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

State Splitting and State Merging in Probabilistic Finite State Automata★

Patrick Adenis Kushal Mukherjee Asok Ray
pxa154@psu.edu kum162@psu.edu axr2@psu.edu
The Pennsylvania State University
University Park, PA 16802, USA

***Abstract*— Probabilistic finite state automata (PFSA) are constructed from symbol sequences for modeling the behavior of dynamical systems. This paper presents construction of *finite history automata* from symbol sequences; such automata, called D-Markov machines, are structurally simple and computationally efficient. The construction procedure is based on: (i) *state splitting* that generates symbol blocks of different lengths according to their relative importance; and (ii) *state merging* that assimilates histories from symbol blocks leading to the same symbolic behavior. A metric on probability distribution of symbol blocks is introduced for trade-off between modeling performance and the number of PFSA states. These algorithms have been tested by two examples.**

***Index Terms*— Probabilistic Finite State Automata, D-Markov Machines, Symbolic Dynamics, State Splitting, State Merging**

I. Introduction

Probabilistic finite states automata (PFSA) [1] have been used for behavior modeling of dynamical systems in a variety of applications (e.g., anomaly detection [2] [3] and pattern recognition [4]). In these applications, the performance of PFSA has been comparable to that of existing techniques (e.g., Bayesian filters, Artificial Neural Networks, and Principal Component Analysis [5]. The procedure for construction of PFSA from the output of a dynamical system is as follows:

1) Coarse-graining of time series to convert the scalar or vector valued data into symbol sequences, where the symbols are drawn from a finite alphabet [6] [7].
2) Identification of statistical patterns from the symbol sequences [4].

In the process of symbol generation, the phase space of time series is partitioned into finitely many non-intersecting and exhaustive segments, each corresponding to a symbol of the alphabet. As the dynamical system evolves in time, it travels through or touches various partition segments in its phase space and the corresponding symbol is assigned to it. In this way, a time series is converted into a symbol sequence. A probabilistic finite state automaton (PFSA) is then used to encode the statistical behavior of this symbol sequence. The statistical patterns depict the dynamical system's behavior in a compact form. Figure 1 illustrates the concept.

★This work has been supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant No. W911NF-07-1-0376, and by the U.S. Office of Naval Research under Grant No. N00014-09-1-0688. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

A PFSA consists of a finite set of states Q that are connected by transitions [1]. Each transition corresponds to a symbol σ in the finite alphabet Σ. At each step, the automaton moves from one state to another (including a self loop) using these transitions, and thus generates a corresponding block of symbols $(s_i)_{i\in\mathbb{N}}$ so that the *probability distributions* over the set of all possible strings defined over the finite alphabet Σ are represented in the space of PFSA. The advantage of such a representation is that a PFSA is simple to encode as it is characterized by the set of states, transitions (one for each symbol $\sigma \in \Sigma$ and state $q \in Q$), and transition's probabilities.

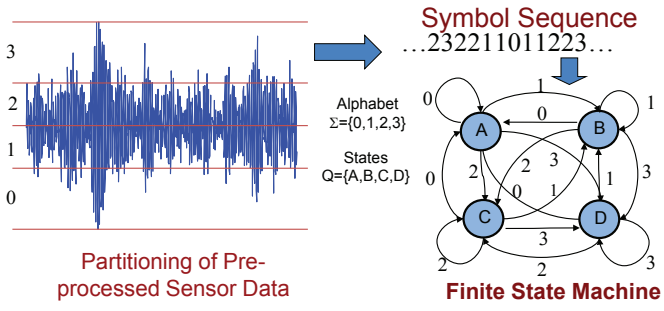

Fig. 1. Underlying concept of a PFSA

The main issue addressed in this paper is the identification of a PFSA model for representing the quasi-stationary probability distribution of the symbolic sequences obtained from a dynamical system.

In a PFSA, a transition from one state to another is independent of the previous history of states. Therefore, states and transitions form a Markov process, which is a special class of Hidden Markov Models [1]. However, from the perspectives of symbol generation, the states are implicit and generation of the next symbol may depend on the complete history of the symbol sequence. Given a probability-distribution, construction of an exact PFSA model appears to be computationally infeasible. That is the motivation of focusing on a certain class of PFSA, namely, the *D-Markov machines* [2] whose properties are briefly described below.

In a D-Markov machine, generation of the next symbol depends only on a *finite* history, i.e., a symbol block of length not exceeding D, where D is a positive integer, that is called the *depth* of the machine. Therefore, D-Markov machines belong to the class of shifts of finite type, i.e., shift spaces

that can be described by a finite set of forbidden symbol blocks [8].

Considering the set of all symbol blocks of length D as the set of states, one may directly construct a D-Markov Machine from a symbol sequence by frequency counting to approximate the probabilities of each transition [9]. Since the number of states increases exponentially as the depth D increases, state merging is necessary for PFSA with relatively large values of D. For example, with $|\Sigma| = 4$ symbols and a depth $D = 5$, the D-Markov machine could have at most $|\Sigma|^D = 1024$ states. Along this line, the major contribution of this paper is formulation of algorithms for:

1) Merging of (possibly redundant) states of the PFSA.
2) Retaining the D-Markov properties, subject to the constraint of a specified bound on the error between the constructed PFSA and the symbolic sequence.

This paper is organized into five sections including this introduction. Section II presents a brief background on PFSA including the definition of a D-Markov machine. Section III develops the algorithms of state splitting and state merging. Section IV presents two examples to explain and validate the algorithms; in the first example, the symbolic sequence is generated from a PFSA that is not D-Markov; and the second example is constructed from a *chaotic system*. Section V concludes this paper with recommendations for future research.

II. Background

This section presents pertinent information regarding D-Markov machines and other pertinent mathematical tools (e,g., entropy rate $H(\Sigma|Q)$ and the metric d) that are used to measure the effectiveness of the algorithms.

A. D-Markov Machines

A probabilistic finite state automaton (PFSA) [10] is a quadruple $K = (\Sigma, Q, \delta, \tilde{\pi})$, where

- Σ is a (nonempty) finite set, called alphabet;
- Q is a (nonempty) finite set, called set of states;
- $\delta : Q \times \Sigma \to Q$ is the state-transition map;
- $\tilde{\pi} : Q \times \Sigma \to [0, 1]$ is the probability matrix (also known as morph matrix) which satisfies to $\sum_{\sigma \in \Sigma} \tilde{\pi}(q, \sigma) = 1$ for all $q \in Q$.

A PFSA generates a symbol sequence $(s_i)_{i \in \mathbb{N}},\ s_i \in \Sigma$ on the underlying Markov-Process of states $(X_i)_{i \in \mathbb{N}},\ X_i \in Q$. The matrix $\tilde{\pi}$ implicitly alludes to the fact that the PFSA satisfies the *Markov condition*, where, generation of a symbol *only* depends on the previous *state*. However, if the state is unknown, the next symbol generation may depend on the complete past history of the symbols generated by the PFSA.

A D-Markov machine generates symbols that solely depend on the (immediate past) history of at most D symbols of the sequence. The positive integer D is called the *depth* of the machine. In other words, for any word[1] $w \in \Sigma^D$ of length D, $\delta^*(q, w)$ is independent of the state q, where δ^* denotes the extended state-transition function of the automaton [10]. Whatever the initial state q is, the finite sequence of transitions represented by the word $w \in \Sigma^D$ always leads to the same final state that could be represented by the word w itself. Consequently, in a D-Markov machine, a certain set of words in Σ^D can be associated to a state of the machine. Moreover, the state transition map δ can be automatically constructed from the words that correspond to individual states.

This paper considers the case where, given a symbol sequence, the task is to identify an underlying D-Markov machine model. Then, the morph matrix $\tilde{\pi}$ can be computed by observing the frequency of appearance of all the words [2].

B. Entropy rate

The entropy rate denoted as $H(\Sigma|Q)$ [11] represents the *predictability* of a machine given the previous state. The lower the entropy rate is, the more predictable is the machine conditioned on the previous state. As the entropy rate reaches 0, the machine is completely deterministic. The entropy rate is computed as follows:

$$\begin{aligned} H(\Sigma|Q) &\triangleq \sum_{q \in Q} \mathbb{P}(q) H(\Sigma|q) \\ &= -\sum_{q \in Q} \sum_{\sigma \in \Sigma} \mathbb{P}(q) \mathbb{P}(\sigma|q) \log \mathbb{P}(\sigma|q) \\ &= -\sum_{q \in Q} \sum_{\sigma \in \Sigma} \wp(q) \tilde{\pi}(q, \sigma) \log \tilde{\pi}(q, \sigma) \end{aligned} \tag{1}$$

where $\wp$ is the stationary state probability vector of the PFSA, which represents the probability of being in a state at any instant of time.

C. Distance d between two PFSA

A metric is introduced to measure the distance between two PFSA K_1 and K_2.

Let $P_i(\Sigma^\ell) \triangleq [\mathbb{P}_i(w)]_{w \in \Sigma^\ell},\ i = 1, 2$, be the steady state probability of generating any word of length ℓ from the PFSA K_i. The metric is then computed as:

$$d(K_1, K_2) \triangleq \sum_{i=1}^{+\infty} \frac{\left\| P_1(\Sigma^\ell) - P_2(\Sigma^\ell) \right\|_1}{2^{i+1}} \tag{2}$$

where $\|\cdot\|_1$ is the traditional L_1-norm, which is guaranteed to converge due to the dominating weight $\frac{1}{2^{i+1}}$ and satisfies the relation $0 \leq d(\cdot, \cdot) \leq 1$.

Since this metric assigns more weight to shorter words, the infinite sum could be truncated to a relatively small order D (typically 8 or 9) for a given tolerance $\varepsilon \ll 1$. This implies that the distance effectively compare the probability of generating words of length D, and is therefore especially adaptable to D-Markov machines whose dynamical behavior is characterized by words of a given maximal depth.

Remark 1: The metric d can also be used to calculate the distance between a PFSA and a symbol sequence, in which case the probabilities are expressed in terms of relative frequency of appearance of each word. In fact, it has been shown that this metric measures the distance between probability-distributions [1].

[1] Σ^ℓ denotes the set of all the words of length ℓ made from the alphabet Σ, and Σ^* is the set of all *finite-length* words including the empty word ϵ.

III. Algorithm Development

This section develops the algorithms to generate reduced-order D-Markov machines. The procedure consists of two major steps, namely *state splitting* and *state merging*.

State splitting increases the number of states to gain more precision in the representation of the data sequence. This is performed by splitting the states that minimize the entropy rate $H(\Sigma|Q)$, thereby, allowing to focus only on the most critical states. Although this process is executed by controlling the exponential growth of states with increasing depth D, the D-Markov machine still may have a large number of states. The subsequent state merging algorithm reduces the number of states in a D-Markov machine, say K_1, by merging those states that behave similarly. This finally leads to a reduced order c from the original D-Markov machine K_1.

A. State Splitting Algorithm

In D-Markov machines, a finite symbol sequence of length D is sufficient to describe the current state. This implies that the number of states of a D-Markov machine of depth D is bounded by $|\Sigma|^D$, where $|\Sigma|$ is the cardinality of the alphabet Σ. As this relation is exponential in nature, the number of states rapidly explodes as D is increased. However, form the perspective of modeling a symbol sequence, some states may be more important than others. Therefore, it is advantageous to have a set of states that correspond to symbol sequences of variable lengths. This is accomplished by starting off with the simplest set of states (i.e., $Q = \Sigma$ for $D = 1$) and subsequently splitting the existing states that result in the largest decrease of the entropy rate. The process of splitting a state $q \in Q$ is done by replacing q by its *branches* as described by the set $\{\sigma q : \sigma \in \Sigma\}$. Maximum reduction of the entropy rate is the governing criterion for selecting the state to split. In addition, the generated set of states must satisfy the self-consistency criterion, which only permits a unique transition to emanate from a state for a given symbol. If $\delta(q, \sigma)$ is not unique $\forall \sigma \in \Sigma$, then the state q is split further. The state-splitting algorithm is described in Algorithm 1.

Figure 2 illustrates the process of state-splitting in a PFSA whose alphabet is $\Sigma = \{0, 1\}$. The notation $\Sigma^* w$ refers to the D-Markov state consisting of all symbol sequences with the word w as the suffix. The final states have been marked in ellipses. In the third layer (i.e., $D = 2$) from the top in Figure 2, the states are $\Sigma^\star 00$, $\Sigma^\star 10$, $\Sigma^\star 01$, and $\Sigma^\star 11$, of which all but $\Sigma^\star 10$ are terminated as final states. Consequently, the state $\Sigma^\star 10$ is further split as $\Sigma^\star 010$ and $\Sigma^\star 110$ that are terminated as final states. Hence, $Q = \{\Sigma^\star 00,\ \Sigma^\star 01,\ \Sigma^\star 11,\ \Sigma^\star 010,\ \Sigma^\star 110\}$ as seen in Figure 2.

Given an alphabet Σ and an associated set Q of states, the symbol generation probability matrix $\tilde{\pi}$ is computed at every stage as follows:

$$\tilde{\pi}(\sigma, q) = P(\sigma|q) = \frac{P(q\sigma)}{P(q)} \tag{3}$$

Algorithm 1 State splitting

Input: Symbol sequence $s_1 s_2 s_3 \ldots\ldots$,
alphabet set Σ
User defined Input: Max. num. of states η_s,
threshold η_t
Output: PFSA $K_1 = \{\Sigma, Q, \delta, \tilde{\pi}\}$
Initialize: Create a 1-Markov machine $Q^* := \Sigma$
repeat
  $Q := Q^*$
  $Q^* = \arg\min_{Q'} H(\Sigma|Q')$
  where, $Q' = Q \backslash q \cup \{\sigma q : \sigma \in \Sigma\}$ and $q \in Q$
until $|Q^*| < \eta_s$ or $H(\Sigma|Q) - H(\Sigma|Q^*) < \eta_t$
for all $q \in Q^*$ and $\sigma \in \Sigma$ **do**
  if $\delta(q, \sigma)$ is not unique **then**
    $Q^* := Q^* \backslash q \cup \{\sigma q : \sigma \in \Sigma\}$
  end if
end for
return $K_1 = \{\Sigma, Q, \delta, \tilde{\pi}\}$

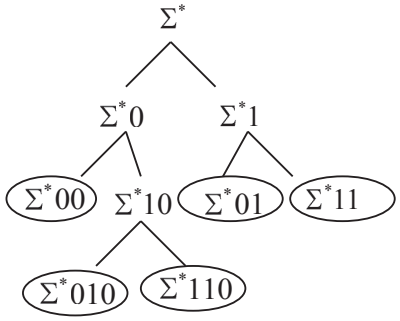

Fig. 2. Tree-representation of state splitting in D-Markov machines

where $\sigma \in \Sigma$, $q \in Q$, and $P(w)$ is the probability of observing a word w in the symbol sequence. An estimate of the morph matrix $\tilde{\pi}$ is the ratio of number of times the state q is followed by the symbol σ, denoted as $N(q\sigma)$, to the number of times the state q occurs ($N(q)$). Therefore,

$$\tilde{\pi}(\sigma, q) \simeq \frac{N(q\sigma)}{N(q)} \tag{4}$$

B. State Merging Algorithm

Once state splitting is performed, the resulting D-Markov machine represents the statistical characteristics of the symbol sequence. Depending on the alphabet size $|\Sigma|$ and depth D, the number of states after splitting may run into hundreds. Although, increasing the number of states of the machine allows for better representation of the sequence, it creates a rapidly increasingly large computational load and memory requirements. The motivation behind the state merging is to reduce the number of states, while preserving the D-Markov structure of the machine. Of course, such a process may cause the PFSA to have degraded precision due to loss of information. This algorithm attempts to minimize this loss.

In the merging algorithm, a stopping rule is constructed by specifying a certain acceptable threshold η on the distance d

between the *merged PFSA* and the original *data sequence*. An alternative stoping rule for the algorithm is to provide a maximal number of states $N_{\max}$ instead of the threshold η.

1) Merging two states: Before the complete state merging algorithm is explained, this subsection details the conditions under which two given states may be merged. The procedure for merging of two states is also delineated.

The process of state merging is addressed by creating an equivalence relation [12] (denoted as $\sim$) between the states. An equivalence relation specifies which states are identified together, thereby partitioning the original set of states into a smaller number of equivalence classes of states, each being a union of original states. The new states are, in fact, equivalence classes as defined by $\sim$.

Let $K_1 = \{\Sigma, Q_1, \delta_1, \tilde{\pi}_1\}$ be the *split PFSA*, and let $q, q' \in Q_1$ be two states that are to be *merged* together. Initially, an equivalence relation is constructed, where none of the states are equivalent to any other state except itself. To proceed with merging q and q', an equivalence relation is imposed between q and q', denoted as $q \sim q'$; however, the transition between original states may not be well-defined anymore, in the following sense: there may exist a $\sigma \in \Sigma$ such that states $\delta_1(q, \sigma)$ and $\delta_2(q', \sigma)$ are not equivalent. In other words, the same symbol may cause a transition to two different states from the merged state $\{q, q'\}$. As the structure of D-Markov machines does not permit this ambiguity [2], these states $\delta_1(q, \sigma)$ and $\delta_2(q', \sigma)$ are *forced* to be merged together, i.e., $\delta_1(q, \sigma) \sim \delta_2(q', \sigma)$. This process is recursive and must be performed until ambiguity in state transitions does not occur. Indeed at each iteration, we reduce the number of states of the future machine, and the machine where all the states are merged is always consistent. Therefore the number of states is a decreasing sequence of positive integers, which must eventually converge. (See Algorithm 2 for the details.)

The state-transition map δ_2 and associated transition probabilities $\tilde{\pi}_2$ for the merged PFSA are defined on $Q_2 \triangleq Q_1/\sim$, the quotient set. If $[q] \in Q_2$ denotes the equivalence class of $q \in Q_1$, then the associated morph matrix $\tilde{\pi}_2$ is obtained by:

$$\begin{aligned}\tilde{\pi}_2([q], \sigma) &= \mathbb{P}\left[s_{i+1} = \sigma \mid \bigcup_{\tilde{q} \in [q]} \{X_i = \tilde{q}\}\right] \\ &= \frac{\sum_{\tilde{q} \in [q]} \mathbb{P}\left[s_{i+1} = \sigma \ ; \ X_i = \tilde{q}\right]}{\sum_{\tilde{q} \in [q]} \mathbb{P}(X_i = \tilde{q})} \\ &= \frac{\sum_{\tilde{q} \in [q]} \tilde{\pi}_1(\tilde{q}, \sigma) \times \wp_1(\tilde{q})}{\sum_{\tilde{q} \in [q]} \wp_1(\tilde{q})} \end{aligned} \tag{5}$$

Hence $\tilde{\pi}_2$ is simply the weighted sum of $\tilde{\pi}_1$ by $\wp_1$ the stationary-probabilities of K_1.

By construction, δ_2 is naturally obtained by:

$$\delta_2([q], \sigma) = [\delta_1(q, \sigma)] \tag{6}$$

Algorithm 3 explains the procedure to obtain the PFSA, where we want q_a and q_b to be merged.

Algorithm 2 Minimal equivalence relation given $q \sim q'$

Input: δ, q, q', Initial equivalence relation $\sim$
Output: Updated equivalence relation $\sim$
NOTE: Recursive function $(\sim) := \texttt{Merge}(\delta, q, q', \sim)$
Set $q \sim q'$;
for all $\sigma \in \Sigma$ **do**
  if $\delta(q_a, \sigma) \nsim \delta(q_b, \sigma)$ **then**
    Set $\sim := \texttt{Merge}(\delta, \delta(q, \sigma), \delta(q', \sigma), \sim)$;
  end if
end for
return $\sim$

Algorithm 3 Minimal PFSA K_2 with q_a and q_b merged

Input: $K_1 = \{\Sigma, Q_1, \delta_1, \tilde{\pi}_1\}$, q_a, q_b
Output: Merged PFSA $K_2 = \{\Sigma, Q_2, \delta_2, \tilde{\pi}_2\}$
Compute $\sim$ using algo.2;
Set $Q_2 := Q_1/\sim$;
Compute $\wp_1$ the stationary-probability of K_1;
for all $[q] \in Q_2$ **do**
  for all $\sigma \in \Sigma$ **do**
    Set $\delta_2([q], \sigma) := [\delta_1(q, \sigma)]$;
    Compute $\tilde{\pi}_2([q], \sigma)$ using (5);
  end for
end for
return $K_2 = \{\Sigma, Q_2, \delta_2, \tilde{\pi}_2\}$

2) State Merging Algorithm: The aim of this algorithm is to decide which states have to be merged. States that behave similarly (i.e., have similar symbol generation probabilities) have a higher priority for merging. A norm $H(q, q') \triangleq \|\tilde{\pi}(q, \cdot) - \tilde{\pi}(q', \cdot)\|_1$ is defined to measure the similarity of two states in terms of future symbol generation. A small value of $H(q, q')$ indicates that the two states have very close probabilities of generating each symbol σ. Note that this norm is upper bounded, $H(q, q') \leq 2$.

First, the two closest states are merged using Algorithm.3. Subsequently, distance d (see subsection II-C) of this merged PFSA from the initial data sequence is evaluated. If the distance is less than a threshold η, this machine is kept and the states next on the priority are merged. On the other hand, if the distance d is greater that the threshold, the process of merging the two particular states is aborted and pair of states with the next smallest value of $H(q, q')$ are selected for merging. This procedure is terminated if no such pair of states exist, for which the distance d between the states is less that η. Details of the procedure are given in Algorithm.4.

Indeed, for any word $w \in \Sigma^D$, the extended transition map $\delta^*_{K_1}(q, w)$ is independent of q for K_1; and it is easy to check that $\delta^*_{K_2}([q], w)$ is also independent of $[q]$, since $\delta^*_{K_2}([q], w) = [\delta^*_{K_1}(q, w)]$. Thus, the D-Markov property of the automaton is preserved.

IV. Examples

This section described two illustrative examples. In the first example, a symbolic sequence is generated from a non-

Algorithm 4 Merging algorithm

Input: PFSA $K_1 = \{\Sigma, Q_1, \delta_1, \tilde{\pi}_1\}$, threshold η, symbol sequence (s_i)
Output: Merged PFSA $K_2 = \{\Sigma, Q_2, \delta_2, \tilde{\pi}_2\}$
Set $K_2 := K_1$
for all $q, q' \in Q_2$ **do**
  if $q \neq q'$ **then**
    Set LIST_STATES$(q, q') = H(q, q')$;
  else
    Set LIST_STATES$(q, q') = 2$;
  end if
end for
sort(LIST_STATES);
Set $(q, q') :=$ **pop**(LIST_STATES);
loop
  Compute K_3 from K_2, q and q' using Algorithm.3;
  if $d\,[K_3, (s_i)] < \eta$ **then**
    Set $K_2 := K_3$;
    Recompute LIST_STATES;
    Set $(q, q') :=$ **pop**(LIST_STATES);
  else
    Set $(q, q') :=$ **pop**(LIST_STATES);
    if q==q' **then**
      Break loop;
    end if
  end if
end loop
return K_2

D-Markov PFSA and the symbol sequence is modeled as a reduced order D-Markov model. In the second example, a (real-valued) data sequence, generated from a chaotic dynamical system, is partitioned. The resulting symbolic sequence is modeled as a D-Markov process by using the algorithms developed in the previous section.

A. Modeling Sequences from a Non-D-Markov PFSA

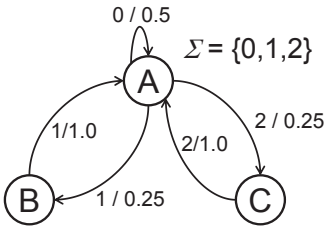

Fig. 3. The PFSA K_0 to generate the symbol sequences in Example 1

The PFSA K_0, presented in Figure 3, is used to generate a data sequence (1,000,000 points in the sequence). K_0 is a variation of the *even shift* machine with three symbols [8].

The state splitting algorithm(1) is used to obtain a D-Markov PFSA (K_1) with depth $D = 8$ and $Q| = 3$ states from the symbol sequence. Note that, without sequential state splitting, the PFSA would have at most $|\Sigma|^D = 3^8 = 6561$ possible states). The evolution of the entropy rate during the splitting process is presented on Figure 4. Moreover, the entropy rate of K_1tends to that of initial machine K_0 as the number of states increases. This is evident from the fact K_1 would represent K_0 better if the number of states is increased.

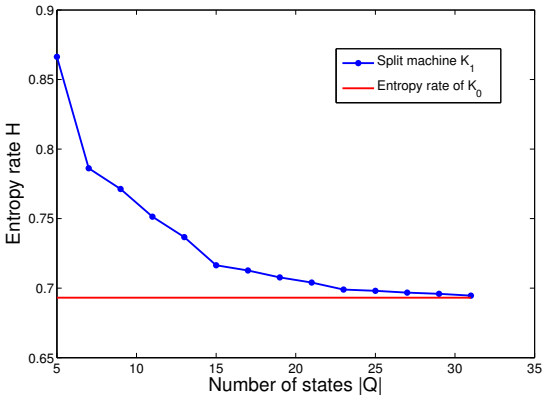

Fig. 4. Entropy rate in Example 1

The algorithm splits the states $111\ldots1$ and $222\ldots2$ that are the non-synchronizing words[2] of the PFSA K_0. State splitting is continued until the probability of being in one of these states becomes very low. The state merging algorithm (see Subsection III-B.2)) is used to construct a reduced order D-Markov PFSA (K_2).

TABLE I
RESULTS OF EXAMPLE 1

	$d(\cdot, K_0)$	$d(\cdot, \{s_n\})$	$\|Q\|$	depth
K_0	0	0.1004	3	non D-Markov
K_1	0.0892	0.0293	31	8
K_2	0.8075	0.7875	11	8

For Example 1, Table I summarizes the following results: – distance of D-Markov PFSA (K_1) and distance of the reduced order D-Markov PFSA (K_2) from the original PFSA K_0; distances of K_0, K_1, and K_2 from the generated symbol sequence $\{s_n\}$; the number of states $|Q|$; and the depth of the D-Markov machine, if applicable.

Although there is loss of information by modeling K_0 as K_2, the results show that this loss is minimal (see Table I). The D-Markov machine has a depth $D = 8$ and its order is diminished to only 11 states from $2^8 = 256$. This example illustrates how a non-D-Markov PFSA can be modeled by a significantly reduced order D-Markov machine by making trade-off between number of D-Markov states and good precision.

B. Modeling Sequences Generated from a Chaotic System

The sequence $\{x_n\}$ is generated iteratively from a logistic map with the initial state $x_1 \triangleq 0.5$ and the iterative map $x_{k+1} \triangleq r x_k(1 - x_k)$, where the parameter r is chosen to be $r = 3.75$, and the iterations x_n always lies in the interval

[2]Non-synchronizing words are those that do not uniquely determine the current state of the PFSA.

$[0, 1]$. The space $[0, 1]$ is partitioned into three mutually disjoint intervals that are associated to the symbol 0, 1 or 2, respectively. The boundaries of disjoint intervals are shown in black dotted-line in Figure 5. A symbol sequence $(s_n) \in \{0, 1, 2\}$ is generated by replacing the value of x_n by the corresponding symbol associated with the partition within which x_n lies.

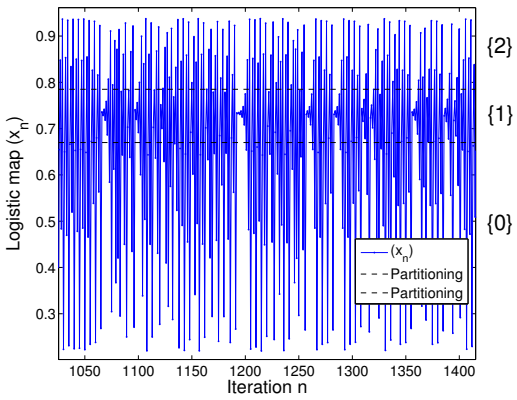

Fig. 5. Output $\{x_n\}$ of the logistic map

As seen in Figure 5, this map exhibits two types of behaviors. The first is an oscillation between a high value and a low value, and the second is oscillations of small amplitude around 0.73.

...2 0 1 1 0 2 0 2 0 1 1 1 1 1 0 2 0 2 0 1 0 2 0 2 0 0 2 0 2 0 1 2 0 2 0 2 0 2 0 0 2
0 2 0 1 2 0 2 0 1 1 1 2 0 2 0 2 0 2 0 0 2 0 2 0 0 2 0 2 0 1 2 0 2 0 1 2 0 2 0 1 1 1 1
1 1 2 0 2 0 1 1 1 1 1 1 2 0 2 0 1 0 2 0 2 0 0 2 0 2 0 1 2 0 2 0 2 0 2 0 0 2 0 2 0 0 2
0 2 0 0 2 0 2 0 0 2 0 2 0 1 1 0 2 0 2 0 1 1 1 2 0 2 0 1 2 0 2 0 1 2 0 2 0 1 1 1 ...

Fig. 6. Excerpt of the symbol sequence $\{s_n\}$

Figure 6 shows an excerpt of the symbol sequence, where the two behaviors are well reproduced in terms of '0202' sequences alternating with '11' sequences.

Figure 7 shows that the reduced order PFSA K_2 is capable of capturing these two behaviors. The state E in Figure 7 randomly switches between the '02' swapping mode (in blue-green) and the '11' mode (in red).

TABLE II

RESULTS OF EXAMPLE 2

	$d(\cdot, (s_n))$	$\lvert Q \rvert$	depth
K_1	0.0124	55	10
K_2	0.2689	8	6

The pertinent results of this state merging are presented in Table II, where it is seen that the merging step allows reduction of the number of states from 55 to 8 while maintaining a specified bound on the metric d.

V. SUMMARY, CONCLUSIONS, AND FUTURE WORK

This paper presents the underlying concepts and algorithms for modeling dynamical systems as *finite history automata* from symbol sequences. These automata, called D-Markov machines [2], are structurally simple and computationally efficient to execute. While a large depth in

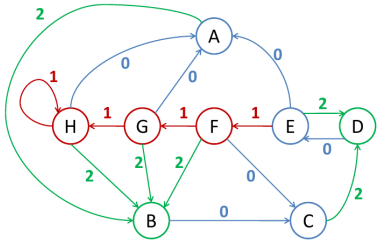

Fig. 7. The D-Markov PFSA K_2 obtained after state merging

D-Markov machines provides a longer history resulting in (possibly) better model prediction, it is accompanied by exponential growth of the number of PFSA states. The proposed algorithm focuses on order reduction in D-Markov machines.

The state merging algorithm, presented here, is heuristic and therefore suboptimal. An important topic for future research is to further investigate this issue and justify the general problem of approximation of a probability-distribution by a PFSA, and more precisely by a D-Markov machine. In this regard, the key topics of future research are as follows.

1) Development of a rigorous mathematical framework for approximation of probability-distribution by a general class of PFSA.
2) Identification of a general relation between the bound of modeling error and the number of states of the D-Markov machine.

REFERENCES

[1] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco, "Probabilistic finite-state machines: Parts I and II," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1013–1039, 2005.

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

[3] S. Gupta and A. Ray, "Symbolic dynamic filtering for data-driven pattern recognition," *PATTERN RECOGNITION: Theory and Application, Chapter 5, Nova Science Publishers, Hauppage, NY, USA ISBN 978-1-60021-717-3*, pp. 17–71, 2007.

[4] ——, "Staistical mechanics of complex systems for pattern identification," *Journal of Statistical Physics*, vol. 134, pp. 337–364, 2009.

[5] C. Rao, A. Ray, S. Sarkar, and M. Yasar, "Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns," *Signal, Image, and Video Processing*, vol. 3, no. 2, pp. 101–114, 2009.

[6] T. W. Liao, "Clustering of time series data-a survey," *Pattern Recognition*, vol. 38, pp. 1857–1874, 2005.

[7] X. Jin, K. Mukherjee, S. Gupta, and A. Ray, "Wavelet-based feature extraction using probabilistic finite state automata for pattern classification," *Pattern Recognition, in press, doi: 10.1016/j.patcog.2010.12.003*, 2011.

[8] D. Lind and B. M. Rudin, *Symbolic Dynamics and Coding*. Cambridge University Press, Cambridge, UK, 1995.

[9] Y. Wen and A. Ray, "A stopping rule for symbolic dynamic filtering," *Applied Mathematics Letters*, vol. 23, pp. 1125–1128, 2010.

[10] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation 2nd ed.* New York, NY, USA: ACM, 2001, pp. 45–138.

[11] T. Cover and J. Thomas, *Elements of Information Theory, 2nd ed.* Wiley, Hoboken, NJ, 2006.

[12] P. R. Halmos, *Naive Set Theory*. Princeton, NJ: Van Nostrand, 1960, pp. 26–29.