Towards Discovering Structural Signatures of Protein Folds based on Logical Hidden Markov Models

Kristian Kersting [a] Tapani Raiko [b] Stefan Kramer [a] Luc De Raedt [a]

[a] *Machine Learning Lab, Computer Science Department*
University of Freiburg
Georges-Koehler-Alle 079
79112 Freiburg, Germany
[b] *Laboratory of Computer and Information Science*
Helsinki University of Technology
P.O. Box 540
Helsinki, FIN-02015 HUT, Finland

With the growing number of determined protein structures and the availability of classification schemes, it becomes increasingly important to develop computer methods that automatically extract structural signatures for classes of proteins. In this paper, we apply a new Machine Learning technique, Logical Hidden Markov Models (LOHMMs), to the task of finding structural signatures of folds according to the classification scheme SCOP. Our preliminary results indicate that LOHHMs are in fact applicable to this task: The number of parameters of our LOHMM is by an order of magnitude smaller than the number of an equivalent HMM, and the generalization performance appears satisfactory. Furthermore, we show that it is easy to extract characteristic patterns from the trained LOHMM. Summing up, we believe that LOHMMs show a great promise for the analysis of such data and for the discovery of structural signatures of protein folds.
Keywords: Hidden Markov Models, Probabilistic Modeling, First-Order Logic, Structural Signatures, SCOP

1. Introduction

In recent years, the number of proteins with determined structure has been growing rapidly due to large-scale structural genomics projects. Consequently, the Protein Data Bank (PDB) is growing at enormous rates. In parallel, researchers have developed classification schemes of proteins based on their sequence, structure and function. The development of classification schemes is a common scientific activity to make sense and gain a deeper understanding of experimental data. Given the determined structures and classification schemes, the discovery of structural characteristics of protein classes becomes an important topic. The primary interest there is to gain insights into structural characteristics of fold classes, but ultimately structural signatures should also be useful for the prediction of protein folds. In fact, human experts performing well in the CASP2 predictive exercise made use of knowledge about structural signatures. So far, most signatures have been discovered by human experts based on extensive manual/visual inspection of the data. However, few experts in the world are in a position to find/provide these signatures, and few systematic attempts exist to catalog known signatures. So, there is a need to develop computer methods that automatically extract structural signatures in a systematic way [TMS01a].

One of the manually curated classification schemes, SCOP, forms the basis of the data material used in this work. The SCOP scheme is due to Alexey G. Murzin, and the website and database is maintained by the MRC Laboratory of Molecular Biology at Cambridge, UK. SCOP defines a multi-layer classification hierarchy of domains (i.e., structures/substructures that are assumed to be folded independently) based on their sequences, structures and functions. At the top level of SCOP, domains are grouped, mostly by secondary structure characteristics, into all-alpha, all-beta, various types of mixed alpha-beta structures and others. In this paper, we focus on mixed alpha-beta structures, because they are structurally among the most complex and diverse. At the next level of SCOP, we have a classification into so-called folds, which has been created considering both the topology and the similarity with domains of known function. In this paper, we focused on one interesting SCOP class (alpha/beta), and then picked the five most populated folds within this class. The task was then to find structural signatures for each of these five folds.

Recently, Hidden Markov Models (HMM) have been used to analyze classes in SCOP [GKHC01]. HMMs are among the most widely and successfully used tools for the analysis of sequence data in bioinformatics. Despite their successes, however, it is well-known that HMMs have a number of weaknesses. One of the major weaknesses is that HMMs handle only sequences of unstructured symbols. They are not capable of handling sequences where the symbols of the output alphabet are structured. In this paper we will overcome this weakness by introducing Logical Hidden Markov Models (LOHMMs). LOHMMs process sequences of logical atoms (hence the term ”logical”). Logical atoms are the primary syntactic component of the predicate calculus. E.g. the atom $he(h(left, alpha), 9)$ denotes a left-handed alpha helix of length 9. In our framework such logical atoms are used as symbols in the alphabet of a LOHMM and also to denote the states. Furthermore, as usual in the predicate calculus, we allow for the use of variables as well as unification. They allow us to make abstraction of specific symbols and states. E.g. the logical atom $he(X, 9)$ would represent the set of helices of any type (X which represents a logical variable) and of length 9. In principle, each LOHMM can be translated back into a regular HMM (by instantiating all its variables in all possible ways), but the size of the resulting HMM is an order of magnitude larger than the size of the original LOHMM. Due to the compact, abstract representation of LOHMMs, these models are much easier to understand, adapt and refine than the traditional HMM. Another feature particularly appealing for applications in molecular biology is that structured data can be handled elegantly.

For instance, in the application presented in this paper, proteins could be analyzed up to a level of detail, where the assignment of strands to sheets, and helices of different types, but identical orientations, are taken into account.

The aim of this paper is to put the following hypotheses to the test:

- H1) LOHMMs are capable of distinguishing between different folds based on a logical representation of the secondary structure of domains.
- H2) The inspection of LOHMMs reveal distinguishing features of folds.
- H3) LOHMMs can be applied to real-world problems as hard as signature discovery.
- H4) In some applications in computational biology, LOHMMs are by at least an order of magnitude smaller than their instantiations which are HMMs.

This paper is organized as follows. In Section 2, we present the task and the dataset. After briefly reviewing hidden Markov models in Section 3, we introduce logical hidden Markov models. Section 4 describes our preliminary experiments with LOHMMs for the discovery of structural signatures. Subsequently, we discuss related work and conclude.

2. Task and Dataset

In this section, we describe the task of finding structural signatures of protein folds and the dataset used. The basis of our study is the SCOP (Structural Classification of Proteins) database due to A. Murzin and maintained by the MRC Laboratory of Molecular Biology. Our goal was to find structural characteristics of the domains at the second level of the SCOP hierarchy, i.e., the level of folds. In our study, we focused on alpha and beta proteins (a/b), a class consisting of domains with mainly parallel beta sheets (beta-alpha-beta units). From this class, we chose the five most populated subclasses, that is, folds: TIM beta/alpha-barrel, NAD(P)-binding Rossmann-fold domains, Ribosomal protein L4, glucosamine 6-phosphate deaminase/isomerase and and leucine aminopeptidas. The overall set-up is quite similar to the one in [TMS01b,TMS01a]. The data have been extracted automatically from the PDB release #96 and SCOP version 1.57. Known redundancies were kept in the dataset, because we did not want to change the distribution for the subsequent modeling of the data using HMMs. From the extraction process, we obtained 720 examples for TIM beta/alpha-barrel, 359 examples for NAD(P)-binding Rossmann-fold domain, 274 examples for Ribosomal protein L4, 441 examples for glucosamine 6-phosphate deaminase/isomerase and 290 examples for leucine aminopeptidas.

Information for domains from the above five folds was extracted in the form of “logical sequences” of secondary structure elements. Logical sequences are sequences of logical atoms (see the Appendix on logic for the terminology). An example of such a sequence (corresponding to a Ribosomal protein L4) is:

$$st(null, 2).he(h(right, alpha), 6).$$
$$st(plus, 2).he(h(right, alpha), 4).$$
$$st(plus, 2).he(h(right, alpha), 4).$$
$$st(plus, 3).he(h(right, alpha), 4).$$
$$st(plus, 1).he(h(right, alpha), 6).$$

There are two predicates in the atoms,i.e.

$$he(HelixType, Length) \text{ and}$$
$$st(Orientation, Length).$$

The predicate $he(HelixType, Length)$ models helices of a certain type and length, whereas $st(Orientation, Length)$ is the predicate for strands of a certain orientation and length. The helix types are:

$$h(left, alpha), h(right, alpha),$$
$$h(left, gamma), h(right, gamma),$$
$$h(left, omega), h(right, omega), h(right, pi),$$
$$h(right, 3to10), 27ribbon$$
$$\text{and } polyproline.$$

Note that, logically speaking, the helix types are represented by structured terms. The orientation of strands can be null (the beginning of a sheet), $plus$ (a parallel strand of a sheet), or $minus$ (an anti-parallel strand of a sheet). The length is defined as the number of (physical) atoms and discretized in the experiments (see below).

Important information other than the secondary structure was not considered in the current representation of the data for the sake of simplicity. It may well be worth investigating more complex representations (e.g., including coils and particular groups within proteins) in the future.

We also extracted more complex logical sequences, where the chain ID and the assignment of strands to different sheets were represented as well. The underlying idea is that one could give a unique identifier to each of the chains and sheets in a protein and then could then add one argument to each of the predicates that would model the identifier. Using this representation, one can model complex dependencies such as

$$st(s1, null, 9).st(s1, plus, 8).$$
$$he(c1, h(left, alpha), 12).st(s1, minus, 10).$$

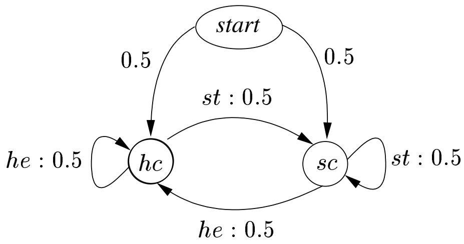


Figure 1. A simple hidden Markov model (HMM) of protein secondary structures. The states hc and sc correspond to consecutive sequences of helices and strands. Edges indicate transitions from one state to another state emitting an helix he or a strand st. An artifical node *start* models the prior probability over states.

This sequence explicitly represents that all three strands belong to the same sheet named $s1$. However, at the point of submission to ECCB, we have not yet finalized our experiments with this enhanced representation.

For each of the above five folds, we are modelling their domains in terms of their secondary structure using a logical variant of HMMs. So, what can be expected from an application of LOHMMs to this task? First of all, it should be clear that we do not obtain structural signatures for each of these classes immediately. What we obtain instead is a model for each fold. Each model provides a precise probabilistic and logical characterization of the respective fold. Structural signatures can then be found by a comparison of models. As will be shown below, it is quite easy to find characteristics upon inspection of the trained models.

3. Hidden Markov Models

Hidden Markov models [RJ86,Rab89] are Markov chains, where each state generates an observation. In our task, helices and strands are the observations. A block of consecutive helices (respectively strands) acts as a kind of hidden state. We will sometimes refer to this blocks as complexes. An oversimplified hidden Markov model explaining the overall sequence of helices and strands is given in Figure 1. This model just serves as an illustration of the concept of HMMs. It further will allow us to motivate the introduction of LOHMMs.

Transitions such as $0.5 : hc \xleftarrow{he} sc$ specify that the model makes a transition from state sc, i.e. a block of consecutive strands to state hc, i.e. a block of consecutive helices with probability 0.5 emitting an helix he. The model generates an output string such as $st, st, he, \ldots$ over a finite alphabet $\{he, st\}$ by choosing an initial state say sc according to a prior distribution π over the states, following a transition to another state, say sc, according to a certain probability, and emitting an output symbol st, following a transition to another state, say hc, emitting st, and so on. The parameters λ of the model are the prior distribution π and the transition probabilities. It is common to specify the prior distributions π over states by introducing an artificial state

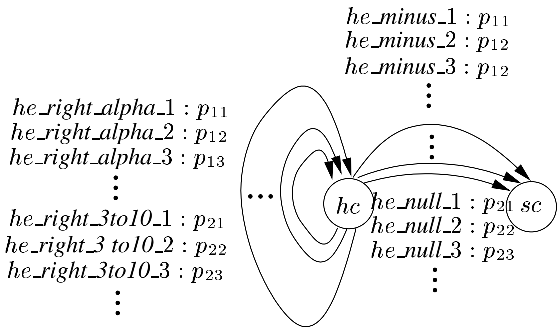


Figure 2. The outgoing arcs of the state hc taking all helix types and lengths into account are drawn. For the sake of simplicity, the full HMM is not drawn.

start, which becomes the only starting state and whose outgoing transitions encode π. We must have that the probability values associated to all outgoing edges of any single state sum to 1.0. The model given in Figure 1 is overly simple and unrealistic because it neglects the observed length, type and orientation of helices and strands, as well as the total length of the observed sequences. If we would like to model these as well, we would need to modify and extend the entire Markov model as follows. Let n_T, n_{LH}, n_{LS} and n_O be the numbers of different helix types, helix lengths, strand lengths, and strand orientations. Assuming $n_T = 10, n_{LH} = n_{LS} = 16$ and $n_O = 3$, one would need to specify 7680 parameters, and the model would become quite confusing as Figure 2 illustrates. Furthermore, the refined model does not yet consider the total length of the observed sequence, which would be important to model drifts in transition probabilities along the position in the secondary structure.

This simple example illustrates the difficulties encountered when using traditional HMMs to model problems where the observed sequences involve structural properties (such as length, type, etc). The number of states and probabilities to specify rapidly explodes and the whole model becomes unmanageable. The reason for this is that the state and emission symbols of traditional HMMs are unstructured. Within this paper, we will show how logical atoms provide us with the relevant structure to alleviate the above mentioned problems within the framework of LOHMMs.

4. Logical Markov Models

Logical (hidden) Markov models (LOHMM) extend the unstructured model representation of HMMs by incorporating complex, internal structure into the specification of transitions (and therefore of emissions). The central concept is the one of *abstract transitions* (cf. below). Abstract transitions specify general regularities among abstract states, which represent sets of states.

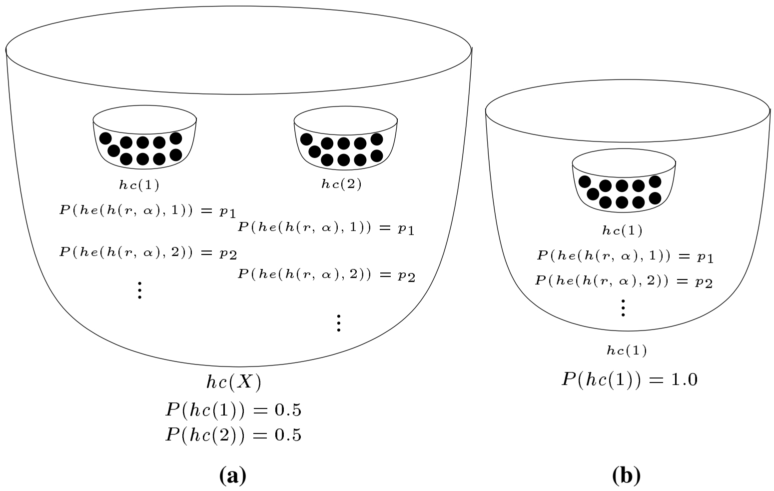


Figure 3. An urn and ball model which illustrates the general idea of a discrete symbol logical (hidden) Markov models. (a) The *abstract state* $hc(X)$ is an urn with two states as balls. In turn, each state is an urn with the emissions $he(h(right, alpha), 1), he(h(right, alpha), 2) \ldots$ as balls (abbreviated by $he(h(r,\alpha), 1)$ and $he(h(r,\alpha), 2)$ in the figure). (b) The state $hc(1)$ interpreted as abstract state.

4.1. Basic Idea of LOHMMs

To explain *logical (hidden) Markov models*, we employ an interpretation of hidden Markov models given by [RJ86]. It is depicted in Figure 3: States correspond to urns each consisting of a number of distinct balls representing the output symbols (in our case helices and strands) . A "helices and strands tossing" experiment consists of drawing balls from these urns in some sequence, and only the sequence of balls drawn is shown to us.

In logical HMMs we summarize sets of states by *abstract states*, which are represented by logical atoms. A logical atom then represents all states that can be obtained by instantiating the atoms (i.e. by replacing the variables by terms). E.g. the abstract state $hc(X)$, where X is a variable, could represent the set of states $\{hc(1), hc(2)\}$ depending on the terms (1 and 2 in this case) in the LOHMM. If the logical atom does not contain any variable (such as in $hc(1)$), it is an ordinary state representing a singleton set.

Now, in analogy to Rabiner's interpretation, abstract states correspond to "urns of urns" (cf. Figure 3.a). In a first step, an "urn" (i.e. a state) is sampled from the encompassing urn (i.e. the abstract state). Subsequently, a ball is drawn from the sampled urn (i.e. a symbol is generated from the state). This mechanism also works for ordinary states, as illustrated in Figure 3.b. In this case, the encompassing urn (or the abstract state) contains exactly one urn, which is always selected (cf. Figure 3.b).

Abstract states are connected by abstract transitions, which summarize sets of transitions

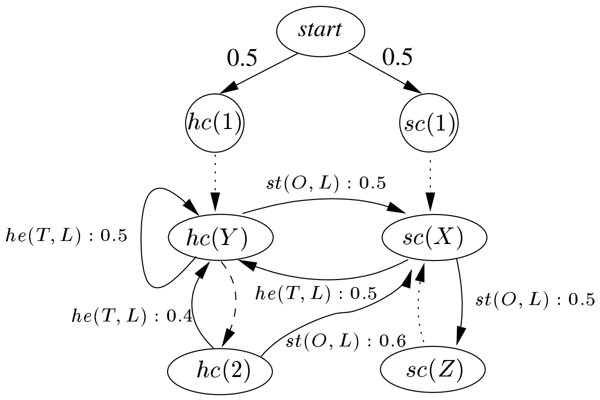


Figure 4. A first-order logical hidden Markov model encoding default reasoning. Dashed edges represent the *more general than* relation. A state is represented by the most specific node unifying with the state.

between states. Together, the abstract states and abstract transitions of a logical (hidden) Markov model provide a compact representation of an equivalent HMM.

4.2. An Example of a LOHMM

Figure 4 shows an example of a LOHMM. The vertices in the model represent abstract states. In such models, we find three different types of edges:

- *Solid edges* between abstract states specify the abstract transitions. Transition probabilities and emission symbols are associated to them. An example transition from Figure 4 is $sc(X) \xleftarrow{st(O,L):0.5} hc(Y)$. Such a solid edge expresses that if one is in one of the states represented by $hc(Y)$ one will go to one of the states in $sc(X)$ with probability 0.5 while emitting the (abstract) symbol $st(O, L)$.
- *Dotted edges* indicate that two abstract states behave in exactly the same way. If we follow a transition to an abstract state with an outgoing dotted edge, we will automatically follow that edge. An example dotted edge in Figure 4 goes from $sc(Z)$ to $sc(X)$. It represents the fact that the two abstract states are identical. The dotted edge is needed in this case because the variables appearing in the abstract states are different. We could not have written this using solid edges alone as the meaning of the solid edge $sc(X) \xleftarrow{st(O,L):0.5} sc(X)$ is different from that of $sc(Z) \xleftarrow{st(O,L):0.5} sc(X)$ (cf. below). In a logical sense, dotted edges correspond to some form of recursion.
- *Dashed edges* represent a kind of default reasoning. This is often used to model exceptions. Consider the dashed edge in Figure 4 connecting $hc(Y)$ and $hc(2)$. This dashed edge denotes that $hc(2)$ is a more specific state than $hc(Y)$. This implies that the set of states represented by the more specific (abstract) state is a subset of that represented by the more general one.

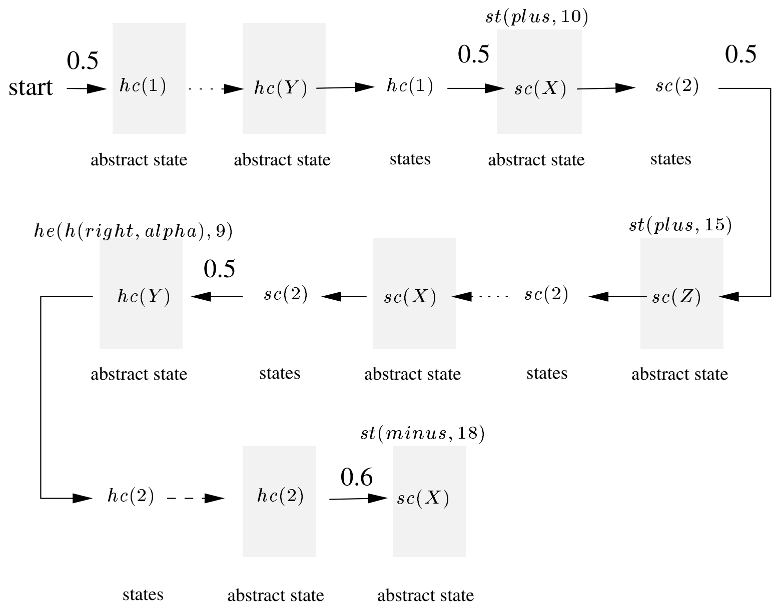


Figure 5. Generating the observation sequence $st(plus, 10)$, $st(plus, 15)$, $he(h(right, alpha), 9)$, $st(minus, 18)$ by the LOHMM in Figure 4.

Logically speaking, the more specific state $hc(2)$ can be obtained by substituting Y by 2 in the more general state $hc(Y)$. Dashed edges and the default reasoning are useful because they represent exceptions. Indeed, in our current example, the outgoing probability labels associated to $hc(2)$ are different from those for $hc(Y)$. This actually implies that the $hc(2)$ acts as an exception to the states represented by $hc(Y)$. So for $Y = 2$ we employ the transitions from $hc(2)$ and for $Y \neq 2$ we follow those indicated by $hc(Y)$.

Before formalizing first order (hidden) Markov models in the next subsection, let us explain how the model in Figure 4 generates the sequence of observations $st(plus, 10), st(plus, 15), he(h(right, alpha), 9)$, $st(minus, 18)$ (cf. Figure 5). It chooses an initial abstract state, say $hc(1)$. Forced to follow the dotted edge, it enters the abstract state $hc(Y)$. In each abstract state, the model samples values for all variables that are not instantiated yet. Since the value of Y was already instantiated in the previous abstract state $hc(1)$, the model samples with probability 1.0 the state $hc(1)$. Now, the model goes over to the abstract state $sc(X)$, emitting $st(plus, 10)$ which in turn was sampled from the "urn of urns" $st(O, L)$. Variable X in $sc(X)$ is not yet bound; so, a value, say 2, is sampled from a given selection distribution (see below). Next, we move on to abstract state $sc(Z)$, emitting $st(plus, 15)$. The variable Z is sampled to be 3. The dotted edge brings us back to $sc(X)$ and automatically unifies X with Z, which is bound to 3. Emitting $he(h(right, alpha), 9)$, the model returns to abstract state $hc(Y)$. Assume that it samples 2 for variable Y, it has to follow the dashed outgoing edge to $hc(2)$, which represents an exception to $hc(Y)$. Going over to $sc(X)$, it emits $st(minus, 18)$.

This process is similar to unrolling recurrent neural networks [DK88] or dynamic Bayesian networks [WZ95], or to grounding clause programs [Llo89].

4.3. Logical Model Description

A *logical (hidden) Markov model* (LOHMM) is a finite set $\mathcal{T}$ of abstract transitions. An *abstract transition at* is an expression of the form

$$h(t'_1, t'_2, \ldots, t'_l) \xleftarrow{o(u_1,u_2,\ldots,u_m):p} b(t_1, t_2, \ldots, t_n).$$

Here, $h(t'_1, t'_2, \ldots, t'_l)$ and $b(t_1, t_2, \ldots, t_n)$ are abstract states. We will denote the set of abstract states occurring in $\mathcal{T}$ by $\mathcal{A}$. From a logical point of view, *at* can be considered a clause augmented with an abstract emission symbol $\mathrm{out}(at) = o(u_1, u_2, \ldots, u_m)$ and a probability value $a(at) = p$ is associated. For those familiar with computational logic, all main logical concepts such as head, body, Herbrand base etc. carry over in a natural way to LOHMMs. The meaning of *at* is:

Let θ be a substitution[1] grounding $b(t_1, \ldots, t_n)$, θ' a substitution grounding $h(t'_1, \ldots, t'_l)\theta$, and θ'' a substitution grounding $o(u_1, \ldots, u_m)\theta\theta'$. The model makes a transition from state $b(t_1, \ldots, t_n)\theta$ to $h(t'_1, \ldots, t'_l)\theta\theta'$ with probability

$$\begin{aligned} p \cdot \mu(h(t'_1, \ldots, t'_l)\theta\theta' \mid h(t'_1, \ldots, t'_l)\theta) \\ \cdot \mu(o(u_1, \ldots, u_m)\theta\theta'\theta'' \mid o(u_1, \ldots, u_m)\theta\theta') \end{aligned}$$

emitting the symbol $o(u_1, \ldots, u_m)\theta\theta'\theta''$.

To explain this definition, let us first consider the case of a traditional HMM, i.e. a clause $h \xleftarrow{o:p} b$. Then the clause simply specifies that if one is in state b there is a transition to state h emitting the symbol o with probability $p \cdot \mu(h \mid h) \cdot \mu(o \mid o)$. Now, $\mu(h\theta \mid h)$ is the probability of going to the ordinary state (ground atom) $h\theta$ starting from the abstract state (not-necessarily ground atom). μ is not only applied to abstract states but also to abstract emission symbols. In the case of a ground clause or a clause in a traditional HMM, $\mu(o \mid o) = \mu(h \mid h) = 1$. So, in the ground case, the above definition implements the traditional semantics in HMMs.

Now, consider the case that we have a non ground clause such as $sc(X) \xleftarrow{st(O,L):0.5} hc(Z)$ and that we are in state $hc(1)$. To follow this abstract transition, we need first to find the substitution $\theta = \{Z = 1\}$ so that $hc(1) = hc(Z)\theta$. Applying this substitution to the iterative clause yields $sc(X) \xleftarrow{st(O,L):0.5} hc(1)$. It specifies that with probability 0.5 we go to the abstract state $sc(X)$. We then need to proceed to the next ordinary state, say $sc(3)$. The probability that we go

[1] Formally speaking, a substitution θ is an assignment of terms to variables, e.g. $\{X = 2, Y = t(Z)\}$ where X and Y are variables, and 2 and $t(Z)$ terms. One applies a substitution θ to an expression e yielding the expression $e\theta$ by simultaneously replacing each occurrence of a variable by its corresponding term. A substitution θ grounds an expression e if $e\theta$ does not contain any variables. E.g. the above substitution would ground $h(X)$.

from the abstract state $sc(X)$ to the ordinary state $sc(3)$ is given by $\mu(sc(X)\theta' = sc(3) \mid sc(X))$ where $\theta' = \{X = 3\}$. Given that we know the next ordinary state, we can obtain the real emission symbol, say $st(plus, 10)$ from the abstract emission symbol $st(L, T)$ by applying μ again. The relevant probability is given by $\mu(st(plus, 10) = st(L, T)\theta'' \mid st(L, T))$ (where $\theta'' = \{L = plus, T = 10\}$). Note that in case of a transition such as $hc(1) \longleftarrow sc(X)$ with $a(hc(1) \longleftarrow sc(X)) = 0.5$, the model makes a transition from any ground instance of $sc(X)$ to $hc(1)$ with probability 0.5 because $hc(1)$ will always be selected. We have no choice, i.e. $\mu(hc(1) \mid hc(1)) = 1.0$.

So far we have not said how μ can be specified. Well, in general, any probabilistic representation such as Markov chains or Bayesian networks might be used to specify μ. However, in our experiments, we followed a naive Bayes approach. To each argument a_i, $1 \leq i \leq l$ of predicate h, we associated a finite domain $\mathrm{D}(a_i)$ and a probability distribution P_i over $\mathrm{D}(a_i)$. Let $V_1, \ldots, V_m$ be the unbound variables occurring in $h(t_1, \ldots, t_l)$ in arguments $a_{i1}, \ldots, a_{im}$, and let $t_{i1}, \ldots, t_{im}$ be the corresponding terms in $h(t_1, \ldots, t_l)\theta$. We set the selection probability to

$$\mu(h(t_1, \ldots, t_l)\theta \mid h(t_1, \ldots, t_l)) = \prod_{j=1}^{m} P_{ij}(t_{ij})$$

assuming that all arguments are independent of each other. E.g., to compute

$$\mu(he(h(right, alpha), 4) \mid he(T, L)),$$

we compute the product of $P_T(h(right, alpha))$ and $P_L(4)$.

The abstract transitions from the artificial state *start* are a special case of a transition. We force the output to be the empty word.

4.4. *Default Reasoning Revisited*

So far, for the sake of simplicity, we have left out in the formal definition one important requirement for $\mathcal{T}$:

> **Requirement 1:** The sum of the probabilities over all outgoing transitions in a LOHMM of each abstract and each ordinary state should be 1.0.

This requirement will be satisfied for HMMs. However, when we have LOHMMs involving exceptions as in Figure 4, it may be violated. Indeed, in this example the state $hc(2)$ four rules apply. This includes the two rules directly specified for $hc(2)$

$$sc(X) \overset{st(O,L):0.6}{\longleftarrow} hc(2),$$
$$hc(Y) \overset{he(T,L):0.4}{\longleftarrow} hc(2).$$

and two rules obtained by instantiating the rules for the abstract state $hc(X)$:

$$sc(X) \overset{st(O,L):0.5}{\longleftarrow} hc(2),$$
$$hc(Y) \overset{he(T,L):0.5}{\longleftarrow} hc(2).$$

This set of rules clearly violates Requirement 1, because the outgoing edges of $hc(2)$ in the grounded LOHMM sum to 2.0.

This is not surprising because we have not yet taken account of the exception for $hc(2)$. Default reasoning was precisely introduced to cope with such exceptions and to allow the more specific rules for $hc(2)$ to override the more general ones for $hc(Y)$. In order to guarantee that Requirement 1 will be full filled also under the presence of exceptions, we require that :

Requirement 2: For each ordinary state s in a LOHMM, there is a unique least general abstract state l that appears in the LOHMM. Furthermore, only those transitions for s that are an instantiation of clauses of the form $h \overset{o}{\longleftarrow} l$ may be used.

Under Requirement 2 the above sketched problem with $hc(2)$ disappears.

4.5. Semantics and Evaluation

Each HMM is a LOHMM consisting of propositional, logical transitions only. Having the described grounding/unrolling process in mind, it is clear that a LOHMM $\mathcal{T}$ defines a HMM given a *selection distribution* μ for each predicate used in $\mathcal{T}$. There are a finite set of abstract transitions and each domain associated to an argument of a predicate is finite. Thus, the set of states and, therefore, the set of (ground) transitions is finite. Note that the selection distribution μ might extend the language of $\mathcal{T}$ by a finite set of constant symbols not occurring in $\mathcal{T}$. Furthermore, the artificial starting state specifies a prior distribution over abstract states and not states. The latter is influenced by the processes of selecting the state. To summarize, there are two primary differences to HMMs. First, transition probabilities are defined by a product of abstract transition probabilities and the selection probability. Second, the set of states represented by an abstract state, and, therefore, the set of predecessors of a state can vary according to the domains associated to predicates – the set of states it is influenced by.

Having the analogy to HMMs in mind, there are three key problems of interest to be solved for LOHMMs. Let be $O = o_1, o_2, \ldots, o_T$ a finite sequence of ground observations:

Evaluation: Given a LOHMM, its parameter set λ, and a set μ of selection distributions, what is the probability $P(O \mid \mu, \lambda)$ that the sequence O was generated by the model?

Most likely state sequence: Given an LOHMM, its parameter set λ, a set μ of selection distributions, and an observation sequence O, find a state sequence S^* that is most likely to produce the observation sequence, i.e. $S^* = \arg\max_S P(S \mid O, \mu, \lambda)$.

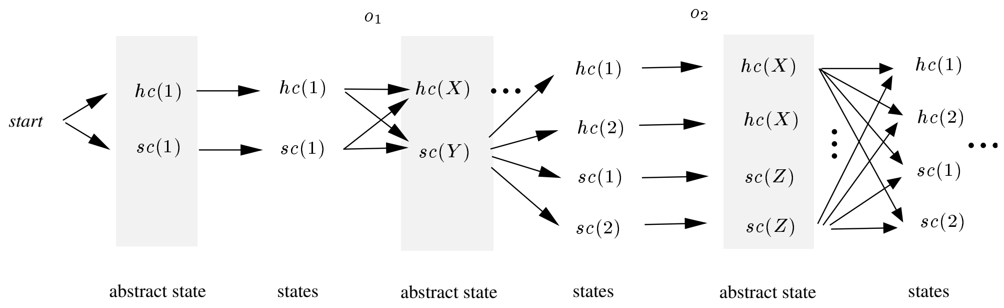


Figure 6. Illustration of the trellis induced by the LOHMM in Figure 4. In contrast with HMMs, there is an additional layer where the states are sampled from abstract states.

Parameter estimation: Given the structure of an LOHMM, a set μ of selection distributions, and a set of observation sequences $\{O_t\}$, what is the most likely parameter set λ^* of the model, i.e. $\lambda^* = \arg\max_\lambda P(\lambda \mid \mu, \{O_t\})$.

Efficient ways to evaluate LOHMMs are adaptations of the *forward* and *backward procedures*. The main difference to the HMM case is that a transition is specified in two phases, as shown in Figure 6. After selecting an abstract transition, μ generates the relevant states from the head of the abstract transition. However, due to space restrictions, we will not explain here how to adapt them.

Having forward and backward procedures, it is possible to adapt the Viterbi algorithm as a solution to the second problem, and, above all, to adapt the Baum-Welch algorithm to maximum-likelihood estimate the abstract transition probabilities. To adapt the Baum-Welch algorithm, we utilized the fact that every (ground) transition is an instance of exactly one abstract transition. Therefore, the expected counts of an abstract transition can be computed based on the counts of its ground instances on the data. Our approach is similar to the one [KP97] explored to learn probabilities for noisy first-order rules. As our algorithm is an instance of the EM algorithm, it increases the likelihood of the data with every update, and according to [MK97], it is guaranteed to reach a stationary point. All standard techniques to overcome limitations of EM algorithms are applicable. Thus, random parameter initializations and random restart avoid saddle points and help to find a global maximum. Pseudocounts ensure that no probability is estimated as zero.

To estimate the probabilities over the domains associated to predicates, i.e. to estimate the selection probabilities implicitly, we adapt the usual parameter learning of naive Bayes models. The probabilities are the fraction of times the domain element was observed to occur over the total number of opportunities. Again, pseudocounts ensure estimates greater than zero. This leads to the following overall EM scheme: In each iteration, estimate first the abstract transition probabilities and then the probabilities over the domains associated to each predicate.

5. Preliminary Experiments

In the experiments described below, we implemented the EM algorithm (with pseudo-counts) using the Prolog system Sicstus-3.8.6. The experiments were ran on a Pentium-III-600 MHz machine. Our task was to classify sequences representing protein secondary structures into one of five folds. To do so, we followed the standard approach to do classification based on HMMs.

We chose a LOHMM like the one in Figure 7, fixed its structure, and randomly generated for each fold a set of initial abstract transition probabilities and domain distributions. From each fold dataset described in Section 2, we randomly sampled a training set consisting of 100 sequences. The remaining sequences were used as a test set. Then, we trained these five LOHMMs, one per fold. We used a simple, but common stopping criterion: EM stops if a change in log-likelihood is less than 10^{-1} from one iteration to the next. To evaluate the learned models, we computed the log-likelihood that each model gave to a sequence in the test sets. If the i-th model was the most likely one, then we classified the sequence as a member of class i. Although we experimented with other LOHMM structures as well, we report only on the classification results of the best performing LOHMM structure.

The best performing LOHMM structure is given in Figure 7. Being in a block of consecutive helices (resp. strands), the model will remain in the block or transition to a new block of strands (resp. helices). This model takes into account type, length and orientation information. Moreover, there are specific abstract transitions for helices of types $h(right, alpha)$ and $h(right, 3to10)$, and for parallel and anti-parallel strands, and for being at the beginning of a sheet. This enabled us to model the "process" within blocks of consecutive helices quite detailed, and of transitions from blocks of consecutive helices to strands and vice versa.

Our implementation of EM took at most five iterations and approximately 5 minutes to estimate the maximum-likelihood parameters per fold. Given our discretization of the helix and strand lengths, the LOHMM consisted of 74 abstract transition and 46 domain distribution probabilities, whereas the grounded LOHMM would consist of over $62,000$ transition probabilities. So, the abstract representation of states and transitions in LOHMMs achieves a remarkable compression of the model, which supports hypothesis H4.

The classification results are summarized by the confusion matrix in Table 1. In this section, the TIM beta/alpha-barrel fold will be denoted as $fold1$, the NAD(P)-binding Rossmann-fold as $fold2$, the Ribosomal protein L4 fold as $fold23$, the glucosamine 6-phosphate deaminase/isomerase fold as $fold37$, and the leucine aminopeptidas fold as $fold55$. In total, 75% (1264 out of 1684) sequences were correctly classified. This result is in the same range as the one reported by [TMS01a] (75% as well). However, we have to emphasize that the datasets are not completely comparable. In contrast to this result, a learner predicting always the majority class would achieve an predictive accuracy of 43%. These results suggest that hypothesis H1 holds.

actual \ predicted	fold1	fold2	fold23	fold37	fold55
fold1	522	28	25	33	12
fold2	26	186	28	13	6
fold23	12	8	131	23	0
fold37	33	30	20	241	17
fold55	0	2	0	4	184

Table 1
Confusion matrix showing actual vs. predicted fold classification.

	Precision	Recall
fold1	0.88	0.84
fold2	0.73	0.72
fold23	0.64	0.75
fold37	0.77	0.71
fold55	0.84	0.97

Table 2
Precision and recall for each fold rounded to second decimal.

In Table 2, we also give our preliminary results in terms of the *recall* and *precision*. Recall is defined as the sum of true positives divided by the sum of true positives and false negatives. Precision is defined as the sum of true positives divided by the sum of true positives and false positives. As can be seen, the recall and precision figures vary among the folds, but within the folds recall and precision are well balanced. In other words, a good precision is not bought at the expense of a good recall, and vice versa.

Finally, we inspected the trained LOHMM for characteristic differences. To this purpose, we generated for each of the five models plots of the estimated probability distributions associated to predicates implicitly defining μ (see Figure 8). Upon visual inspection, differences can be found as follows (hypothesis H2):

- Regarding the helix types, $fold23$ differs from the others in that the probability of selecting right-handed alpha helices seems to be lower. Also, the probability of right-handed 3to10 helices to be selected seems to be higher than for the other folds.
- According to the strand lengths, we can group the first three and the last two folds.
- As for strand orientations, we have uniform "patterns" for $fold1$, $fold2$ and $fold23$, but characteristic patterns for $fold37$ and $fold55$.

We are planning to subject these observations to reviews by domain experts in the area of protein chemistry. Overall, despite the preliminary character of the experiments, we believe it is already justified to claim hypothesis H3.

6. Related Work

[GKHC01] presented an approach to sequence annotation based on profile HMMs trained on the primary structure of domains for each superfamily in SCOP. The present study is conducted on a different level of abstraction: Firstly, we are working with a secondary structure representation, and not a primary structure representation. Secondly, we are dealing with SCOP folds, not SCOP superfamilies. It might be interesting to apply our approach also at the (more detailed) superfamily level. The goal in [GKHC01] was to annotate sequences based on a library of HMMs that represent all proteins of known structure. In contrast, our short-term goal was to give a proof of the principle, with the intermediate-term goal of providing a tool that helps to gain insights into structural characteristics.

[TMS01b,TMS01a] applied the Machine Learning and Inductive Logic Programming (ILP) tool Progol to a similar task as the one tackled in this paper. The task there was also to predict SCOP folds based on a high-level logical representation. The difference is that we are working with a larger, more recent dataset, a different representation, and that we are applying a different Machine Learning approach based on probability theory.

The idea of LOHMMs is also related to Hierarchical HMMs (see e.g. [FST98]), where states themselves may be composed of HMMs on a smaller scale. The difference is that the abstract states and transitions in our approach do not *consist* of more detailed states and transitions, but *summarize* states and transitions, without grounding them a priori. Furthermore, although we have not explored the relationship yet, LOHMMs seem to be closely related to HMMs based on tree automata (see e.g. [FSV02]).

7. Conclusion

In this paper, we have applied Logical Hidden Markov Models (LOHMMs), a new Machine Learning technique combining probability theory and logic, to the task of finding structural signatures of protein folds. LOHMMs offer the possibility to specify states and transitions in an abstract fashion, and thereby offer a significant reduction in the model size compared to regular HMMs. We have shown that the generalization performance is satisfactory as well, and that it is easy to extract characteristic patterns from the trained models. Although we know that there is much room for improvements, the application has shown that LOHMMs have a potential to become a valuable tool for the discovery of structural signatures in protein folds.

8. Acknowledgements

This research was partly supported by the European Union IST programme under contract number IST-2001-33053 (Application of Probabilistic Inductive Logic Programming – APRIL).

9. Appendix

9.1. *Logical Preliminaries*

Here, we provide some logical preliminaries. For a general introduction, we refer to [Llo89].

A (relational) alphabet consists of a set P of predicate symbols and a non-empty set C of constant symbols. Each *predicate symbol* has an arity n. A *term* t is either a variable V or a constant symbol c. An *atom* $p(t_1, \ldots, t_n)$ is a predicate symbol p followed by a bracketed n-tuple of terms t_i where $t_i \in \mathrm{D}(p)_i$ if t_i is not a variable. An *iterative clause cl* is a formula of the form $A \leftarrow B$, where A and B are logical atoms, and can be read as A if B. All variables in iterative clauses are universally quantified, although this is not explicitly written. We call $A =: \mathrm{head}(cl)$ the head and $B =: \mathrm{body}(cl)$ the body of the clause. The set of variables in a term, atom or clause E, is denoted as $\mathrm{vars}(E)$. A substitution $\theta = \{V_1/t_1, \ldots, V_n/t_n\}$, is an assignment of terms t_i to variables V_i. Applying a substitution θ to a term, atom or clause e yields the instantiated term, atom, or clause $e\theta$ where all occurrences of the variables V_i are simultaneously replaced by the corresponding term t_i (w.r.t. to the domains of predicates). A substitution θ is called a *unifier* for a finite set S of atoms if $S\theta$ is singleton. A unifier θ for S is called a *most general unifier* (MGU) for S, denoted by $\mathrm{mgu}(S)$, if, for each unifier σ of S, there exists a substitution γ such that $\sigma = \theta\gamma$. A term, atom or clause E is called *ground* when it contains no variables, i.e. $\mathrm{vars}(E) = \emptyset$.

References

[DK88] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-1988)*, 1988.

[FST98] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: analysis and applications. *Machine Learning*, 32, 1998.

[FSV02] P. Frasconi, G. Soda, and A. Vullo. Hidden markov models for text categorization in multi-page documents. *Journal of Intelligent Information Systems, (special issue on Automated Text Categorization)*, 18:195–217, 2002.

[GKHC01] J. Gough, K. Karplus, R. Hughey, and C. Chothia. Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. *Journal of Molecular Biology*, 313(4):903–919, 2001.

[KP97] D. Koller and A. Pfeffer. Learning probabilities for noisy first-order rules. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-1997)*, pages 1316–1321, Nagoya, Japan, August 23-29 1997.

[Llo89] J. W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 2. edition, 1989.

[MK97] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New York, 1997.

[Rab89] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2), 1989.

[RJ86] L. R. Rabiner and B. H. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, January:4–16, 1986.

[TMS01a] M. Turcotte, S. Muggleton, and M. J. E. Sternberg. Discovery of structural signatures of protein fold and function. *Journal of Molecular Biology*, 306(3):591–605, 2001.

[TMS01b] M. Turcotte, S. Muggleton, and M. J. E. Sternberg. The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Machine Learning*, 43(1/2):81–95, 2001.

[WZ95] R. J. Williams and D. Zipser. Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity. In *Back-propagation:Theory, Architectures and Applications*. Hillsdale, NJ: Erlbaum, 1995.

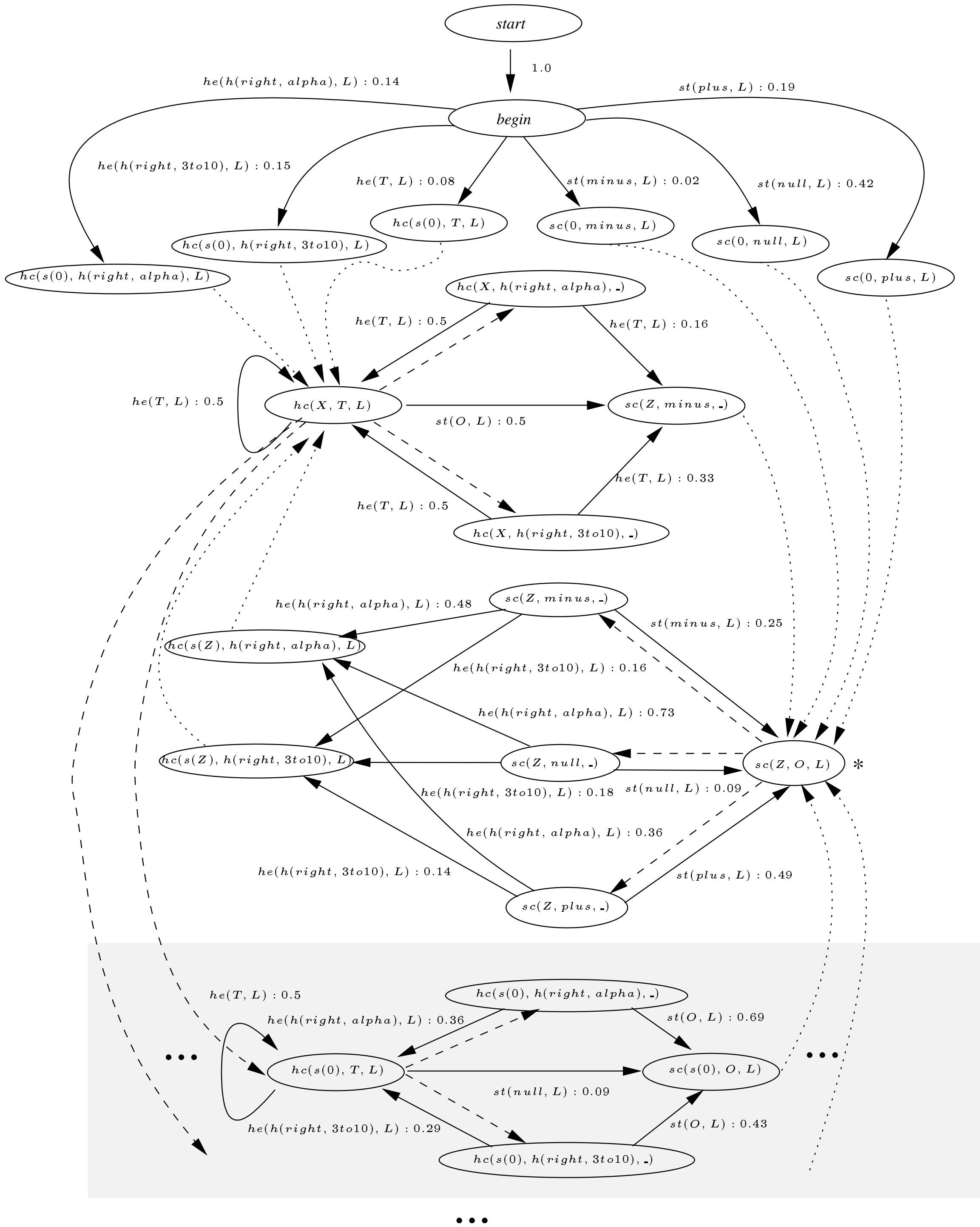
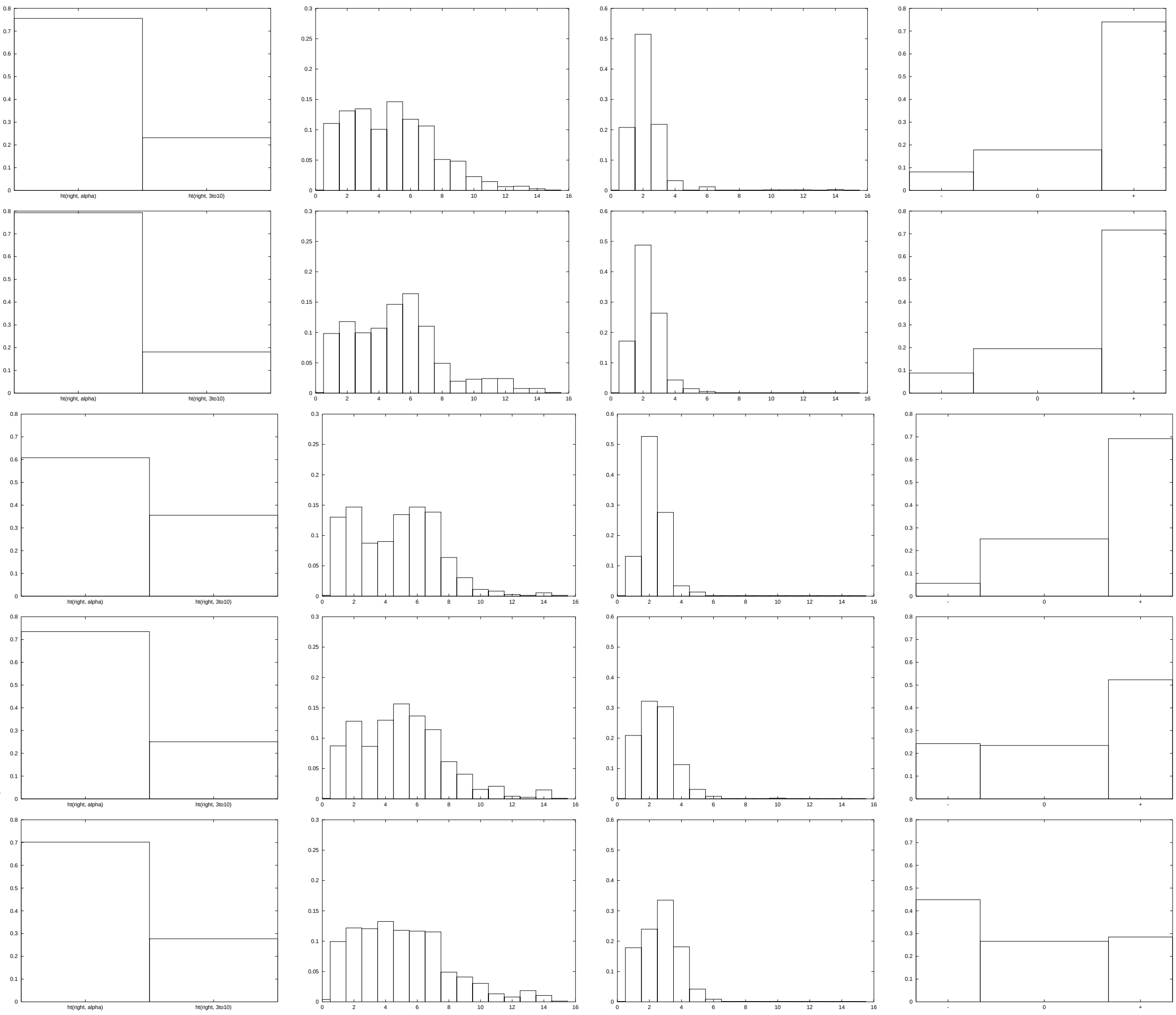


Figure 7. A first order (hidden) Markov model of protein secondary structures. The probabilities are those for *fold1*. For the sake of simplicity, we omitted the explicitly modelled *end* state. Abstract states where the outgoing abstract transition probabilities do not sum to 1.0 have an abstract transition to *end*. Furthermore, there are six copies of the shaded part for $hc(s^2(0), T, L), \ldots, hc(s^7(0), T, L)$. This explicitly models the abstract transition for the first seven blocks of helices. Note that the abstract state labeled with "*" is only used for default reasoning. The predicates $hc(HelixCounter, Type, Length)$ and $sc(HelixCounter, Orientation, Length)$ represent blocks of consecutive helices (resp. strands). The $HelixCounter$ enables us to have general directed transitions from one block to exactly one successor block. While this model might appear quite complex, its HMM counterpart would consist of more than $62,000$ transitions.



fold1
fold2
fold23
fold37
fold55

helix types helix lengths strand lengths
strand orientations

Figure 8. Estimated selection distributions for the five folds. The distribution over helix types shows, that only the types $h(right, alpha)$ and $h(right, 3to10)$ occured in the data. Note that, due to pseudocounts, no probability value is zero.