

Transducer Inference by Assembling Specific Languages^{*}

Piedachu Peris and Damián López

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia,
Camino de Vera s/n
46071 Valencia, Spain
{pperis,dlopez}@dsic.upv.es

Abstract. Grammatical Inference has recently been applied successfully to bioinformatic tasks as protein domain prediction. In this work we present a new approach to infer regular languages. Although used in a biological task, our results may be useful not only in bioinformatics, but also in many applied tasks. To test the algorithm we consider the transmembrane domain prediction task. A preprocessing of the training sequences set allows us to use this heuristic to obtain a transducer. The transducer obtained is then used to label problem sequences. The experimentation carried out shows that this approach is suitable for the task.

Keywords: Inference of regular languages, bioinformatics, protein motif location.

1 Introduction

Formal Language Theory and Grammatical Inference (GI) are playing an important role in the development of new methods to process biological data [1,2]. Many works propose GI techniques to tackle bioinformatic tasks as: secondary structure identification [3], protein motifs detection [4,5], optimal consensus sequence discovery [6,7], gene prediction [8] or multiple sequence alignment [9].

The selection of proteins with certain characteristics from amino acid sequences is a central goal of computational biology. One aspect of this problem is to detect certain sub-sequences, known as domains or motifs, with some interesting functional features.

Among the membrane related proteins, some are integrally in the membrane and others span the hydrophobic core of the membrane. Note that this fact classifies the segments of the protein sequence into: transmembrane, inside and outside regions. In this work we deal with transmembrane protein sequences which, from now on, we will refer to as membrane proteins. Figure 1 shows a schematic representation of these proteins.

Under a biological point of view, membrane proteins play an important role in a variety of important biological functions [10,11], mainly as receptors or

^{*} Work supported by the spanish CICYT under contract TIN2007-60769.

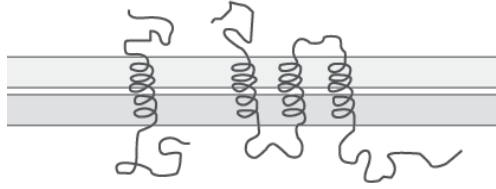


Fig. 1. Schematic representation of two transmembrane proteins (one single-spanning and one multi-spanning). The grey bars represent the (two) lipidic layers of the membrane.

transporters. In order to identify relevant features of a given membrane protein, as well as their role in the cell [12], the number of transmembrane segments of a protein and some characteristics such as loop lengths are to be taken into account.

Thus, an important and interesting task is to predict the location of transmembrane domains along the sequence, since these are the basic structural building blocks defining the protein topology. Several works have dealt with this prediction task from different approaches, mainly using Hidden Markov Models (HMM) [13,14,15], neural networks [16,17] or statistical analysis [18]. A rich literature is available on proteins prediction. For reviews on different methods for predicting transmembrane domains in proteins, we refer the reader to [19,20,21].

In our work, we use a grammatical approach to locate transmembrane motifs within protein sequences. We extend previous work by Peris et al. that propose approaches to predict the coiled-coil and the transmembrane domain [4,5].

Briefly, the approach proposed in this work is based on the assumption that all the transmembrane, inside and outside regions share common features suitable to be modelled grammatically. Thus, we propose a method that infers automata for each kind of region and another automaton to model the sequence of regions that appear in the training set. A substitution allows us to obtain the final transducer. Several works have proposed operations over languages by means of automata transformations. For instance, [22] used a specialization strategy from positive examples that consists in creating recursive automata by replacing transitions by states of the same automaton.

The results of our experimentation are compared with other existing approaches and show that this approach improves previous work performance. Our work is organized as follows: Section 2 summarizes some definitions and the notation used; Section 3 explains our approach to the problem; Section 4 shows the experimental results and the indexes used to compare our results with previous ones. Finally, some conclusions and future lines of research end the paper.

2 Notation and Definitions

Let Σ be an alphabet and Σ^* the set of words over the alphabet. For any word $x \in \Sigma^*$ let x_i denote the i -th symbol of the sequence and let $|x|$ denote the

length of the word. Let also λ denote the empty word. A language over Σ is any set $L \subseteq \Sigma^*$.

A finite automaton is defined as a tuple $A = (Q, \Sigma, \delta, I, F)$, where Q is a finite set of states, Σ is an alphabet, $I, F \subseteq Q$ are the sets of initial and final states and $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function. For the sake of clarity, we will consider the transition function as a subset of $Q \times \Sigma \times Q$. This function can also be extended in a natural way to consider words over an alphabet instead of symbols. The language accepted by the automaton is $L(A) = \{x \in \Sigma^* : \delta(q_0, x) \cap F \neq \emptyset\}$. A finite automaton is deterministic (DFA) if the transition function is defined as $Q \times \Sigma \rightarrow Q$.

A language L is *k-testable in the strict sense* (*k-TSS*) if there exist sets $P, S \subseteq \Sigma^{k-1}$ and $N \subseteq \Sigma^k$ such that $L - \{\lambda\} = (P\Sigma^* \cap \Sigma^*S) - \Sigma^*N\Sigma^*$.

A *finite state transducer* is defined by a system $\tau = (Q, \Sigma, \Delta, q_0, Q_F, E)$ where: Q is a set of states and $Q_F \subseteq Q$ is the set of final states; Σ and Δ are the input and output alphabets respectively; q_0 is the initial state, and $E \subseteq (Q \times \Sigma \times \Delta^* \times Q)$ is the set of transitions of the transducer.

Given an input word $x = a_1, a_2, \dots, a_n$, a successful path in a transducer is a sequence of transitions $(q_0, a_1, o_1, q_1), (q_1, a_2, o_2, q_2), \dots, (q_{n-1}, a_n, o_n, q_n)$ where $q_n \in Q_F$, and $a_i \in \Sigma^*$, $o_i \in \Delta^*$ and $q_i \in Q$ for $1 \leq i \leq n$. Note that a path can be denoted as $(q_0, a_1a_2 \dots a_n, o_1o_2 \dots o_n, q_n)$ whenever the sequence of states are not of particular concern. A transduction is defined as a function $t : \Sigma^* \rightarrow \Delta^*$ where $t(x) = y$ if and only if there exists a successful path (q_0, x, y, q_n) . We refer the interested reader to [23].

3 Combination of Specific Languages

Usually, the class of 2-TSS is referred to as the class of *local* languages. This is a very important subclass of regular languages. A strong result that relates both classes states that a language $L \subseteq \Sigma^*$ is regular if and only if there exists a finite alphabet Σ' , a local language $K \subseteq \Sigma'^*$ and a morphism h that maps Σ'^* into Σ^* such that $L = h(K)$.

In [24] the authors take into account this result to propose a methodology that allows, whenever relevant expert information is available, to infer regular languages from positive data. This methodology (MGGI) has been widely and successfully applied mainly to language and dialog processing (for instance [25,26]). Note also that the inclusion of this *expert* knowledge allows to address practical tasks under a GI approach using only positive presentation. This is important because negative information usually is hard to define in applied tasks.

In a similar way MGGI does, we assume that, under a classical GI framework and for some practical tasks, it is possible to detect sub-sequences of the training set M that model/share a common feature f . This allows the sequences to be labelled, and therefore, to take into account this knowledge.

In this work we isolate the sub-sequences with the same feature label in order to build *more specific* training subsets. Thus, we obtain, for each feature f considered, a set of samples M_f that allows us to infer a language L_f .

$$M = \left\{ \begin{array}{l} b_1 a_1 a_1 a_1 b_1 b_1 b_1 a_2 b_2 c_2 a_2 a_2 c_2 b_3 a_3 c_3 a_3 b_3 a_3 b_4 a_4 a_4 b_4 a_4 c_4 b_4 a_4, \\ a_4 b_4 b_4 c_4 a_4 b_1 b_1 c_1 a_1 c_1 a_3 c_3 a_3 b_3 b_3 a_3 b_4 c_4 a_4 a_4 b_4 a_2 a_2 c_2 b_2 a_2, \\ c_1 b_1 a_1 b_1 b_1 a_1 a_1 c_2 c_2 b_2 a_2 b_2 a_2 a_2 b_3 a_3 c_3 b_2 a_2 a_2 b_4 a_4 a_4 c_4 a_4 c_4, \\ a_1 b_1 b_1 b_1 c_2 b_2 c_2 a_2 a_2 b_3 a_3 a_4 a_4 a_4 b_4 a_4 c_2 a_2 a_2 b_2 a_2, \\ a_4 b_4 b_4 a_4 a_4 a_2 a_2 b_2 a_2 a_2 b_3 b_3 a_3 b_3 b_3 a_3 b_2 a_2 a_2 b_2 a_2 a_2 a_2 \end{array} \right\}$$

$$S(M) = \left\{ \begin{array}{l} 1234, \\ 41342, \\ 12324, \\ 12342, \\ 4232 \end{array} \right\} \quad M_1 = E(M, 1) = \left\{ \begin{array}{l} baaabbbb, \\ bbcac, \\ cbabbaaaa, \\ abbbb \end{array} \right\} \quad M_3 = E(M, 3) = \left\{ \begin{array}{l} baacaba, \\ acaabba, \\ bac, \\ baba, \\ abbaabba \end{array} \right\}$$

$$M_2 = E(M, 2) = \left\{ \begin{array}{l} abcaac, \\ aacba, \\ ccbabaaa, \\ baab, \\ cbcaaa, \\ caaba, \\ aaabaa, \\ baabbabaaa \end{array} \right\} \quad M_4 = E(M, 4) = \left\{ \begin{array}{l} baababacba, \\ abbca, \\ bcaab, \\ baacac, \\ aaaba, \\ abbaa \end{array} \right\}$$

Fig. 2. Scheme of the preprocessing. The different features considered are denoted by sub-indexes in the original training set M . Note that the derived training sets $E(M, i)$ contain only those sub-sequences with the corresponding feature.

The original training set M is modified by substituting the extracted sub-sequences by a unique feature identifier.

In a more formal way, let Σ denote the strings alphabet and $\Delta = \{i_1, i_2, \dots, i_k\}$ the feature identifiers alphabet. Let us denote the labelled alphabet by Σ_Δ . Thus, Σ_Δ^* denotes the set of all possible strings over Σ whose symbols are labelled with symbols in Δ . Let the homomorphism $h : \Sigma_\Delta \rightarrow \Sigma$ be defined as $h(a_I) = a$, for each $a_I \in \Sigma_\Delta$. That is, the homomorphism that erases the labelling of the symbols. Let us define the *simplification* function as follows:

$$S : \Sigma_\Delta^* \rightarrow \Delta^*$$

$$S(u_1 u_2 \dots u_m) = I_1 I_2 \dots I_m \text{ where: } u_i \in \Sigma_{\{I_i\}}^* \quad (1)$$

Let us define the *extraction* function as follows:

$$E : \Sigma_\Delta^* \times \Delta \rightarrow \mathcal{P}(\Sigma^*)$$

$$E(u_1 u_2 \dots u_k, I) = \{h(u_i) : u_i \in \Sigma_{\{I\}}^*\} \quad (2)$$

We also extend these functions to operate on sets of strings, thus, for any set of labelled strings M and a label identifier I :

$$S(M) = \bigcup_{w \in M} S(w) \quad E(M, I) = \bigcup_{w \in M} E(w, I)$$

Figure 2 shows schematically an example of this preprocessing.

Please, note that it may be possible that the considered features do not cover completely the training sequences in M . In that case it is possible to consider an extra identifier to label those sub-sequences.

Note also that, it is possible to use whichever combination of inference algorithms with the feature training sets obtained. The main assumption we make is that the more specific the data, the better the automata inferred. Let us denote by $L_{S(M)}$ ($A_{S(M)}$) the language (resp. automaton) inferred from the set $S(M)$, as well as L_{M_i} (A_{M_i}) will denote the language (resp. automaton) inferred considering the strings in M_i .

In order to build the final transducer, a substitution over $L_{S(M)}$ is carried out. This substitution guarantees that, for each transition (p, f_i, q) of $A_{S(M)}$, it is possible to reach in the transducer the state q with x from the state p , where $x \in L_{M_i}$. An implementation of this language operation may consist of substituting each transition (p, f_i, q) in $A_{S(M)}$ by the automaton that models the same feature (A_{M_i}), adding λ transitions in order to connect state p with the initial state of A_{M_i} , as well as the final states of A_{M_i} with state q . All the transitions of A_{M_i} should also be modified to consider the same feature output symbol f_i .

Example 1 illustrates our approach.

Example 1. Let us consider the following set of labelled strings:

$$M = \{a_1 a_1 b_2 c_2 c_2 c_3 c_3, b_2 b_2 c_2 c_3 c_3, b_2 c_2 c_3 b_2 c_2 c_3\}$$

from the alphabet Σ_Δ . The output of the function 1 when applied to each string in M is the set of sequence identifiers $S(M) = \{123, 23, 2323\}$. In the same way, when the set M is considered, the function 2 returns the sets of sub-strings labelled with the same feature identifier, that is: $M_1 = \{aa\}$, $M_2 = \{bcc, bbc, bc\}$ and $M_3 = \{cc, c\}$.

Then, an automaton is inferred for each set obtained by the preprocessing: $A_{S(M)}$, A_{M_1} , A_{M_2} and A_{M_3} . If it is considered an algorithm to infer local languages, the resulting automata are shown in Figure 3.

Finally, a substitution over $L_{S(M)}$ is carried out. We note here that several approaches can be used to obtain a probabilistic transducer. The study of which one has the better experimental behaviour is not pursued in this paper.

In this work, we infer probabilistic automata for the preprocessed sets, although probabilities are not shown in Figure 3. In order to maintain the distribution of probability as much as possible, we follow the procedure described above. Thus, the implementation of the substitution replaces each transition (p, l, q) with the corresponding automaton (i.e. M_l) and connects the state p and the initial state of M_l , as well as the final state(s) of M_l and q , with λ -transitions. The transitions of M_l are also modified to consider the transduction. The λ -transitions are removed using a traditional algorithm. The resulting (non-deterministic) transducer is neither determinized nor minimized. For this example, the final transducer is shown in Figure 4.

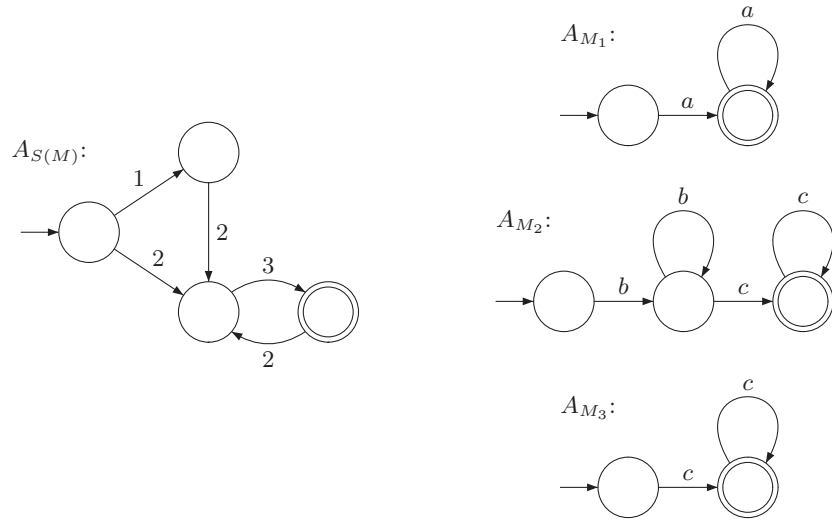


Fig. 3. Automata inferred for the sets obtained by the preprocessing of M

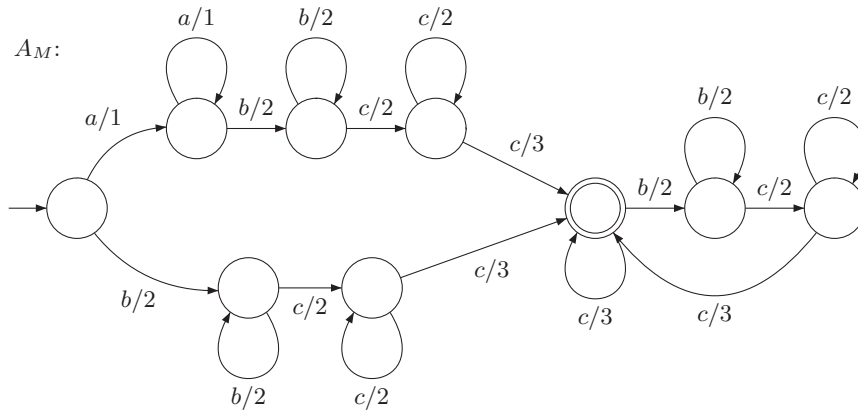


Fig. 4. Transducer obtained by our approach when it is considered the data set $M = \{a_1 a_1 b_2 c_2 c_2 c_3 c_3, b_2 b_2 c_2 c_3 c_3, b_2 c_2 c_3 b_2 c_2 c_3\}$

4 Experimental Results

4.1 Dataset

We used a dataset composed of 160 membrane proteins in order to evaluate the performance of our approach. Of the proteins included in the dataset, 108 are multi-spanning proteins and 52 are single-spanning. This dataset has

been introduced by [13], and will be referred to as the TMHMM set. Most of the topology data included in TMHMM have been determined with biochemical and genetic methods. Only the structure of a small number of membrane protein domains have been determined at an atomic resolution. On the one hand, this gives the set biological relevance, but on the other hand, these methods are considered not completely reliable, and may output contradictory topologies for the same protein sequence. We removed from the original dataset those protein sequences for which different (biochemical or genetic) methods output different topologies. This dataset is available for the community at: <http://people.binf.ku.dk/krogh/TMHMM/>.

4.2 Performance Measures

In literature, different measures have been proposed to evaluate sequence analysis methods, especially gene-finding methods. An exhaustive review of these measures can be found in [27]. The most used measures in functional domain location tasks are probably: recall or sensitivity (Sn); and precision or specificity (Sp). These measures can be computed as follows:

$$Sn = \frac{TP}{TP + FN} \quad Sp = \frac{TP}{TP + FP}$$

where:

True positives (TP): correctly localized amino acids into a TM domain.

True Negatives (TN): correctly annotated amino acids out of a TM domain.

False positives (FP): amino acids out of a TM domain annotated as belonging to a domain.

False Negative (FN): amino acids into a TM domain not correctly localized (annotated as out of any domain).

Sn and Sp , however, taken by themselves, do not constitute an exhaustive measure. A measure that is more complete and summarizes both Sn and Sp is the *Correlation Coefficient (CC)* or Matthews Correlation Coefficient [28], which also presents some interesting statistical properties. It is computed as follows:

$$CC = \frac{(TP \cdot TN) - (FN \cdot FP)}{\sqrt{(TP + FN) \cdot (TN + FP) \cdot (TP + FP) \cdot (TN + FN)}}$$

The main drawback of the CC measure is that it is not defined if any factor of the root is equal to zero. Some measures have been proposed to tackle this issue. We have selected the *Approximate Correlation (AC)* which is calculated as follows:

$$AC = \left(\frac{1}{4} \left[\frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN + FN} \right] - 0.5 \right) \cdot 2$$

In the results, we omitted the samples for which it was not possible to calculate CC (independently of the dataset considered). On the other hand, AC

has a 100% coverage, which could explain the considerable difference that can be observed in some experiments between AC and CC values.

We also used a segment-based measure, called Segment overlap, (Sov_δ^{obs}) defined in [29]:

$$Sov_\delta^{obs} = \frac{1}{N} \sum_s \frac{\min(E) - \max(B) + 1 + \delta}{\max(E) - \min(B) + 1} len(s_1)$$

where N is the total number of amino acids observed within all the domains of the protein, s_1 and s_2 are two overlapped segments, E is $\{end(s_1); end(s_2)\}$, B is $\{beg(s_1); beg(s_2)\}$ and δ is a parameter for the accepted (maximal) deviation. We used a value of $\delta = 3$.

We have also calculated the number of transmembrane segments correctly predicted at three accuracy thresholds: 100%, 90% and 75%, that is, number of segments with the 100%, 90% or more, and 75% or more of their amino acids are correctly predicted. This measure is similar to Sensibility, but it is based on segments. This measure allows to obtain a reliable evaluation for those segments that contain false negatives not only at the ends of the segment. For example, this occurs when a domain sequence is predicted as more than one segment, and there are some false negatives between two of this predicted segments. In other words, these values show the continuous coverage degree of the prediction. A drawback of these measures is their sensitivity to overprediction. Therefore, it is necessary to complement it with the Sp measure.

4.3 Results

We considered inference of k -TSS languages to obtain the automata which model the distinct regions. Looking for the best behaviour, the experimentation was run using values of k from 2 to 7. We recall that, in this task, three different labels are considered, that is: inner regions, outer regions, and transmembrane domains. The best results were obtained using the following k -values: $k = 3$ to infer the inner and transmembrane automata models; $k = 6$ to infer the outer model; and $k = 2$ to infer the automata for the set of label sequences.

In order to test our method, we followed a *leaving one out* scheme. In this scheme, for each protein of the dataset, a transducer is obtained using the rest of sequences in the dataset. The transducer is then used to process the sequence left out. This process is repeated until all proteins have been used as test sequences.

Table 1 shows the accuracy of our approach (igS), along with the results we obtained with TMHMM 2.0[13] and $igTM$ [5]. In order to do the comparison, we run our method, TMHMM and $igTM$ over the same dataset of 160 proteins. $igTM$ is a Grammatical Inference approach which achieves values close to 80% in both specificity and sensitivity. TMHMM 2.0 is a method based on a Hidden Markov Model, whose prediction accuracy for membrane domain location is near 83%. It is worth to note here that this tool is available as a closed package and, therefore, no *leaving one out* procedure has been carried out for this method. Thus, these results may have some slight bias when compared with other results.

Table 1. Results of the experiments carried out with *igS* over the 160 proteins dataset, compared to the results of TMHMM 2.0 and the two best configurations of *igTM* over the same dataset

		TMHMM database							
		Sn	Sp	CC	AC	Sov_{δ}^{obs}	100% \geq 90%	\geq 75%	
igS		0.877	0.784	0.733	0.728	0.722	0.591	0.693	0.856
igTM	config. 1	0.808	0.810	0.707	0.702	0.680	0.474	0.603	0.756
	config. 2	0.819	0.796	0.715	0.707	0.707	0.490	0.618	0.789
TMHMM 2.0		0.900	0.879	0.830	0.827	0.915	0.339	0.636	0.920

With respect to *igTM*, the method we propose in this work improves the sensitivity (Sn of 0.88), and, despite a slightly lower specificity (Sp of 0.78), obtains a better AC (from 0.707 to 0.722). This method also improves the Sov_{δ}^{obs} obtained by *igTM*. This means that our proposal behaves better at the ends of the predicted membrane domains.

Another relevant improvements are the results obtained in the segment-based measures. Thus, a 59.1% of segments were correctly covered by the prediction. There was also obtained a slight better coverage of the membrane domains at 90% of accuracy. Nevertheless, in order to do a fair comparison, these results have to be combined with the Sp measure, because it is possible that this value is due to the presence of false positives at the boundaries of the correctly predicted segments. The improved value of Sov_{δ}^{obs} may show that this is not the case. Nevertheless, the improvement of the Sp value is key for further developments.

5 Conclusions and Future Lines of Work

This paper describes *igS*, an application of Grammatical Inference to the task of transmembrane domain prediction. Grammatical Inference has already been used to tackle this task [5].

The method we propose in this work introduces a preprocessing where the protein is divided into sub-sequences that belong to different domains according to the topology of the protein (inner, outer or transmembrane). We consider the inference of k -TSS languages to obtain an automaton for each of these subsets. We also generate a k -TSS language for the set of label sequences. Our method then combines the automata of each subset with the automaton of labels using a language substitution.

The experimental results of this method outperformed the previous GI approach to the task (*igTM* [5]). The improved accuracy of the method may be attributed to the higher specificity of generated languages, due to the classification of protein sub-sequences introduced in the preprocessing. Nevertheless, this approach is slightly less accurate than the method based on HMM. This may be caused by the need of more data in training phase in GI.

We note that, although applied to a bioinformatic task, this approach may be useful whenever there exists relevant information to label the training sequences (which is usually the case in pattern recognition tasks).

In the future, we plan to combine *igS* together with (an)other prediction method(s). This may allow to raise the specificity of the approach and therefore to improve the results. At present, we are testing other inference algorithms to learn the automata, the use of new labelling information of the sequences [30,31], and larger datasets, by merging the existing ones. The influence of the substitution procedure in the experimental behaviour remains also as future work.

References

1. Searls, D.B.: The language of genes. *Nature* 420, 211–217 (2002)
2. Sakakibara, Y.: Grammatical inference in bioinformatics. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(7), 1051–1062 (2005)
3. Yokomori, T., Kobayashi, S.: Learning local languages and their application to dna sequence analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* 20(10), 1067–1079 (1998)
4. Peris, P., López, D., Campos, M., Sempere, J.M.: Protein motif prediction by grammatical inference. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) *ICGI 2006. LNCS (LNAI)*, vol. 4201, pp. 175–187. Springer, Heidelberg (2006)
5. Peris, P., López, D., Campos, M.: Igtm: an algorithm to predict transmembrane domains and topology in proteins. *BMC-Bioinformatics* 9, 367–378 (2008)
6. Brazma, A., Johansen, I., Vilo, J., Ukkonen, E.: Pattern discovery in biosequences. In: Honavar, V.G., Slutzki, G. (eds.) *ICGI 1998. LNCS (LNAI)*, vol. 1433, pp. 257–270. Springer, Heidelberg (1998)
7. Arimura, H., Wataki, A., Fujino, R., Arikawa, S.: A fast algorithm for discovery optimal string patterns in large databases. In: Richter, M.M., Smith, C.H., Wiehagen, R., Zeugmann, T. (eds.) *ALT 1998. LNCS (LNAI)*, vol. 1501, pp. 247–261. Springer, Heidelberg (1998)
8. Peris, P., López, D., Campos, M.: Localización de genes en el adn mediante inferencia gramatical. In: Universidad de Valencia (ed.) *Proceedings of the XII Congreso de la Sociedad Española de Neurociencia, Universidad de Valencia (2007)* (spanish)
9. Campos, M., López, D., Peris, P.: Incremental multiple sequence alignment. In: Rueda, L., Mery, D., Kittler, J. (eds.) *CIARP 2007. LNCS*, vol. 4756, pp. 604–614. Springer, Heidelberg (2007)
10. Wallin, E., von Heijne, G.: Genome-wide analyses of integral membrane proteins from eubacterial, archaean, and eukaryotic organisms. *Protein Science* 7(4), 1029–1038 (1998)
11. Mitaku, S., Ono, M., Hirokawa, T., Boon-Chieng, S., Sonoyama, M.: Sonoyama. Proportion of membrane proteins in proteomes of 15 single-cell organisms analyzed by the sosui prediction system. *Biophysical Chemistry* 82(2-3), 165–171 (1999)
12. Sugiyama, Y., Polulyakh, N., Shimizu, T.: Identification of transmembrane protein functions by binary topology patterns. *Protein Engineering Design and Selection (PEDS)* 16(7), 479–488 (2003)

13. Sonnhammer, E.L.L., von Heijne, G., Krogh, A.: A hidden markov model for predicting transmembrane helices in protein sequences. In: Glasgow, J.L., Littlejohn, T.G., Major, F., Lathrop, R.H., Sankoff, D., Sensen, C. (eds.) ISMB, pp. 175–182. AAAI, Menlo Park (1998)
14. Tusnády, G.E., Simon, I.: The hmmtop transmembrane topology prediction server. *Bioinformatics* 17(9), 849–850 (2001)
15. Viklund, H., Elofsson, A.: Best alpha-helical transmembrane protein topology predictions are achieved using hidden markov models and evolutionary information. *Protein Science* 13(7), 1908–1917 (2004)
16. Fariselli, P., Casadio, R.: Htp: a neural network-based method for predicting the topology of helical transmembrane domains in proteins. *Computer Applications in the Biosciences* 12(1), 41–48 (1996)
17. Michael Gromiha, M., Ahmad, S., Suwa, M.: Neural network-based prediction of transmembrane α -strand segments in outer membrane proteins. *Journal of Computational Chemistry* 25(5), 762–767 (2004)
18. Pasquier, C., Promponas, V.J., Palaios, G.A., Hamodrakas, J.S., Hamodrakas, S.J.: A novel method for predicting transmembrane segments in proteins based on a statistical analysis of the SwissProt database: the PRED-TMR algorithm. *Protein Eng.* 12(5), 381–385 (1999)
19. Sadvskaya, N.S., Sutormin, R.A., Gelfand, M.S.: Recognition of transmembrane segments in proteins: Review and consistency-based benchmarking of internet servers. *J. Bioinformatics and Computational Biology* 4(5), 1033–1056 (2006)
20. Bagos, P.G., Liakopoulos, T., Hamodrakas, S.J.: Evaluation of methods for predicting the topology of beta-barrel outer membrane proteins and a consensus prediction method. *BMC Bioinformatics* 6, 7 (2005)
21. Punta, M., Forrest, L.R., Bigelow, H., Kernysky, A., Liu, J., Rost, B.: Membrane protein prediction methods. *Methods* 41(4), 460–474 (2007)
22. Tellier, I.: How to split recursive automata. In: Clark, A., Coste, F., Miclet, L. (eds.) ICGI 2008. LNCS (LNAI), vol. 5278, pp. 200–212. Springer, Heidelberg (2008)
23. Berstel, J.: *Transductions and Context-Free Languages*. Teubner Studienbücher, Stuttgart (1979)
24. Vidal, E., García, P., Casacuberta, F.: Local languages, the sucesor method, and a step towards a general methodology for the inference of regular grammars. *IEEE Trans. on PAMI* 9(6), 841–845 (1987)
25. Segarra, E., Hurtado, L.: Construction of Language Models using the Morphic Generator Grammatical Inference (MGGI) Methodology. In: Proc. of Eurospeech, Rhodes (Grecia), pp. 2695–2698 (1997)
26. Grau, S., Segarra, E., Sanchis, E., García, F., Hurtado, L.F.: Incorporating semantic knowledge to the language model in a speech understanding system. In: IV Jornadas en Tecnologia del Habla, pp. 145–148 (2006)
27. Burset, M., Guigo, R.: Evaluation of gene structure prediction programs. *Genomics* 34(3), 353–367 (1996)
28. Mathews, B.W.: Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica Biophysica Acta* 405(2), 442–451 (1975)
29. Rost, B., Sander, C., Schneider, R.: Redefining the goals of protein secondary structure prediction. *J. Mol. Biol.* 235, 13–26 (1994)
30. Reed Murphy, L., Wallqvist, A., Levy, R.M.: Simplified amino acid alphabets for protein fold recognition and implications for folding. *Protein Engineering* 13(3), 149–152 (2000)
31. Li, T., Fan, K., Wang, J., Wang, W.: Reduction of protein sequence complexity by residue grouping. *Protein Engineering* 16(5), 323–330 (2003)