

Structured Hidden Markov Model: a General Framework for Modeling Complex Sequences

Ugo Galassi, Attilio Giordana, and Lorenza Saitta

Dipartimento di Informatica, Università Amedeo Avogadro
Via Bellini 25G, Alessandria, Italy

Abstract. Structured Hidden Markov Model (S-HMM) is a variant of Hierarchical Hidden Markov Model that shows interesting capabilities of extracting knowledge from symbolic sequences. In fact, the S-HMM structure provides an abstraction mechanism allowing a high level symbolic description of the knowledge embedded in S-HMM to be easily obtained. The paper provides a theoretical analysis of the complexity of the matching and training algorithms on S-HMMs. More specifically, it is shown that Baum-Welch algorithm benefits from the so called locality property, which allows specific components to be modified and retrained, without doing so for the full model. The problem of modeling duration and of extracting (embedding) readable knowledge from (into) a S-HMM is also discussed.

1 Introduction

Since their introduction, Hidden Markov Models (HMMs) proved to be a fundamental tool in solving real-world pattern recognition problems, notably speech recognition [14] and DNA analysis [4]. However, HMMs are stochastic models including a potentially high number of parameters; hence, many research efforts have been devoted to constrain their structure in such a way to reduce the complexity of the parameter estimation task. To this aim have been proposed, for instance, the *hierarchical* HMM [5, 12] and the *factorial* HMM [8],

In this paper we aim at reducing the generality of the HMM's structure as well, but with an additional goal with respect to previous works: in fact, not only parameter estimation must be efficiently performed, but also the model itself should offer a high level, interpretable description of the knowledge it encodes, in a way understandable by a human user. In several application domains (*e.g.*, Molecular Biology [4]), this requirement is of primary concern when evaluation of the model has to be done by humans. Moreover, in this way, domain knowledge provided by an expert could be easily integrated, as well.

More specifically, this paper investigates a variant of HMM called *structured* HMM [7] (S-HMM in the following). An S-HMM is a graph built up, according to precise composition rules, with several "independent" sub-graphs (sub-models).

After a brief introduction of the S-HMM, its properties are formally analyzed: first of all it will be shown how an S-HMM can be locally trained using the classical Baum-Welch algorithm, considering only a subset of the sub-models

occurring in the compound one. A nice consequence of this property is that an S-HMM can be constructed and trained incrementally, by adding new sub-models or revising existing ones as new information comes in. A newly added sub-model may have different origins: for instance, it may be provided by an expert as a-priori knowledge, or it can be produced by an independent learning process. An interesting property of an S-HMM is that sub-models may also correspond to *gaps* in the observed sequences, *i.e.*, to not meaningful or not interesting regions. This ability elegantly solves the problem of building models of complex and *sparse* patterns. Finally, S-HMM is compared to other existing approaches such HHMM [5].

2 The Structured HMM

The basic assumption underlying an S-HMM (see Bouchaffra and Tan [3]) is that a sequence $O = \{o_1, o_2, o_3, \dots, o_T\}$ of observations could be segmented into a set of subsequences O_1, O_2, \dots, O_N , each one generated by a sub-process with only weak interactions with its neighbors. This assumption is realistic in many practical applications, such as, for instance, speech recognition [14, 15], and DNA analysis [4]. In speech recognition, regions are phonetic segments, like syllables, corresponding to recurrent structures in the language. In DNA, they may be biologically significant segments (*motifs*) interleaved with non-coding segments (such as *junk-DNA*). S-HMMs aim exactly at modeling such kind of processes, and, hence, they are represented as directed graphs, structured into sub-graphs (*blocks*), each one modeling a specific kind of sub-sequences.

Informally, a block consists of a set of states, only two of which (the *initial* and the *end* state) are allowed to be connected to other blocks. As an S-HMM is itself a block, a nesting mechanism is immediate to define.

2.1 Structure of a Block

In this section, a formal definition of S-HMM will be provided. Adopting the notation used in [14], O will denote a sequence of observations $\{o_1, o_2, \dots, o_T\}$, where every observation o_t is a symbol v_k chosen from an alphabet V . An HMM is a stochastic automaton characterized by a set of states Q , an alphabet V , and a triple $\lambda = \langle A, B, \pi \rangle$, being:

- $A : Q \times Q \rightarrow [0, 1]$ a probability distribution, a_{ij} , governing the transition from state q_i to state q_j ;
- $B : Q \times V \rightarrow [0, 1]$ a probability distribution, $b_i(v_k)$, governing the emission of symbols in each state $q_i \in Q$;
- $\pi : Q \rightarrow [0, 1]$ a distribution assigning to each state $q_i \in Q$ the probability of being the start state.

A state q_i will be said a *silent* state if $\forall v_k \in V : b_i(v_k) = 0$, *i.e.*, q_i does not emit any observable symbol. When entering a silent state, the time counter must not be incremented.

Definition 1. A basic block of an S-HMM is a 4-tuple $\lambda = \langle A, B, I, E \rangle$, where $I, E \in Q$ are silent states such that: $\pi(I) = 1, \forall q_i \in Q : a_{iI} = 0$, and $\forall q_i \in Q : a_{Ei} = 0$.

In other words, I and E are the input and the output states, respectively. Therefore, a composite block can be defined by connecting, through a transition network, the input and output states of a set of blocks.

Definition 2. Given an ordered set of blocks $\Lambda = \{\lambda_i | 1 \leq i \leq N\}$, a composite block is a 4-tuple $\lambda = \langle A_I, A_E, I, E \rangle$, where:

- $A_I : \mathbf{E} \times \mathbf{I} \rightarrow [0, 1], A_E : \mathbf{I} \times \mathbf{E} \rightarrow [0, 1]$ are probability distributions governing the transitions from the output states \mathbf{E} to the input states \mathbf{I} , and from the input states \mathbf{I} to the output states \mathbf{E} of the component blocks Λ , respectively.
- For all pairs $\langle E_i, I_j \rangle$ the transition probability $a_{E_i I_j} = 0$ if $j \neq i$.
- $I \equiv I_1$ and $E \equiv E_N$ are the input and output states of the composite block, respectively.

According to Definition 2 the components of a composite block can be either basic blocks or, in turn, composite blocks. In other words, composite blocks can be arbitrarily nested. Moreover, we will keep the notation S-HMM to designate non-basic blocks only.

As a special case, a block can degenerate to the *null block*, which consists of the start and end states only, connected by an edge with probability $a_{IE} = 1$. The *null block* is useful to provide a dummy input state I or a dummy output state E , when no one of the component block is suited to this purpose.

An example of S-HMM structured into three blocks $\lambda_1, \lambda_2, \lambda_3$, and two *null blocks* λ_0, λ_4 , providing the start and the end states, is described in Figure 1.

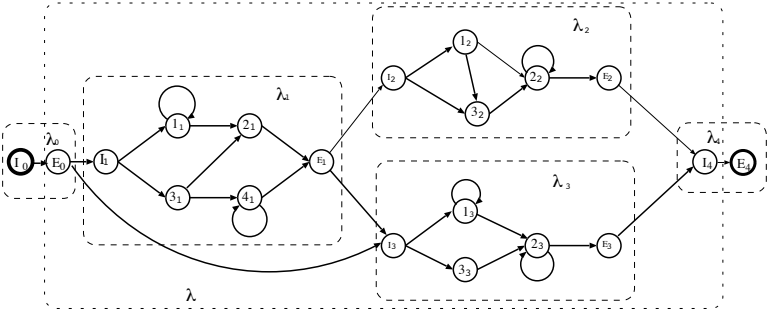


Fig. 1. Example of Structured Hidden Markov Model composed of three interconnected blocks, plus two *null blocks*, λ_0 and λ_4 , providing the start and end states. Distribution A is non-null only for explicitly represented arcs.

2.2 Estimating Probabilities in S-HMM

As formally stated in [14], three problems are associated with the HMM approach:

1. given a model λ and a sequence of observations O , compute the probability $P(O|\lambda)$;
2. given a model λ and a sequence of observations O , assumed to be generated by λ , compute the most likely sequence of states in λ ;
3. given a model λ and a sequence of observations O (or a set of sequences [15]), estimate the parameters in λ in order to maximize $P(O|\lambda)$.

The classical solution to Problem 1 and 3 relies on two functions α and β , plus other auxiliary functions γ and ξ , defined on α and β . The classical solution to Problem 2 relies on Viterbi algorithm [6], which implements a function computationally analogous to α . In the following we will extend α and β to S-HMMs in order to prove some properties to be exploited by an incremental learning algorithm.

Given a sequence of observations $O = \{o_1, o_2, \dots, o_t, \dots, o_T\}$ and a model λ , the function $\alpha_t(i)$ computes, for every time t ($1 \leq t \leq T$), the joint probability of being in state q_i and observing the symbols from o_1 to o_t .

Let us consider an S-HMM λ containing N blocks. We want to define the recursive equations allowing α to be computed. In order to do this, we have to extend the standard definition in order to include silent states: when leaving a silent state, the time counter is not incremented. When entering the block $\lambda_k = \langle A_k, B_k, I_k, E_k \rangle$ with N_k states, at time r ($1 \leq r \leq T$), the following equations are to be used:

$$\begin{aligned}
 \alpha_r(I_k) &= P(o_1, \dots, o_r, q_r = I_k) \\
 \alpha_t(j) &= \alpha_{t-1}(I_k) a_{I_k j}^{(k)} b_j^{(k)}(o_t) + \sum_{i=1}^{N_k} \alpha_{t-1}(i) a_{ij}^{(k)} b_j^{(k)}(o_t) \\
 &\quad (r+1 \leq t \leq T, 1 \leq j \leq N_k, q_j \neq I_k, q_j \neq E_k) \\
 \alpha_t(E_k) &= \alpha_t(I_k) + \sum_{i=1}^{N_k} \alpha_t(i) a_{i E_k}^{(k)}
 \end{aligned} \tag{1}$$

Notice that the above equations only depends upon external states through the values of $\alpha_r(I_k)$ ($1 \leq r \leq T$) computed for the input state; moreover, the block propagates $\alpha_t(E_k)$ ($1 \leq t \leq T$) to the following blocks only through the output state. Finally, $\alpha_1(I_1) = 1$ and $\alpha_T(E_N) = P(O|\lambda)$.

Function $\beta_t(i)$ is complementary to $\alpha_t(i)$, and computes the probability of observing the symbols $o_{t+1}, o_{t+2}, \dots, o_T$, given that q_i is the state at time t . For β a backward recursive definition can be given:

$$\begin{aligned}
 \beta_r(E_k) &= P(o_{r+1}, \dots, o_T | q_r = E_k) \\
 \beta_t(i) &= \beta_{t+1}(E_k) a_{i E_k}^{(k)} + \sum_{j=1}^{N_k} \beta_{t+1}(j) b_j^{(k)}(o_{t+1}) a_{ij}^{(k)} \\
 &\quad (1 \leq t \leq r-1, 1 \leq i \leq N_k, q_i \neq E_k, q_i \neq I_k) \\
 \beta_t(I_k) &= \beta_t(E_k) + \sum_{j=1}^{N_k} \beta_t(j) a_{I_k j}^{(k)}
 \end{aligned} \tag{2}$$

From equations (2), it follows that $P(O|\lambda) = \beta_1(I_1)$.

Definition 3. An S-HMM is said a forward S-HMM when for all non-basic blocks the matrix A_I and A_E define a directed acyclic graph.

For a forward S-HMM it is easy to prove the following theorem.

Theorem 1. In a forward S-HMM, the complexity of computing functions α and β is:

$$C \leq T(\sum_{h=1}^{N_C} N_h^2 + M \sum_{k=1}^N N_k^2)$$

being N_h the dimension of matrix $A_I^{(h)}$ of the h -th block, M the cardinality of the alphabet, N_C the number of composite blocks, and N the number of basic blocks.

Proof. Notice that the second summation in the right-hand side of the formula corresponds to the computation of α and β inside the basic blocks, whereas the first summation is due to the block interconnection. Following the block recursive nesting and starting from the basic blocks, we observe that, in absence of any hypothesis on distribution A , each basic block is an HMM, whose complexity for computing α and β is upperbounded by $N_k^2 MT$ [14]. As the global network interconnecting the basic blocks is a directed forward graph, every basic block needs to be evaluated only once.

Let us consider now a composite block; the interconnecting structure is an oriented forward graph, by definition, and, then, equations (1) and (2) must be evaluated only once on the input (output) of every internal block S-HMM $_h$. As a conclusion, the complexity for this step is upperbounded by TN_h^2 .

3 S-HMMs are locally trainable

The classical algorithm for estimating the probability distributions governing state transitions and observations are estimated by means of the Baum-Welch algorithm [2, 14], which relies on the functions α and β defined in the previous section. In the following we will briefly review the algorithm in order to adapt it to S-HMMs. The algorithm uses two functions, ξ and γ , defined through α and β . Function $\xi_t(i, j)$ computes the probability of the transition between states q_i (at time t) and q_j (at time $t + 1$), assuming that the observation O has been generated by model λ :

$$\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} \quad (3)$$

Function $\gamma_t(i)$ computes the probability of being in state q_i at time t , assuming that the observation O has been generated by model λ , and can be written as:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \quad (4)$$

The sum of $\xi_t(i, j)$ over t estimates the number of times transition $q_i \rightarrow q_j$ occurs when λ generates the sequence O . In an analogous way, by summing $\gamma_t(i)$ over

t, an estimate of the number of times state q_i has been visited is obtained. Then a_{ij} can be re-estimated (a-posteriori, after seeing O) as the ratio of the sum over time of $\xi_t(i, j)$ and $\gamma_t(i)$:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)} \quad (5)$$

With a similar reasoning it is possible to obtain an a-posteriori estimate of the probability of observing $o = v_k$ when the model is in state q_j . The estimate is provided by the ratio between the number of times state q_j has been visited and symbol v_k has been observed, and the total number of times q_j has been visited:

$$\bar{b}_j(k) = \frac{\sum_{\substack{t=1 \\ i=1 \text{ or } o_t=v_k}^{T-1} \alpha_t(j) \beta_t(j)}{\sum_{t=1}^{T-1} \alpha_t(j) \beta_t(j)} \quad (6)$$

From (1) and (2) it appears that, inside basic block λ_k , equations (5) and (6) are immediately applicable. Then the Baum-Welch algorithm can be used without any change to learn the probability distributions inside basic blocks.

On the contrary, equation (5) must be modified in order to adapt it to re-estimate transition probabilities between output and input states of the blocks, which are silent states. As there is no emission, α and β propagate through transitions without time change; then, equation (5) must be modified as in the following:

$$\bar{a}_{E_i I_j} = \frac{\sum_{t=1}^{T-1} \alpha_t(E_i) a_{E_i I_j} \beta_t(I_j)}{\sum_{t=1}^{T-1} \alpha_t(E_i) \beta_t(I_j)} \quad (7)$$

It is worth noticing that functions α and β depend upon the states in other blocks only through the value of $\alpha_t(I_k)$ and $\beta_t(E_k)$, respectively. This means that, in block λ_k , given the vectors $\alpha_1(I_k), \alpha_2(I_k), \dots, \alpha_T(I_k)$ and $\beta_1(E_k), \beta_2(E_k), \dots, \beta_T(E_k)$, Baum-Welch algorithm can be iterated inside a block without the need of re-computing α and β in the external blocks. We will call this a *locality property*. The practical implication of the locality property is that a block can be modified and trained without any impact on the other components of an S-HMM.

4 Applying S-HMM to Real World Tasks

Most HMM applications can be reduced to classification (instances of *Problem 1*) or interpretation (instances of *Problem 2*) tasks. Word recognition and user/process profiling are typical classification tasks. Sequence tagging and knowledge extraction, as done in DNA analysis, are typical interpretation tasks. In this section we will focus on the problem of knowledge extraction, but most of the proposed solutions also hold for classification tasks.

A model for interpreting a sequence is a *global* model, able to identify interesting patterns (*i.e.*, *motifs*, adopting Bio-Informatics terminology), which occur with significant regularity, as well as *gaps*, *i.e.*, regions where no regularities are

found. Having a global model of the sequence is important, because it allows inter-dependencies among motifs to be detected. Nevertheless, a global model must account for the distribution of the observations on the entire sequence, and hence it could become intractable. We tame this problem by introducing special basic blocks, designed to keep low the complexity of modeling the irrelevant parts of sequences.

In the following we address the following issues: (a) how to construct basic blocks modeling motifs; (b) how to construct models of gaps between motifs; (c) how to segment a sequence detecting the instances of the basic blocks; (d) how to extract a readable interpretation from a sequence.

4.1 Motif Modeling

A motif is a subsequence frequently occurring in a reference sequence set. Motif occurrences may be different from one to another, provided that an assigned equivalence relation be satisfied. In the specific case, the equivalence relation is encoded through a basic block of an S-HMM. Many proposals exist for HMM architectures oriented to capture specific patterns. Here, we will consider the Profile HMM (PHMM), a model developed in Bio-Informatics [4], which well fits the needs of providing a readable interpretation of a sequence. The basic assumption underlying PHMM is that the different instances of a motif originate from a canonical form, but are subject to insertion, deletion and substitution errors.

As described in Figure 2, a PHMM has a left-to-right structure with a very restricted number of arcs. Moreover, it makes use of *typed* states: *Match* states, where the observation corresponds to the expectation, *Delete* states (silent states) modeling deletion errors, and *Insert* states modeling insertion errors supposedly due to random noise. According to this assumption, the observation distribution in all insert states is the same, and it can be estimated just once.

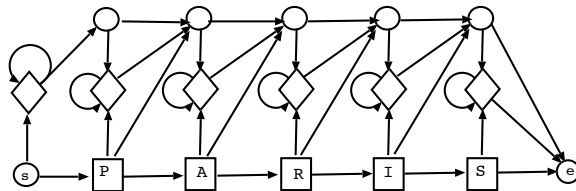


Fig. 2. Example of Profile Hidden Markov Model. Circles denote states with *no-observable* emission, rectangles denote *match states*, and diamond denote *insert states*.

After training, the canonical form can be easily extracted from a PHMM, by collecting the maximum likelihood observation sequence from the match states.

4.2 Modeling Duration and Gaps

The problem of modeling durations arises when the time span covered by an observation or the interval length between two observations is important. In the HMM framework, this problem has been principally faced in Speech Recognition and in Bio-informatics. However, the problem setting is slightly different in the two fields, and consequently the dominant approach tends to be different. In speech recognition, the input is a continuous signal, which, after several steps of signal processing, is segmented into variable length intervals each one labeled with a symbol. Then, the obtained symbolic sequence is fed into a set of HMMs, which accomplish the recognition of long range structure, such as syllables or words, requiring thus to deal with interval durations [11]. In Bio-informatics the major application for HMMs is the analysis of DNA strands [4]. Here, the input sequence is a string of equal length symbols. The need of modeling duration comes from the presence of gaps, *i.e.*, substrings where no coding information is present. The gap duration is often a critical cues to interpret the entire sequence.

The approach first developed in Speech Recognition is to use Hidden Semi-Markov Models (HSMM), which are HMMs augmented with probability distributions over the state permanence [11, 10, 13, 17, 16]. An alternative approach is the so called *Expanded HMM* [10]. Every state, where it is required to model duration, is expanded into a network of states, properly interconnected. In this way, the duration of the permanence in the original state is modeled by a sequence of transitions through the new state network in which the observation remain constant. The advantage of this method is that the markovian nature of the HMM is preserved. Nevertheless, the complexity increases according to the number of new states generated by expansion.

A similar solution is found in Bio-Informatics for modeling long gaps. In this case, the granularity of the sequence is given, and so there is no expansion. However, the resulting model of the gap duration is similar to the one mentioned above. A Profile HMM [4], naturally models the duration of observations according to the expansion technique, but it is only able to model short gaps inside a motif, attributed to random noise controlled by a Poisson statistics. Nevertheless, single insertion states do not correctly model long gaps occurring in between two motifs. The most appropriate probability distribution for this kind of gaps may vary from case to case, but it is never the exponential decay defined by an insert state with a self-loop.

Two HMM topologies, suitable for modeling gaps, are reported in Figure 3. The architecture in Figure 3(a) can be used to model any duration distribution over a finite and discrete interval. However, the drawback of this model is the potentially large number of parameters to estimate. In all cases, the observation is supposed to be produced by random noise. Model in Figure 3(b) exhibits an Erlang's distribution, when the Forward-Backward algorithm is used. Unfortunately, the distribution of the most likely duration computed by Viterbi algorithm still follows an exponential law. Therefore, this model, which is more compact with respect to the previous one, is not useful for the task of segmenting and tagging sequences by means of Viterbi algorithm.

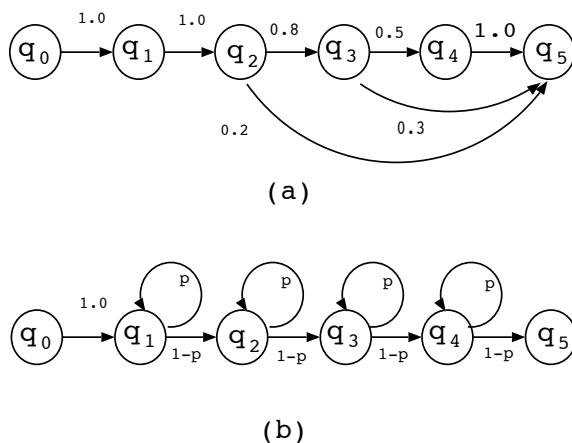


Fig. 3. Possible HMMs for modeling duration

4.3 Sequence Segmentation

In the S-HMM framework, sequence segmentation means detecting where boundaries between blocks are most likely located. Segmentation provides a probabilistic interpretation of a sequence model, and plays a fundamental role when an S-HMM is used for knowledge extraction.

Two methods exist for accomplishing this task. The classical one is based on Viterbi algorithm in order to find the most likely path in the state space of the model. Then, for every pair of blocks λ_r , λ_s on the path, the most likely time instant for the transition from the output state of λ_r to the input state of λ_s is chosen as the boundary between the two blocks. In this way a unique, non ambiguous segmentation is obtained, with a complexity which is the same as for computing α and β .

The second method, also described in [14], consists in finding the maximum likelihood time for the transition from λ_r to λ_s by computing:

$$\tau_{rs} = \underset{t}{\operatorname{argmax}} \left(\frac{\xi_t(E_r, I_s)}{\gamma_t(E_r)} \right) \quad (8)$$

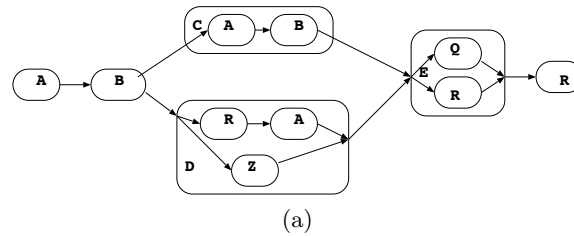
Computing boundaries by means of (8) requires a complexity $O(T)$ for every boundary that has to be located, in addition to the complexity for computing one α and β .

The advantage of this method is that it can provide alternative segmentations by considering also blocks that do not lie on the maximum likelihood path. Moreover, it is compatible with the use of gap models of the type described in Figure 3(c), because it does not use Viterbi algorithm.

4.4 Knowledge Transfer

When a tool is used in a knowledge extraction task, two important features are desirable: (a) the extracted knowledge should be readable for a human user; (b) a human user should be able to elicit chunks of knowledge, which the tool will exploit during the extraction process.

The basic HMM does not have such properties, whereas task oriented HMMs, such as Profile HMM, may provide such properties to a limited extent. On the contrary, the S-HMM structure naturally supports high level logical descriptions. An example of how an S-HMM can be described in symbolic form is provided



(b) **Basic block description:**
 $motif(x) \wedge MLS(x, "ctgaac") \wedge AvDev(x, 0.15) \rightarrow A(x)$
 $motif(x) \wedge MLS(x, "cctctaaa") \wedge AvDev(x, 0.15) \rightarrow R(x)$
 $motif(x) \wedge MLS(x, "tatacgc") \wedge AvDev(x, 0.15) \rightarrow Q(x)$
 $gap(x) \wedge AvDr(x, 11.3) \wedge MnDr(x, 8) \wedge MxDr(x, 14) \rightarrow B(x)$
 $gap(x) \wedge AvDr(x, 15.6) \wedge MnDr(x, 12) \wedge MxDr(x, 19) \rightarrow Z(x)$

(c) **Block structure logical description:**

$$\begin{aligned}
 &A(x) \wedge B(y) \wedge follow(x, y) \rightarrow C([x, y]) \\
 &R(x) \wedge A(y) \wedge follow(x, y) \rightarrow D([x, y]) \\
 &Z(x) \rightarrow D(x), \quad Q(x) \rightarrow E(x), \quad R(x) \rightarrow E(x) \\
 &B(x) \wedge C(y) \wedge follow(x, y) \rightarrow G([x, y]) \\
 &B(x) \wedge D(y) \wedge follow(x, y) \rightarrow G([x, y]) \\
 &A(x) \wedge G(y) \wedge E(z) \wedge R(w) \wedge follow(x, y) \wedge \\
 &\quad \wedge follow(y, z) \wedge follow(z, w) \rightarrow MySEQ([x, y, z, w])
 \end{aligned}$$

(d) **Block structure as a regular expression:**

$$A (B (AB \mid (RA \mid Z))) (Q \mid R) R$$

Fig. 4. Structured HMMs are easy to translate into an approximate logic description.

in Figure 4. Basic blocks and composite blocks must be described in different ways. Basic blocks are either HMMs (modeling subsequences), or gap models. In both cases, a precise description of the underlying automaton will be complex, without providing readable information to the user. Instead, an approximate description, characterizing at an abstract level the knowledge captured by a block, is more useful. For instance, blocks corresponding to regularities like *motifs* can be characterized by providing the maximum likelihood sequence (MLS) as the nominal form of the sequence, and the average deviation (AvDv) from the nominal form. Instead, gaps can be characterized by supplying the average duration (AvDr), and the minimum (MnDr) and maximum (MxDr) duration.

On the contrary, the model's composite structure is easy to describe by means of a logic language. As an example, Figure 4(c) provides the translation into Horn

clauses, whereas Figure 4(d) provides the translation into regular expressions. In both cases, richer representations can be obtained by annotating the expressions with numeric attributes.

By using a logic description language, or regular expressions, a user can also provide the specification of an S-HMM structure, or part of it, which will be completed and trained by a learning algorithm. Logic formulas as in Figure 4(c) can be immediately translated into the structure of composite blocks. Nevertheless, also an approximate specification for basic blocks, as described in Figure 4, can be mapped to block models when the model scheme is given. For instance, suppose that motifs are described by Profile HMMs, and gaps by the scheme of Figure 3(a) or (b). Then, the maximum likelihood sequence provided in the logic description implicitly sets the number of match states and, together with the average deviation, provides the prior for an initial distribution on the observations. In an analogous way, minimum, maximum and average values specified for the gap duration can be used to set the number of states and a prior on the initial probability distribution. Then, a training algorithm can tune the model parameters.

5 Conclusive Remarks

In the previous sections we have formally defined an S-HMM, showing how it can be used to model complex patterns in symbolic sequences. S-HMMs are an attempt to unify in a formal framework previous attempts, which we briefly review in this section.

First of all it is worth noticing that S-HMMs derive from Hierarchical HMMs (HHMM)[5], in that both construct a hierarchy of HMMs. The difference is that a block in an S-HMM can be reached from only one state in a block at higher level, whereas in a generalized HHMM, many ancestors may exist. For this reason, an HHMM is not easy to handle for the purposes addressed in this paper; in fact, re-structuring or compiling an HHMM into a single level HMM poses non trivial problems. S-HMMs address this point by means of silent states and of further constraints on the structure. In this way, the same structure can be considered at the desired abstraction level without needing any reformulation of the HMMs. The most important property of an S-HMM is *locality*, which can be exploited for incrementally constructing very complex models.

A similar proposal was put forward by Bouchaffra and Tan [3], who proposed Structural HMMs (SHMM in the following). The initial goal of SHMMs was analogous to the one that inspired S-HMM, *i.e.*, to capture in an HMM local regularities in the observation sequences. However, SHMMs have been developed primarily for segmentation problems, setting the emphasis on data structure, while the structure of the model remains somehow implicit. S-HMMs explicitly put the emphasis on the model structure and on its compositional properties. Then, quite different formalizations have been obtained. Nevertheless, the relations between the two models deserve further investigations.

Another relevant point concerns modeling duration; many authors tend to handle this aspect by using continuous time distributions on the state permanence. We only considered modeling duration through the expanded state approach, which is made possible because of the low complexity inherent to an S-HMM. Modeling durations with continuous distributions over the time does not seem the right way to go, because it will dramatically increase the complexity, being then not suitable to exploit the S-HMM structure.

References

1. T. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, Menlo Park, California, 1994. AAAI Press.
2. L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximisation techniques occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.
3. D. Bouchaffra and J. Tan. Structural hidden markov models using a relation of equivalence: Application to automotive designs. *Data Mining and Knowledge Discovery*, 12:7996, 2006.
4. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.
5. S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32:41–62, 1998.
6. G. D. Forney. The viterbi algorithm. *Proceedings of IEEE*, 61:268–278, 1973.
7. U. Galassi, A. Giordana, and L. Saitta. Incremental construction of structured hidden markov models. In *proceedings IJCAI-2007*, pages 2222–2227, 2007.
8. Z. Ghahramani and M. Jordan. Factorial hidden markov models. *Machine Learning*, 2:1–31, 1997.
9. N. W. Grundy, T. Bailey, C. Elkan, and M. Baker. Meta-meme: Motif-based hidden markov models of biological sequences. *Computer Applications in the Biosciences*, 13(4):397–406, 1997.
10. A. B. Josep. Duration modeling with expanded hmm applied to speech recognition.
11. S. Levinson. Continuous variable duration hidden markov models for automatic speech recognition. *Computer Speech and Language*, 1:29 – 45, 1986.
12. K. Murphy and M. Paskin. Linear time inference in hierarchical hmms. In *Advances in Neural Information Processing Systems (NIPS-01)*, volume 14, 2001.
13. J. Pyllkknen and M. Kurimo. Using phone durations in finnish large vocabulary continuous speech recognition, 2004.
14. L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.
15. L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NY, 1993.
16. D. Tweed, R. Fisher, J. Bins, and T. List. Efficient hidden semi-markov model inference for structured video sequences. In *Proc. 2nd Joint IEEE Int. Workshop on VSPETS*, pages 247–254, Beijing, China, 2005.
17. S.-Z. Yu and H. Kobashi. An efficient forward-backward algorithm for an explicit duration hidden markov model. *IEEE Signal Processing Letters*, 10(1), 2003.