2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Unsupervised Inductive Learning In Symbolic Sequences via Recursive Identification of Self-Similar Semantics

Ishanu Chattopadhyay, Yicheng Wen, Asok Ray, Shashi Phoha
The Pennsylvania State University, USA

***Abstract*—This paper presents a new pattern discovery algorithm for constructing probabilistic finite state automata (PFSA) from symbolic sequences. The new algorithm, described as Compression via Recursive Identification of Self-Similar Semantics (CRISSiS), makes use of synchronizing strings for PFSA to localize particular states and then recursively identifies the rest of the states by computing the n-step derived frequencies. We compare our algorithm to other existing algorithms, such as D-Markov and Casual-State Splitting Reconstruction (CSSR) and show both theoretically and experimentally that our algorithm captures a larger class of models.**

I. Introduction & Motivation

The principal focus of this work is the development of an efficient algorithm for discovering patterns in symbolic data streams, within the framework of Probabilistic Finite State Automata (PFSA) over pre-defined symbolic alphabets.

A finite state automaton (FSA) is essentially a finite graph where the nodes are known as states and the edges are known as transitions, which are labeled with letters from an alphabet. A string or a symbol string generated by a FSA is a sequence of symbols belonging to an alphabet, which are generated by stepping through a series of transitions in the graph. Probabilistic finite state automata, considered in this paper, are finite state machines with probabilities associated with the transitions. The PFSA formalism has been extensively studied as an efficient framework for learning the causal structure of observed dynamical behavior [1]. This is an example of inductive inference [2], defined as the *process of hypothesizing a general rule from examples*. In this paper, we are concerned with the special case, where the *inferred general rule* takes the form of a PFSA, and the examples are drawn from a stochastic regular language. Conceptually, in such scenarios, one is trying to learn the *structure inside of some black box, which is continuously emitting symbols* [1]. The system of interest may emit a continuous valued signal; which must be then adequately partitioned to yield a symbolic stream. Note that such partitioning is merely *quantization* and not *data-labeling*, and several approaches for efficient symbolization have been reported [3].

Probabilistic automata are more general compared to their non-probabilistic counterparts [4], and are more suited to modeling stochastic dynamics. It is important to distinguish between the PFSA models considered in this paper, and the ones considered by Paz [5], and in the detailed recent review by Vidal *et al.* [6]. In the latter framework, symbol generation probabilities are not specified, and we have a distribution over the possible end states, for a given initial state and an observed symbol. In the models considered in this paper, symbol generation is probabilistic, but the end state for a given initial state, and a generated symbol is unique. Unfortunately, authors have referred to both these formalisms as *probabilistic finite state automata* in the literature. The work presented here specifically considers the latter modeling paradigm considered and formalized in [1], [7], [8], [9].

The case for using PFSA as pattern classification tools is compelling. Finite automata are simple, and the sample and time complexity required for learning them can be easily characterized. This yields significant computational advantage in time constrained applications, over more expressive frameworks such as belief (Bayesian) networks [10], [11] or Probabilistic Context Free Grammars (PCFG) [12], [13] (also see [14] for a general approach to identifying PCFGs from observations) and hidden Markov models (HMMs) [15]. Also, from a computational viewpoint, it is possible to come up with provably efficient algorithms to optimally learn PFSA, whereas "optimally learning HMMs is often hard" [1]. Furthermore, most reported work on HMMs [15], [16], [17] assumes the model structure or topology is specified in advance, and the learning procedure is merely training, $i.e.$, finding the right transition probabilities. For PFSA based analysis, researchers have investigated the more general problem of learning the model topology, as well as the transition probabilities, which implies that such analysis can then be applied to domains where there is no prior knowledge as to what the correct structure might look like [18].

The basic algorithmic scheme for constructing a PFSA based description of hidden dynamics from physical observations proceeds as follows: A symbol string is obtained from the output of the dynamical system of interest, either directly or through proper symbolization of the time-series data [3]. Thereafter, an irreducible PFSA is constructed from the symbol sequence via some proper construction algorithms reported in the literature. The key of such a construction algorithm is to find the correct notion of states from the symbol string. It has been shown in [9] that a state in a PFSA is nothing but an equivalence class of strings over the alphabet under the Nerode equivalence relation.

Among these PFSA construction algorithms, D-Markov [19] algorithm can capture all D-Markov machines, which is essentially Markov chains of any finite order, with a correct choice of depth d. The Causal-State Splitting Reconstruction (CSSR) [8] starts with assuming that the process is an independent, identically-distributed (i.i.d.) sequence, with one causal state and then split it to a probabilistic suffix tree of depth L_{max}. Each node on the tree defines a state labeled with a suffix and any two nodes (or states) are merged if their next-symbol generating probabilities are the same by use of the χ^2 (or Kolmogorov-Smirnov) hypothesis testing, no matter where the nodes are located on the tree. Finally, the transitions are

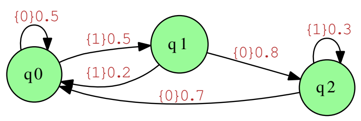

Fig. 1. An example of Synchronizable Machines

checked to maintain deterministic. CSSR is shown to be able to identify the even process, which is not a Markov chain of any finite order. Therefore, it is claimed that the class of PFSA that CSSR can capture strictly includes those captured by D-Markov algorithm.

In this paper, we define a class of PFSA, called synchronizable machines, which contains all D-Markov machines. We show that although the even process is in the class of synchronizable machines, CSSR cannot in general capture the structure of a synchronizable machine efficiently. However, the proposed CRISSiS algorithm, that we develop in this paper, can identify all synchronizable machines.

The rest of the paper is organized as follows. In section II we give the preliminary of the formal language theory. In section III we present in detail the three steps of the CRISSiS algorithm. In section IV, an example is used to illustrate the CRISSiS algorithm. In section V, we discuss the effect of the parameters on the algorithm and we compare CRISSiS with CSSR. The paper is concluded in section VI, with recommendations for future work.

II. PRELIMINARY

In the formal language theory [20], an alphabet Σ is a (nonempty finite) set of symbols. A string x over Σ is a finite-length sequence of symbols in Σ. The length of a string x, denoted by $|x|$, represents the number of symbols in x. The Kleene closure of Σ, denoted by $\Sigma^\star$, is the set of all finite-length strings of events including the null string ϵ. All strings of length $d \in \mathbb{N}$ is denoted by $\Sigma^d \subset \Sigma^\star$. The string xy is called the concatenation of x and y.

Definition 1 (PFSA): A probabilistic finite state automaton is a tuple $G = (Q, \Sigma, \delta, q^0, \tilde{\pi})$, where

- Q is a (nonempty) finite set, called set of states;
- Σ is a (nonempty) finite set, called input alphabet;
- $\delta : Q \times \Sigma \to Q$ is the state transition function;
- $q^0 \in Q$ is the start state;
- $\tilde{\pi} : Q \times \Sigma \to [0, 1]$ is an output mapping which is known as the probability morph function and satisfies the condition $\sum_{\tau\in\Sigma} \tilde{\pi}(q_j, \tau) = 1$ for all $q_j \in Q$.

The transition map δ naturally induces an extended transition function $\delta^\star : Q \times \Sigma^\star \to Q$ such that $\delta^\star(q, \epsilon) = q$ and $\delta^\star(q, \omega\tau) = \delta(\delta^\star(q, \omega), \tau)$ for $q \in Q$, $\omega \in \Sigma^\star$ and $\tau \in \Sigma$.

Definition 2 (Synchronizing String): Given a PFSA $G = (Q, \Sigma, \delta, q^0, \tilde{\pi})$, a string $\omega \in \Sigma^\star$ is called a synchronizing string for a particular state $q \in Q$ if $\delta^\star(q_i, \omega) = q$ for all $q_i \in Q$.
Synchronizing strings allow us to localize the states in a PFSA even if the initial condition of the process is unknown. A PFSA is called synchronizable if there exists a synchronizing string.

Definition 3 (Irreducibility): A PFSA $G = (Q, \Sigma, \delta, q^0, \tilde{\pi})$ is called irreducible if for any $q_i, q_j \in Q$, there exits a string $\omega_{ij} \in \Sigma^\star$ such that $\delta(q_i, \omega_{ij}) = q_j$.

Given a PFSA $G = (Q, \Sigma, \delta, q^0, \tilde{\pi})$, it is easy to generate a random symbol sequence $\mathbb{X} = \{X_k\}_{k=0}^{\infty}$, where $X_k \in \Sigma$. Let $q^k \in Q$ denote the state of $\mathbb{X}$ at time k after symbol the X_{k-1} is generated. A future symbol X_k is generated based on the next symbol distribution at state q^k, namely, $\tilde{\pi}(q^k, \cdot)$ and the state transition follows $q^{k+1} = \delta(q^k, X_k)$.

III. The CRISSiS algorithm

Consider a stationary symbol sequence $\mathbb{S}$ of length N, which is generated by a synchronizable and irreducible PFSA G. We wish to find out G by looking at the sequence $\mathbb{S}$. We have the following assumptions on G.

- G is irreducible;
- G is a minimal realization;
- G is a synchronizable machine.

The CRISSiS algorithm consists of three steps.

A. Identification of a Shortest Synchronizing String

Proposition 1: ω is a synchronizing string for a state $q \in Q$ if and only if q contains the language $\{\Sigma^\star\omega\}$.
In other words, if ω is a synchronizing string, then $\forall \nu \in \Sigma^\star$ and $\forall u \in \Sigma^\star$,

$$\Pr(u|\omega) = \Pr(u|\nu\omega) \tag{1}$$

Theoretically speaking, Equation 1 can be used to check if a particular string ω is a synchronizing string. However, in reality, we need to resolve the following two issues. The first one is how to obtain the estimation of the conditional probabilities from the symbolic sequence $\mathbb{S}$. The second issue is that it is impossible to check Equation 1 for all $\nu \in \Sigma^\star$ and $u \in \Sigma^\star$.

Definition 4 (Count Function): Given a symbolic sequence $\mathbb{S}$ over the alphabet Σ, the count function, $\# : \Sigma^\star \to \mathbb{N} \cup \{0\}$, of a sub-string $\omega \in \Sigma^\star$, is defined as the integer count of the number of possible overlapping occurrences of ω in $\mathbb{S}$.

Simply put, to find out $\#\omega$, we fix a window of size $|\omega|$, slide the window over $\mathbb{S}$ by one symbol at a time, and count how many times ω occurs.

Definition 5 (d-th Order Derived Frequency): For a symbolic sequence $\mathbb{S}$, the d-th derived frequency of a string $\omega \in \Sigma^\star$, denoted by $\vartheta_d(\omega)$, is the probability mass function over the space Σ^d such that $\forall \sigma_1\sigma_2 \ldots \sigma_d \in \Sigma^d$,

$$\vartheta_d(\omega) = \frac{\#(\omega\sigma_1\sigma_2 \ldots \sigma_d)}{\sum_{\tau_1\tau_2\ldots\tau_d\in\Sigma^d} \#(\omega\tau_1\tau_2 \ldots \tau_d)} \tag{2}$$

Clearly,$\forall \sigma \in \Sigma$ and $\forall \omega \in \Sigma^\star$,

$$\hat{\Pr}(\sigma|\omega) = \frac{\#(\omega\sigma)}{\sum_{\tau\in\Sigma} \#(\omega\tau)} = \vartheta_1(\omega)\big|_\sigma \tag{3}$$

We have introduced the derived frequencies as estimates of the conditional probabilities. For the second issue, we check Equation 1 up to only L_1-step history and L_2-step future rather than all $\Sigma^\star$. In other words, for a symbolic sequence $\mathbb{S}$ and a sub-string ω, the null hypothesis for w being a synchronizing string is

$$\vartheta_d(\omega) = \vartheta_d(u\omega), \forall u \in \bigcup_{i=1}^{L_1} \Sigma^i, \forall d = 1, 2, \ldots, L_2 \quad (4)$$

where L_1 and L_2 are two parameters for the algorithm. In D-Markov or CSSR algorithm, both L_1 and L_2 are taken to be one.

We use χ^2 test to check this null hypothesis (Equation 4) for each u and d. If one of these tests is rejected, then ω is not a synchronizing string. Otherwise, ω is considered as a synchronizing string. Algorithm 2 described the procedure of checking if ω is a synchronizing string.

The procedure of step 1 is to search for a shortest synchronizing string ω_{syn} by checking all the L_1-step histories and L_2-step futures. We start with all strings of length one and see whether a particular string ω is a synchronizing string according to statistical test on Equation 4. If ω passes the statistical test, then we find a synchronizing string ω and then go to step 2. Otherwise, we increase the length of the strings and do this check for these longer strings again and this routine repeats until we find a synchronizing string. The pseudo-code of step 1 is shown in Algorithm 1.

B. Recursive Identification of States

States are equivalence class of strings under Nerode equivalence class. For any two strings ω_1 and ω_2 in a state q,

$$\Pr(\omega|\omega_1) = \Pr(\omega|\omega_2). \quad (5)$$

Each state q is uniquely identified with these future conditional probabilities. Therefore, Equation 5 can be applied to check if two states q_1 and q_2 are the same states provided that $\omega_1 \in q_1$ and $\omega_2 \in q_2$. Note that we have the similar issues in Equation 5 as the ones in Equation 1. Similarly we can use the derived frequencies to estimate the conditional probabilities and also introduce a parameter L_2. Then Equation 5 becomes

$$\vartheta_d(\omega_1) = \vartheta_d(\omega_2), \forall d = 1, 2, \ldots, L_2 \quad (6)$$

We only check the L_2-step future of the conditional probabilities to distinguish two states.

To be able to use Equation 6 for discovering the states, ω_1 and ω_2 must be synchronizing strings. For D-Markov machines with certain depth d, every sufficiently long string (of length no less than d) is a synchronizing string for a particular state. In other words, after observing the long string, we know which state the PFSA is. This is generally not true for synchronizable machines because there exists some infinite-length strings which are not synchronizing strings because the initial condition of the process is unknown to us. For example, for the PFSA in Figure 1, the infinite string of all one's is not a synchronizing string.

The follow proposition says that any string with a synchronizing string as its prefix is also a synchronizing string.

Proposition 2: If ω is a synchronizing string for $q_i \in Q$, then $\omega\tau$ is also a synchronizing string for $q_j = \delta(q_i, \tau)$ where $\tau \in \Sigma$.

The procedure of step 2 (Algorithm 1) is described as follows. Let Q denote the set of states to be discovered for PFSA. Initialize Q to contain only one state defined by the synchronizing string ω_{syn} found in step 1. The construct a tree with ω_{syn} as the root node and split the root node to $|\Sigma|$ children. The child nodes are regarded as candidate states with representation $\omega_{syn}\sigma$ for $\sigma \in \Sigma$. Start with any child node and use Equation 6 to check if this node matches with any states in Q (Algorithm 3). If it does, then remove this child node and the original transition from its parent to itself with symbol σ and add a new transition from its parent node to the matched state with the same symbol σ. Do not further split this branch any more. If the child node does not match with any states in Q, then it is considered as a new state. The set of states Q is updated with this new state added. Furthermore, the new state split into its $|\Sigma|$ children (candidate states) by adding one extra symbol to the string that represents the new state. Finally, go to another untouched candidate states and repeat this procedure until there is no more candidate states to visit.

Step 2 must terminate since G has only finite number of states. The node of the tree corresponds to the states of the PFSA and the edges of the tree are exactly the state transition function δ of the PFSA.

C. Estimation of Morph Probabilities

We obtained the set of states and the state transition map δ from step 2. We need to compute the probability morph function δ. The idea is to feed the symbolic sequence $\mathbb{S}$ through the structure δ. Let N_i denote how many times $\mathbb{S}$ visits a state $q_i \in Q$ and N_{ij} denotes how many times a symbol $\sigma_j \in \Sigma$ is generated from a state q_i in $\mathbb{S}$. Note that the distribution of the outgoing symbols at a particular state follows a multinominal distribution of dimension $|\Sigma| - 1$. We can simply use its maximum likelihood estimator (MLE) to compute $\tilde{\pi}$, i.e.

$$\hat{\tilde{\pi}}(q_i, \sigma_j) = \frac{N_{ij}}{N_i} \quad (7)$$

However, we do not know which state to start from. Again we can make use of the synchronizing string we found in the first step. We first search for the first occurrence of ω_{syn} in the symbol sequence $\mathbb{S}$ and localize the state. Then we can count all the N'_{ij}s and N'_is afterwards. This simple routine is described in Algorithm 1.

IV. An Example

To illustrate how the CRISSiS algorithm works, let us consider an simple example, the three-state synchronizable machine over the binary alphabet in Figure 1. Let us denote this machine by H. We simulate a binary symbolic sequence of length ten thousand from H. Table I gives the frequency counting of some strings occurred in the sample. For simplicity of the argument, Let us select both L_1 and L_2 to be one, which means we only check one-step both in history and in future for the derived frequencies.

The first step is to find a synchronizing string by checking Equation 4. We start with checking shortest strings and then to longer strings. In this example, we check 0, 1, 00 and so on. Table II lists the derived frequencies of the strings 0, 1, 00 and their 1-step history. For 0, we find neither $\vartheta_1(0) = \vartheta_1(00)$ nor $\vartheta_1(0) = \vartheta_1(10)$ can pass the χ^2 test. We move on to the symbol 1 and it fails the test again. We then check the string 00 and since the derived frequencies in the third column are

Algorithm 1 Main Algorithm

```
Inputs: Symbolic string $\mathbb{X}$, $\Sigma$, $L_1$, $L_2$, significance level $\alpha$;
Outputs: PFSA $\hat{G} = \{Q, \Sigma, \delta, \tilde{\pi}\}$;
/** Step 1: find the synchronizing string $\omega_{syn}$ **/
$\omega_{syn} \leftarrow$ null;
$d \leftarrow 0$;
while $\omega_{syn}$ is null do
    $\Omega \leftarrow \Sigma^d$;
    for all $\omega \in \Omega$ do
        if (isSynString($\omega$,$L_1$)) then
            $\omega_{syn} \leftarrow \omega$;
            break;
        end if
    end for
    $d \leftarrow d + 1$;
end while
/** Step 2: find the states Q and the structure $\delta$ **/
$Q \leftarrow \{\omega_{syn}\}$;
$\widetilde{Q} \leftarrow \{\}$;
Add $\omega_{syn}\sigma_i$ to $\widetilde{Q}$ and $\delta(\omega_{syn}, \sigma_i) = \omega_{syn}\sigma_i$ for all $\sigma \in \Sigma$;
for all $\omega \in \widetilde{Q}$ do
    if $\omega$ occurs in $\mathbb{X}$ then
        $\omega^\star \leftarrow$ matchStates($\omega, Q, L_2$)
        if $\omega^\star$ is null then
            Add $\omega$ to Q;
            Add $\omega\sigma_i$ to $\widetilde{Q}$ and $\delta(\omega, \sigma_i) = \omega\sigma_i$ for all $\sigma \in \Sigma$;
        else
            Replace all $\omega$ by $\omega^\star$ in $\delta$;
        end if
    end if
end for
/** Step 3: find the morph function $\tilde{\pi}$ **/
Find $k$ such that $X_k$ is the symbol after the 1st occurrence of $\omega_{syn}$ in $\mathbb{X}$;
Initialize $\tilde{\pi}$ to zero;
$state \leftarrow \omega_{syn}$;
for all $i \geqslant k$ in $\mathbb{X}$ do
    $\tilde{\pi}(state, X_i) \leftarrow \tilde{\pi}(state, X_i) + 1$;
    $state \leftarrow \delta(state, X_i)$
end for
Normalize $\tilde{\pi}$ for each state;
```

very close to each other, it passes the χ^2 test and 00 is the synchronizing string. This completes the first step.

In the second step, we start with the synchronizing string 00, which defines a state, and 00 is split into two candidate states 000 and 001, where the single circles denote the candidates states and the double circles denote the identified states. We compare the derived frequencies of each candidate states to all the identified states according to Equation 6. Since $\vartheta_1(000) = [0.494\ 0.506]$ is quite close to $\vartheta_1(00) = [0.500\ 0.500]$, we consider 000 and 00 are the same states. We remove the state 000 and its transition from 00 and put a self-loop on the state 00. Now we look at another candidate state 001 and find that $\vartheta_1(001) = [0.800\ 0.200]$ is not close to $\vartheta_1(00)$, so 001 is identified as a new state and is further split into two candidate states 0010 and 0011. We repeat this same procedure for states 0010 and 0011. $\vartheta_1(0010) = [0.703\ 0.297]$ does not match with either 00 or 001, and therefore is identified as a new state and is split to 00100 and 00101. $\vartheta_1(0011) = [0.500\ 0.500]$ is close to $\vartheta_1(00)$, and hence 0011 is considered the same as 00. The state 0011 is removed and a new transition is added from 001 to 00 with symbol 1. Finally, after computing the derived frequencies for 00100 and 00101 ($\vartheta_1(00100) = [0.505\ 0.495]$ and $\vartheta_1(00101) = [0.7000.300]$), they are identified to the existing states 00 and 0010, respectively. The graph does not contain any more candidate states and step 2 is completed. Note that this graph gives us the states and the structures of the PFSA, which is the same as the model H we started with.

Algorithm 2 isSynString(ω,L_1)

```
Outputs: true or false;
for $D = 0$ to $L_1$ do
    for all $s \in \Sigma^D$ do
        if $\vartheta_d(\omega) = \vartheta_d(s\omega)$ fails the $\chi^2$ test for some $d \leqslant L_2$ then
            return false;
        end if
    end for
end for
return true
```

Algorithm 3 matchStates(ω, Q, L_2)

```
for all $i \in Q$ do
    if $\vartheta_d(\omega) = \vartheta_d(Q(i))$ (Eq. 6) passes the $\chi^2$ test for all d then
        return $Q(i)$, the $i$-th element of Q;
    end if
end for
return null;
```

In the last step, feeding the sampled string through the graph allows us to compute the MLE of the morph probabilities by Equation 7.

V. Performance of CRISSiS

A. Time Complexity

Suppose the length of the input symbolic sequence $\mathbb{S}$ over alphabet Σ is N and the underlying unknown PFSA G, which generates $\mathbb{S}$, has $|Q|$ number of states.

TABLE I
COUNT STATISTICS OF THE STRINGS FROM TEN THOUSAND SAMPLE POINTS FROM THE PFSA IN FIGURE 1

ω	$\#\omega$	ω	$\#\omega$	ω	$\#\omega$
0	62711	0000	8673	00100	9881
1	37291	0001	8892	00101	4181
00	35164	0010	14062	001000	4990
01	27546	0011	3536	001001	4891
10	27545	0100	14206	001010	2926
11	9745	0101	7245	001011	1255
000	17565	1000	8892		
001	17599	1001	8707		
010	21451	1100	3393		
011	6094	1101	2701		
100	17599				
101	9946				
110	6094				
111	3651				

TABLE II
DERIVED FREQUENCIES OF SOME STRINGS FROM THE SAMPLE OF H

ω	$\vartheta_1(\omega)$	ω	$\vartheta_1(\omega)$	ω	$\vartheta_1(\omega)$
0	[0.561 0.439]	1	[0.739 0.261]	00	[0.500 0.500]
00	[0.500 0.500]	01	[0.779 0.221]	000	[0.494 0.506]
10	[0.639 0.361]	11	[0.625 0.375]	100	[0.505 0.495]

TABLE III
DERIVED FREQUENCIES OF THE SIMULATED SEQUENCE FROM THE 3D-MARKOV MODEL

ω	$\vartheta_1(\omega)$	ω	$\vartheta_1(\omega)$
0	[0.538 0.462]	000	[0.914 0.086]
00	[0.538 0.462]	010	[1 0]
10	[0.539 0.463]	100	[0.100 0.900]
		110	[0.400 0.600]

In step 1, we search for a shortest synchronizing string. It is reported in [21] that the length of a shortest synchronizing string is on the order of $|Q|^3$. Therefore, the number of strings we need to check is $|\Sigma|^{O(|Q|^3)}$. To test each string, we need to go to L_1-step history and L_2-step history to compute the derived frequencies from $\mathbb{S}$, which gives $O(|\Sigma|^{L_1+L_2} \cdot N)$. The running time for step 1 is $|\Sigma|^{O(|Q|^3)+L_1+L_2} \cdot O(N)$. In step 2, the number of candidate states we check is $O(|Q|)$. For each candidate state, we need to check its L_2-step future with all the existing states, which gives $O(|Q||\Sigma|^{L_2}N)$. In the final step, searching for the synchronizing string and compute the estimates both take $O(N)$. Hence, the time complexity of the CRISSiS algorithm is $O(N) \cdot \left(|\Sigma|^{O(|Q|^3)+L_1+L_2} + |Q||\Sigma|^{L_2}\right)$. It is still linear in the input N. We also note that it is exponential in the length of the synchronizing string (or $|Q|$), L_1 and L_2. Both L_1 and L_2 are usually quite small. Then main computational burden lies in the first step, especially when the length of the synchronizing string is very large.

B. Choice of the Parameters

In step 1, we introduced a parameter L_1 in Equation 4. A small L_1 may render a wrong identification of a synchronizing string. We simulated a symbolic sequence from a 3D-Markov machine, where any string of length 3 is a shortest synchronizing string. Table III listed the derived frequencies of the simulated sequence. Consider the string 0, it turns out that $\vartheta(0)$, $\vartheta(00)$, and $\vartheta(10)$ are very close. If $L_1 = 1$, the algorithm would think 0 as a synchronizing string and terminate the step 1. The output model is not irreducible and contains three transient states q_0,q_1, and q_2. Although in this particular example, it seems that we can still recover the model by removing the transient states, the mis-identification of the synchronizing string will cause a wrong identification of the structure if some states are accidentally merged with those transient states in step 2. We can avoid this mistake by increasing L_2 to 2. Table III clearly shows 0 cannot pass the test in Equation 4 with $L_1 = 2$.

In step 2, we introduced another parameter L_2 to simplify Equation 5 to Equation 6 where we only distinguish states up to their L_2-step future. This may lead to a wrong identification of the states and structure in PFSA if there are two states that have exactly the same L_2-step future but different after that. Note that in this model, the 1-step derived frequencies of the state 00 and 11 are the same. We run CRISSiS algorithm on this sequence with $L_2 = 1$ and $L_2 = 2$, respectively. We observe that the algorithm merged the two states 00 and 11 into a single state since only 1-step derived frequencies are checked for identifying the states.

We have demonstrated that the performance of the CRISSiS algorithm highly depends on the choice of L_1 and L_2. Smaller L_1 and L_2 may lead to a wrong identification of the system model. Theoretically, $L_1 = \frac{|Q|^3-|Q|}{6}$ [21] and $L_2 = |Q|$ would guarantee the algorithm to identify any synchronizable machines as long as the symbolic sequence is sufficiently long because PFSA has only finite number of states. However, we do not know the number of states of the true model and even if we know, the running time of CRISSiS goes exponentially as one increases L_1 and L_2. Instead, a more systematic way is that we start with smaller values of L_1 and L_2 ($L_1 = L_2 = 1$) and identify a model. Then we gradually increases L_1 and L_2 and see whether the new identified model matches with the previous model. By this means, one can roughly find out the smallest L_1 and L_2 that gives the correct model. In many practical problems, $L_1 = L_2 = 2$ is sufficient for identification of the dynamical system.

C. Comparison with CSSR

The CSSR algorithm is primarily designed to capture the sufficient statistics of a process up to an order L_{max} by finding out the casual states. In contrast to D-Markov algorithm [19] and variable-length Markov models (VLMMs) algorithm [22], each state in CSSR can contain multiple suffixes and this is achieved by some state merging procedure based on the 1st-order derived frequencies of the suffixes in CSSR. Hence, CSSR is nothing but a generalized D-Markov algorithm with some state merging procedure. CSSR is shown to be efficient in capturing the D-Markov machines of some particular order L as long as the parameter L_{max} is no less than L. However, we shall show that for those synchronizable machines that are not D-Markov models of any finite order, CSSR cannot obtain a compact representation in general but CRISSiS can.

For the three-state synchronizable model, which is an infinite order D-Markov model with finite representation in PFSA, we simulated 20 symbolic sequences of length $N = 10^4, 10^5$, and 10^6, respectively. The CSSR algorithm is applied to each simulated sequence with a different parameter L_{max} varying from 3 to 8. For comparison, the CRISSiS algorithm is also applied to the same sequences with $L_1 = L_2 = 1$. The χ^2 hypothesis testing method with confidence level 0.95 is used in both algorithms. We compute the average number of states for each data length as a measure of model complexity. Furthermore, to measure the prediction error of the model, we introduce the total-variation error, which is the L_∞ distance between the actual distribution of strings of length 10 and that from the constructed model. Table V-B gives the results from both algorithms. We observe that CSSR is not able to discover the minimal casual states. As L_{max} becomes large, the constructed machine by CSSR contains more and more number of states and the prediction error for strings of length 10 decreases. The reason is that each state contains infinite number of suffixes in the actual model. Although the prediction error gets improved as L_{max} increases but the constructed model becomes too complex to be useful in

TABLE IV
COMPARISON BETWEEN CRISSIS AND CSSR

	CSSR						CRISSiS
	$L_{max}=3$	$L_{max}=4$	$L_{max}=5$	$L_{max}=6$	$L_{max}=7$	$L_{max}=8$	$L_1=L_2=1$
$N=10^4$							
Averaged Number of States	7	10	13	17	24	35	4
Averaged Total-variation Error($\times 10^{-3}$)	2.9	2.2	2.2	1.8	1.7	1.6	1.9
$N=10^5$							
Averaged Number of States	7	12	18	28	39	57	3
Averaged Total-variation Error($\times 10^{-3}$)	2.1	1.5	1.5	0.7	0.6	0.5	0.5
$N=10^6$							
Averaged Number of States	7	12	20	32	49	74	3
Averaged Total-variation Error($\times 10^{-3}$)	2.1	1.5	1.4	0.7	0.5	0.3	0.2

practice. On the contrary, CRISSiS gives much better results. Except for $N = 10^4$ possibly due to the lack of data, CRISSiS successfully discovers the casual states. The prediction error is also very close to or better than that of CSSR with large L_{max}. Ideally, if we had access to the infinite length sequence, we would expect CRISSiS to recover the model perfectly.

VI. CONCLUSION & FUTURE WORK

This paper presents a new PFSA construction algorithm for symbolic sequences. The proposed CRISSiS algorithm searches for a synchronizing string for state localization and recursively identifies the structure of the unknown PFSA from the localized state. The algorithm is linear in the data length. The algorithm is shown to be able to capture the structure of all synchronizable machines, whose structures cannot be correctly discovered by CSSR or D-Markov. Hence, the CRISSiS algorithm is a more general approach than CSSR or D-Markov algorithms for PFSA construction.

A. Future Work

In the future, the following issues will be investigated:

1) Relaxation of the condition of having a synchronizing string for computation of PFSA descriptions from observed symbolic streams. This extension will allow the compression algorithm to capture all finite memory models that can possibly arise.
2) Apply the proposed construction in pattern recognition problems arising in sensing scenarios. Particularly, the problems of multi-modal sensing with heterogeneous sensing suites will be investigated in the context of identifying relevant patterns in the framework of probabilistic automata.
3) Generalization of the proposed algorithm to allow distributed computation of emergent patterns in data collected from possibly non-colocated sensors.
4) Develoment of rigorous methodologies and results to estimate the amount of data required to construct a valid PFSA model.

REFERENCES

[1] K. Murphy, "Passively learning finite automata," Santa Fe Institute, Tech. Rep., 1996. [Online]. Available: citeseer.ist.psu.edu/murphy95passively.html

[2] D. Angluin and C. H. Smith, "Inductive inference: Theory and methods," *ACM Comput. Surv.*, vol. 15, no. 3, pp. 237–269, 1983.

[3] V. Rajagopalan and A.Ray, "Symbolic time series analysis via wavelet-based partitioning," *Signal Processing*, vol. 86, no. 11, pp. 3309–3320, 2006.

[4] S. Lucas and T. Reynolds, "Learning deterministic finite automata with a smart state labeling evolutionary algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 7, pp. 1063–1074, July 2005.

[5] A. Paz, *Introduction to probabilistic automata (Computer science and applied mathematics)*. Orlando, FL, USA: Academic Press, Inc., 1971.

[6] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco, "Probabilistic finite-state machines - part i," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 7, pp. 1013–1025, July 2005.

[7] C. R. Shalizi, K. L. Shalizi, and J. P. Crutchfield, "An algorithm for pattern discovery in time series," *Technical Report, Santa Fe Institute*, October 2002. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0210025

[8] C. R. Shalizi and K. L. Shalizi, "Blind construction of optimal nonlinear recursive predictors for discrete sequences," in *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*. Arlington, Virginia, United States: AUAI Press, 2004, pp. 504–511.

[9] I. Chattopadhyay and A. Ray, "Structural transformations of probabilistic finite state machines," *International Journal of Control*, vol. 81, no. 5, pp. 820–835, May 2008.

[10] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[11] D. Heckerman and D. Geiger, "Learning Bayesian Networks," Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-95-02, December 1994. [Online]. Available: citeseer.ist.psu.edu/article/heckerman95learning.html

[12] F. Jelinek, J. D. Lafferty, and R. L. Mercer, "Basic methods of probabilistic context free grammars," *Speech Recognition and Understanding. Recent Advances, Trends, and Applications*, vol. F75, pp. 35–360, 1992.

[13] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco, "Probabilistic finite-state machines - part ii," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 7, pp. 1026–1039, July 2005.

[14] A. Corazza and G. Satta, "Probabilistic context-free grammars estimated from infinite distributions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 8, pp. 1379–1393, Aug. 2007.

[15] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech processing," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[16] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman, "A generalized hidden markov model for the recognition of human genes in dna." AAAI Press, 1996, pp. 134–142.

[17] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert, "Approaches to the automatic discovery of patterns in biosequences," *J Comput Biol.*, vol. 5, no. 2, pp. 279–305, 1998.

[18] J. P. Crutchfield and K. Young, "Inferring statistical complexity," *Physical Review Letters*, vol. 63, pp. 105–108, 1989.

[19] A. Ray, "Symbolic dynamic analysis of complex systems for anomaly detection," *Signal Processing*, vol. 84, no. 7, pp. 1115–1130, 2004.

[20] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation, 2nd ed.* Addison-Wesley, 2001.

[21] A. Trahtman, "The existence of synchronizing word and cerny conjecture for some finite automata," in *Second Haifa Workshop on Graph Theory, Combinatorics and Algorithms*, 2002.

[22] D. Ron, Y. Singer, and N. Tishby, "The power of amnesia: Learning probabilistic automata with variable memory length," *Machine Learning*, vol. 25, 1996.