathrm{OTST}}(aa)$ these three states are distinguishable. States λ and aaa are not distinguishable, but as they represent the same state, no training set can be found that forces the distinguishability of both. Moreover, all the edges of the canonical OST of t are represented on the OTST made with T.

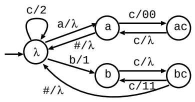

Fig. 3. *Transducer with states (a and b) not distinguishable by any transduction.*

Example 2: Let $t : \{a, b, c\}^* \to \{0, 1, 2\}^*$ the partial subsequential function (fig. 3) defined by: $t(x) = \lambda$ if $x = \lambda$, $2^m 0^{2n}$ if $x = c^m ac^{2n}$, $2^m 1^{2n+1}$ if $x = c^m bx^{2n+1}$, and $\emptyset$ otherwise. Then as $T_t(a) = \{(c^{2n}, 0^{2n})\}$ and $T_t(b) = \{(c^{2n+1}, 1^{2n})\}$ no training pair can exist that allow the distinguibishality of both.

If a method for distinguishing such states can be found then the same technique used for the proof of the convergence in the limit of OSTIA can be used. Obviously, for distinguishing such states additional information must be used. In the following sections we will explore the possibilities offered by the use of negative samples or the knowledge of the domain.

5 Identification using negative information

As we have seen before, on a partial function, $T_t(u)$ can be different from $T_t(v)$ but $\forall w \in \mathrm{Dom}(u) \cap \mathrm{Dom}(v)(w, w') \in T_t(u) \Leftrightarrow (w, w') \in T_t(v)$, just because $\exists w \in \mathrm{Dom}(u)$ such that $w \notin \mathrm{Dom}(v)$ or vice versa.

Such states can be distinguished if we have information about strings not belonging to the domain, i.e. if we know that $vw \notin \mathrm{Dom}(t)$ and $(uw, u'w') \in t$ then the states $T_t(u)$ and $T_t(v)$ must be different.

Definition 4. $S_- \in X^*$ is a negative sample of the partial function $f : X^* \to Y^*$ if $S_- \subset X^* - \mathrm{Dom}(f)$.

Then, for partial subsequential functions the representative sample must be defined as follows.

Definition 5. Given a *partial* subsequential function $t : X^* \to Y^*$, the pair (T, S_-) is said to be *representative* of t iff:

1. $\forall u \in K(t)\ \exists (uv, w) \in T \mid v \in X^*, w \in Y^*$
2. $\forall u \in SP(t)\ \forall v \in K(t)$ such that $T_t(u) \neq T_t(v) \Rightarrow$
   $(\exists (uw, u'w'), (vw, v'w'') \in T | (w, w') \in T_t(u), (w, w'') \in T_t(v), w' \neq w'') \vee$
   $(\exists uw \in S_-, (vw, v'w') \in T | w \in \mathrm{Dom}(v) - \mathrm{Dom}(u), (w, w') \in T_t(v)) \vee$
   $(\exists vw \in S_-, (uw, u'w') \in T | w \in \mathrm{Dom}(u) - \mathrm{Dom}(v), (w, w') \in T_t(u))$

3. $\forall u \in K(t) \exists (uv, u'v'), (uw, u'w') \in T | (v, v'), (w, w') \in T_t(u), \mathrm{lcp}(\{v', w'\}) = \lambda$

This representative sample ensures that all the states of the kernel that represent different states of the (partial) subsequential function can be distinguished.

The original algorithm can be modified in such a way that the rejection of the negative sample is checked every time that a successful block of merges is performed (line 19 in fig. 1). This new **if** condition will be "$\neg subseq(\tau)$ **or** $S_- \cap \mathrm{Dom}(\tau) \neq \emptyset$", so the new transducer is checked before accepting it. The new transducer adds a new input parameter: a finite set of input strings, S_-, not belonging to the domain.

Example 3: Let t the partial subsequential function defined in example 2. Let $T = \{(a, \lambda), (acc, 00), (acccc, 0000), (bc, 1), (c, 2)\}$ a sample drawn from this function and $S_- = \{ac\}$ a negative sample of the domain.

The algorithm begins building the OTST of the training sample (fig. 4.a). Next, the algorithm, attempts to merge states λ and a (fig. 4.b). This transducer has two edges $(\lambda, c, 2, c)$ and $(\lambda, c, 00, ac)$ which violate the *subseq* condition. Given that $c \neq \#$ and $c > a$ then c is pushed back through the two edges and the states c and ac are merged (fig. 4.c). The rejection of the negative sample must be checked now. As $\tau(ac) = 2$ then $S_- \cap \mathrm{Dom}(\tau) \neq \emptyset$ and the algorithm returns to the fig. 4.a transducer. Next, it tries to merge states λ and b, but then the edges (λ, c, s, c) and $(\lambda, c, \lambda, bc)$ violate the *subseq* condition. Now, the **if** condition does not fulfill, the inner most loop is exited and the algorithm goes back to the transducer of fig. 4.a. After merging the states a and b, (fig. 4.d) and all the subsequent *push_back* and merge operations, the subsequential transducer of fig. 4.e is obtained. As $ac \in \mathrm{Dom}(\tau)$ it must be rejected and the fig. 4.a transducer is recovered. The next two merges (λ with c and λ with $a\#$) can be performed without problems and the fig. 4.f transducer is obtained. Following the algorithm the next successful merges are: λ with $c\#$ (fig. 4.g), a with acc (fig. 4.h), λ with bc (fig. 4.i) and b with bcc (fig. 4.j) which yields the transducer inferred from the training sample.

6 Identification using domain information

In real transductions tasks it is unusual to have negative samples. Instead, sometimes a description of the domain is available or can be inferred from the input strings of the transduction sample.

First, we will show now that if the language that describes the domain is known, then all the different states of the OST that realizes any subsequential function can be distinguished and hence the identification in the limit can be proved. As the domain language of a subsequential function is regular, we can assume that the minimum DFA that describes this language is available (D). Let u, v be two non-distinguishable states from the transduction. If $T_t(u) \neq T_t(v)$ such that $\exists w \in \mathrm{Dom}_t(u) \oplus \mathrm{Dom}_t(v)$, then, as D is the minimum DFA describing the domain language, $T_D(u) \neq T_D(v)$ and $\delta_D(q_0, u) \neq \delta_D(q_0, v)$.

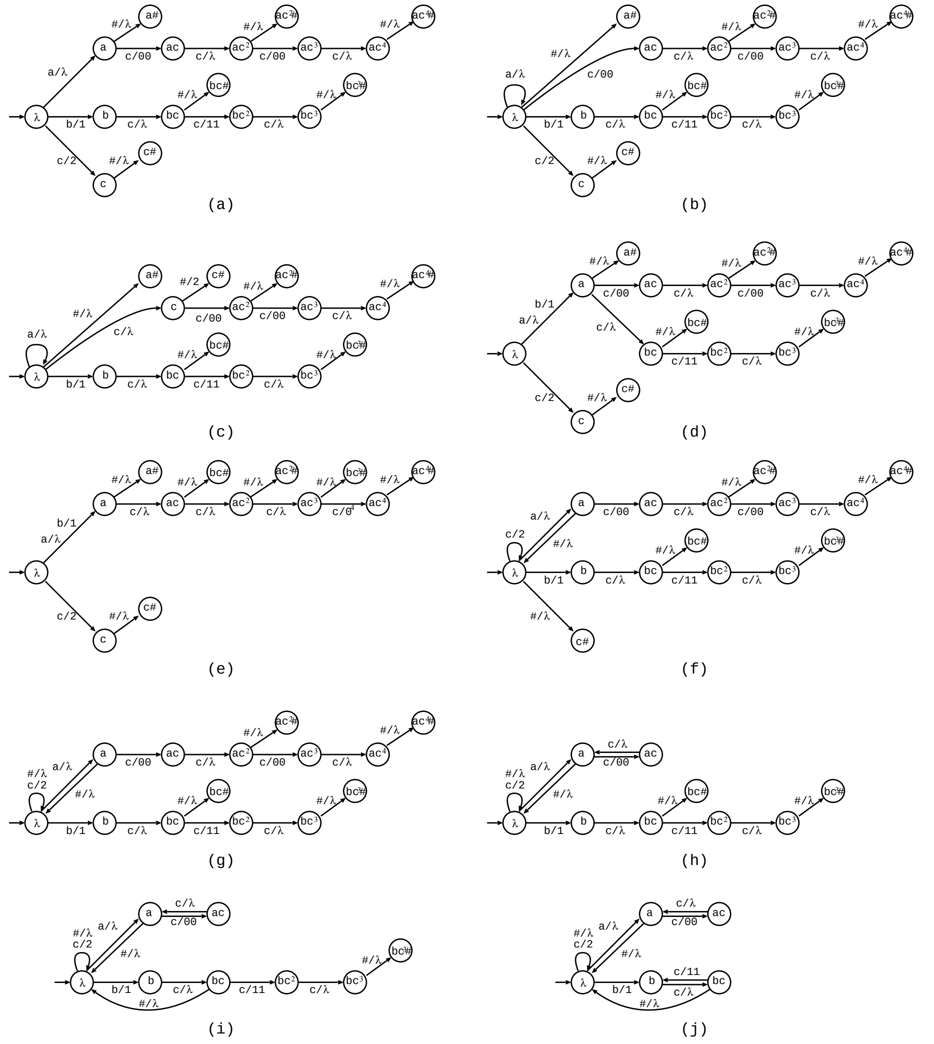

Fig. 4. *Some key steps of the OSTIA-N algorithm as applied to the training set* $T = \{(a, \lambda), (acc, 00), (acccc, 0000), (bc, 1), (bccc, 111), (c, 2)\}$ *and with* $S_{-} = \{ac\}$

Therefore, if the merge of two states u, v such that $\delta_D(q_0, u) \neq \delta_D(q_0, v)$ is forbidden in the algorithm, the merge of states that are not distinguishable by the transduction can be avoided and then the identification can be reached. The new OSTI algorithm adds a new input parameter (the DFA modeling the domain language, D) and a condition before trying to merge a pair of states (from line 9 to line 19 in fig. 1): "**if** $\delta_D(q_0, \text{input\_prefix}(p)) \neq \delta_D(q_0, \text{input\_prefix}(q))$ **then**".

In this algorithm the function $\text{input\_prefix}(q) : Q \to X^*$ is introduced, this functions returns the shortest string u such that $(q_0, u, u', q) \in \prod_{\tau}$. The result of $\delta_D(q_0, \text{input\_prefix}(p))$ can be calculated with no cost, labeling each state q of the OTST with the state of D that is reached with the unique string u such that $(q_0, u, u', q) \in \prod_{\text{OTST}}$. These labels do not change during execution because only states with the same label are merged. Because the edges having # as input symbol are not true edges but a representation of the function σ of the subsequential transducer, the labels of states accessed by such edges are not relevant and we have arbitrarily labeled them with the initial state of D.

On the other hand, in spite of it is not possible to guarantee anything about the identification in the limit when the language that describes the domain is unknown, an inferred description of it [Mic,80] [Ang,82] [GV,90] can be used as a way to introduce *heuristic information* in the inference process. As shown by some previous experimental results [Varó,94] [OCVJ,94] [CGV,94] this additional information about the domain accelerates the learning algorithm and reduces the error rate of the transducers inferred.

This technique ensures that the domain of the transducer obtained is always included in the language accepted by D, even if D is not minimum or does not exactly describe the domain of the target transducer.

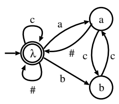

Fig. 5. *Automaton for the domain language of the partial function of example 2.*

Example 4: Let t be the partial function defined in example 2. Let D be the automaton describing the domain language (fig. 5) and let $T = \{(a, \lambda), (acc, 00), (acccc, 0000), (bc, 1), (bccc, 111), (c, 2)\}$ be a sample drawn from this function.

As in the previous case, the algorithm begins building the OTST of the training sample, but in this case, the states are labeled with the corresponding state of D (fig 6.a). Then, each state $q \in Q_{\text{OTST}} - \{\lambda\}$ has a label $label(q) = \delta_D(label(p), a)$ where $(p, a, x, q) \in E_{\text{OTST}}$ and $label(\lambda) = \lambda$ (fig 6.a).

Then, it tries to merge states λ and a but it cannot be done because they have different labels. Following lexicographic order, the next pair of states with identical labels is λ and c. They can be merged and the transducer of fig. 6.b is obtained. After merging states λ and $a\#$ the fig. 6.c transducer is obtained. Next, the algorithm, tries to merge the states b and ac (fig. 6.d). As the resulting transducer is not subsequential the inner loop will try to transform it in a subsequential one by pushing back symbols and merging states. At the end the transducer in fig. 6.e is obtained. This transducer does not fulfills the **if** condition in the inner most loop. Therefore, the transducer is rejected and the transducer in fig. 6.c is recovered.

Following the algorithm, the next successful merges are: λ with $c\#$ (fig. 6.f), a with abb (fig. 6.g), λ with bc (fig. 6.h) and b with bcc (fig. 6.i) which yields the transducer inferred from the training sample.

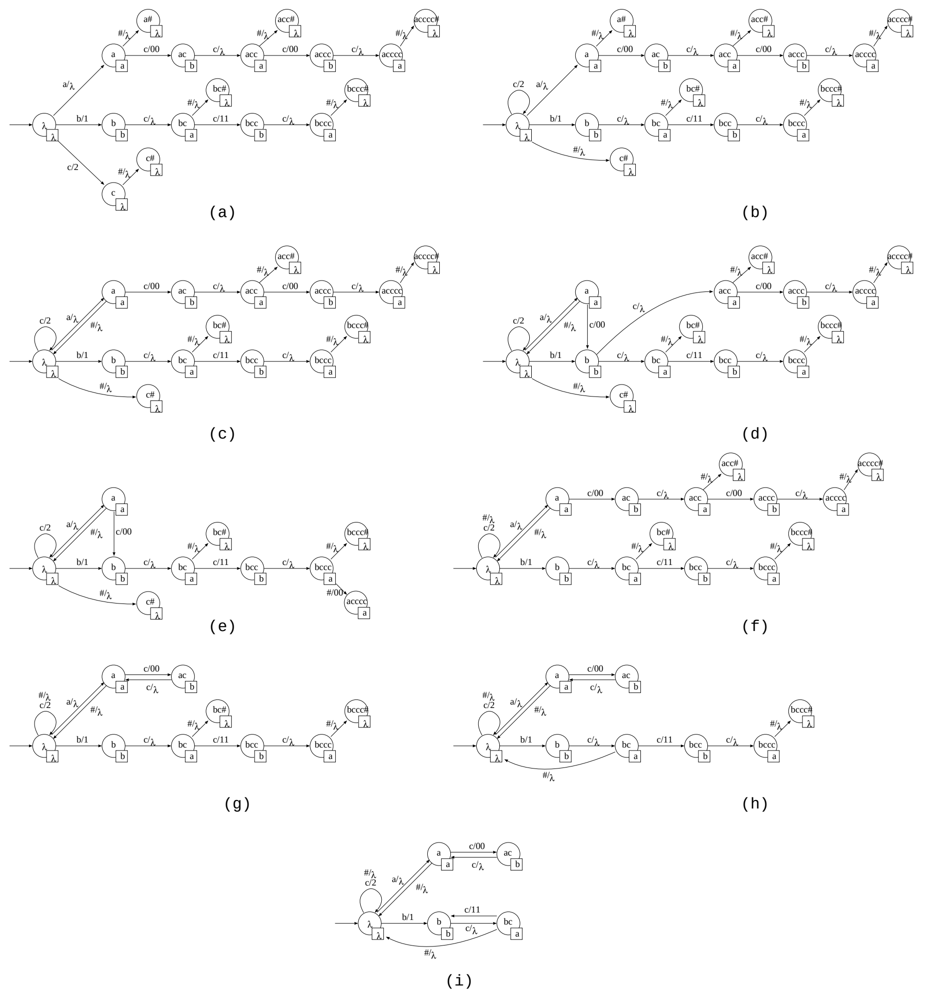

Fig. 6. *Some key step of the OSTIA-D as applied to the training set* $T = \{(a, \lambda), (acc, 00), (acccc, 0000), (bc, 1), (bccc, 111), (c, 2)\}$

7 Discussions and conclusions

In this work the OSTI algorithm has been extended to allow the identifiability in the limit of any partial subsequential function by using input–output pairs plus some additional information in the form of either negative examples of the domain or a description of the domain by means of a DFA. The techniques presented here are not mutually exclusive and can be used together if necessary.

In addition to the theoretical interest, these methods can be used to introduce heuristic information and help the basic OSTI algorithm even when not enough input–output pairs are available.

References

[Ang,82] D. Angluin, "Inference of Reversible Languages". *J. ACM* 29, 741–765 (1982).

[AS,83] D. Angluin, C. H. Smith. "Inductive inference: theory and methods". *Computing Surveys*, **15**, 237–269 (1983).

[Ber,79] J. Berstel. *Transductions and Context-Free Languages.* Teubner, Stuttgart (1979).

[CGV,94] A. Castellanos, I. Galiano, E. Vidal. "Application of OSTIA to Machine translation Tasks". *Grammatical Inference and Applications.* 2nd International Colloquium on Grammatical Inference. R.C. Carrasco and J. Oncina eds. El Campello (Spain) (1994).

[Gold,67] E. M. Gold. "Language identification in the limit". *Information and Control*, **10**, 447–474 (1967).

[Gold,78] E. M. Gold. "Complexity of automaton identification from given data". *Information and Control*, **37**, 302–320 (1978).

[GOV,90] P. García, E. Vidal, J. Oncina. "Learning Locally Testable Languages in the Strict Sense", *Proceedings of the First International Workshop on Algorithmic Learning Theory*, Japanese Society for Artificial Intelligence, 325–337 (October 1990).

[GV,90] P. García, E. Vidal. "Inference of k-Testable Languages in the Strict Sense and application to Syntactic Pattern Recognition". *IEEE Trans. Pattern Analysis and Machine Intelligence*, **12**, 920–925 (1990).

[LRC,84] P. Luneau, M. Richetin, C. Cayla. "Sequential learning from input–output behavior", *Robotica*, **1**, 151–159 (1984).

[Mic,80] L. Miclet. "Regular Inference with a Tail-Clustering Method", *IEEE Trans.*, SMC-**10**, 737–743 (1980).

[Onc,91] J. Oncina. "Aprendizaje de lenguajes regulares y funciones subsecuenciales". Ph. D. dissertation, Universidad Politécnica de Valencia (1991).

[OG,91] J. Oncina, P. García. "Inductive learning of subsequential functions". Univ. Politécnica de Valencia, Tech. Rep. DSIC II 31 (1991).

[OGV,93] J. Oncina, P. García, E. Vidal. "Learning Subsequential Transducers for Pattern Recognition Interpretation Tasks". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15, 448–458 (1993).

[OCVJ,94] J. Oncina, A. Castellanos, E. Vidal, V. M. Jiménez. "Corpus based machine translation through subsequential transducers". Proceedings of the *3rd. International Conference on the Cognitive Science of Natural Language Processing*, Dublin (July 1994).

[Tak,88] Y. Takada, "Grammatical inference for even linear languages based on control sets", *Inform. Processing Lett.*, **28**, no.4, 193–199 (1988).

[Varó,94] M. A. Varó. "Inferencia de Transductores Subsecuenciales con Información de Dominio y Rango" (in Spanish). Final year project supervised by J. Oncina. Facultad de Informática, Universidad Politécnica de Valencia (1994).

[Vel,78] L. P. J. Velenturf. "Inference of sequential machines from sample computation", *IEEE Trans. Comput.*, **27**, 167–170 (1978).

[VGS,90] E. Vidal, P. García, E. Segarra, "Inductive learning of finte-state transducers for the interpretation of unidimensional objects", *Structural Pattern Analysis.* R. Mohr, T.Pavlidis and A. San Feliu eds. New York: World Scientific, 17–35 (1990).

This article was processed using the LaTeX macro package with LLNCS style