Sequence-Function Protein Analysis Using Annotated Context Free Grammars *

Salvatore Spinella Eva Sciacca
Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, I-10149 Torino, Italy;
{spinella,sciacca}@di.unito.it

Paola Giannini
Dipartimento di Informatica, Università del Piemonte Orientale,
Via Bellini 25/G, 15100 Alessandria, Italy
giannini@mfn.unipmn.it

Identification and understanding of protein function is a fundamental task in the analysis of complex biological systems. Protein structures and functions are encoded in the underlying amino acid sequence. The exact relationship between primary structure of the protein, its three-dimensional structure and its function is one of the fundamental unanswered questions in biology. The mapping of protein sequences with respect to their structure and function may benefit from the analogy that we have in the structuring of languages, particularly in the assignment of meaning to words. Starting from Searls in 1993 [4] the analogy between biology and linguistic has been studied by a growing number of researchers. Thus, language analysis [1] has found applications to biological sequences, using various types of "vocabulary", for example the nucleotides in the case of DNA (e.g. [5]), and the standard 20 amino acids in the case of proteins (e.g. [6]).

We made use of Context-Free Grammars (CFGs) and a protein classification algorithm to develop *Annotated Context Free Grammars* (ACFGs). ACFGs are CFGs in which non terminal symbols are annotated using an n-gram Bayesian classifier. ACFGs are used to analyse the connection between protein chains and protein functions and can be applied to the interpretation and detection of amino acids involved in different functional regions. ACFGs are built using a bottom-up analysis, inspired by LZ77 compression algorithm [7], starting with the input of all protein sequences and attempting to rewrite them backward to the starting symbol. The analysis procedure locates *the most basic motifs* of a given length substituting them with a new non terminal symbol annotated with the most likely domain of the motifs. The most basic motifs are the most frequent motifs occurring in the protein family according to the theory of

*This research is founded by the BioBITs Project (Converging Technologies 2007, area: Biotechnology-ICT)

conservation across protein domains. In fact, it is well known that structural and functional similarities among proteins are frequently conserved in a stretch of few amino acids [2].

Proteins are composed of evolutionarily conserved units called domains, corresponding to subunits of the 3-D structure of a protein, that have distinct molecular function and structure. The sequential order of domains in a protein sequence is known as its protein domain architecture. Architectures are useful for classifying evolutionarily related proteins.

The aim of this work was to build ACFGs for specific families of proteins. To achieve this goal we represent the *primary structure of a protein* as a string $\sigma \in \Sigma^+$ where Σ is an alphabet for the set of 20 amino acids, and D is a set of domain identifiers of an architecture of proteins in a specific family. Starting with a specific family of proteins $\mathfrak{F} \subset \Sigma^+$, we build an ACFG that generates the strings in $\mathfrak{F}$ and whose non terminal symbols are labelled with domains.

Let us first introduce the (grammar) notation needed. Let $\sigma \in \Sigma^+$, with $\sigma[h \dots h+k]$ we denote the subsequence of σ starting at the h-th symbol and ending at the $h+k$-th. An Annotated Context-Free Grammar G is a 5-tuple $G = (V, \Sigma, S, \delta, R)$ where: V is a finite set of non-terminal symbols, (representing different subsequences of amino acids in the protein sequence), Σ is a finite set of terminals, (representing the set of 20 amino acids), S is the start symbol of the grammar, $\delta : \Sigma^+ \to D$ is a function mapping strings (substrings of the initial primary protein structures) into a set of domain identifiers D, and R is a set of productions $A_d \to w$, or $A \to w$ where $A \in V$, $d \in D$, and $w \in (V \cup \Sigma)^+$. As usual with $w \Rightarrow_R w'$ we mean that the string w' is obtained from w by replacing a non terminal-symbol A with v where $A \to v \in R$, and $w \Rightarrow_R^* w'$ is the transitive and reflexive closure of $\Rightarrow_R$.

The algorithm is parametrized by the length of the most frequent pattern that constitutes motifs of $\mathfrak{F}$. The annotated grammar is computed incrementally. At each step i it is generated a production $A_d \to w$, where A is a new non terminal symbol, w is the most frequent pattern of $\mathfrak{F}^{(i)}$ (a rewriting of the strings in $\mathfrak{F}$ in which portion of the strings are replaced by non terminal symbols), and d is a domain. The domain annotation is performed through classification by means of the function $\delta : \Sigma^+ \to D$, such that if $\delta(a_1 \dots a_k) = d$, then for all proteins σ such that $\sigma[h \dots h+k] = a_1 \dots a_k$ then $\sigma[h \dots h+k]$ is the portion of σ corresponding to the domain d.

Algorithm 1 shows the pseudo code of the proposed approach. The function $\texttt{MostFrequentPattern}(\mathfrak{F}^{(i)}, l)$ searches for the most frequent pattern w of length l in $\mathfrak{F}^{(i)}$. The function $\texttt{Substitute}(v, A^{(i)}, w)$ substitutes the non terminal symbol $A^{(i)}$ for the pattern w in the string v. As we can see the substitution is done in all strings of $\mathfrak{F}^{(i)}$, and in the right-hand-side of the production of $R^{(i)}$, so that the language generated by the productions $R^{(i+1)}$ plus the new production is the same as the language generated by $R^{(i)}$. So that, at each step, we produce an additional classification. The algorithm ends, when there are no more sequences $\sigma \in \mathfrak{F}^{(n)}$ of length greater or equal to l, returning the grammar produced.

Let $(V, \Sigma, S, \delta, R)$ be the grammar returned from the algorithm. The j-th

input: FASTA Sequences $\mathfrak{F}$, length l, function δ
$\mathfrak{F}^{(0)} \leftarrow \mathfrak{F} = \{\sigma_1, \ldots, \sigma_{|\mathfrak{F}|}\}$;
$R^{(0)} \leftarrow \{S \rightarrow S_j, S_j \rightarrow \sigma_j \mid 1 \leq j \leq |\mathfrak{F}|\}$;
$V^{(0)} \leftarrow \{S\} \cup \{S_j \mid 1 \leq j \leq |\mathfrak{F}|\}$;
$i \leftarrow 0$;
repeat
  $w \leftarrow \texttt{MostFrequentPattern}(\mathfrak{F}^{(i)}, l)$;
  let $A^{(i)}$ be a fresh non terminal symbol and;
  let $\sigma \in \Sigma^+$ be such that $w \Rightarrow^*_{R^{(i)}} \sigma$;
  $R^{(i+1)} \leftarrow \{A^{(i)}_{\delta(\sigma)} \rightarrow w\} \cup \{A_d \rightarrow \texttt{Substitute}(v, A^{(i)}, w) | A_d \rightarrow v \in R^{(i)}\} \cup \{S_j \rightarrow \texttt{Substitute}(v, A^{(i)}, w) | S_j \rightarrow v \in R^{(i)}\}$;
  $\mathfrak{F}^{(i+1)} \leftarrow \{\texttt{Substitute}(v, A^{(i)}, w) \mid v \in \mathfrak{F}^{(i)}\}$;
  $V^{(i+1)} \leftarrow V^{(i)} \cup \{A^{(i)}\}$;
  $i \leftarrow i + 1$;
until $\forall \sigma \in \mathfrak{F}^{(i)}(|\sigma| < l)$;
return $(V^{(i)}, \Sigma, S, \delta, R^{(i)})$

Algorithm 1: Annotated Context-Free Grammar pseudo-code

protein primary structure σ_j is derived from the symbol S_j, that is $S_j \Rightarrow^*_R \sigma_j$. Moreover, its derivation tree has the following properties: (a) the node corresponding to S in not annotated and has $|\mathfrak{F}|$ children; (b) the node corresponding to S_j in not annotated and has at most $l-1$ children; (c) all internal nodes (corresponding to non-terminal symbols except for the symbols S and S_j) are annotated and have exactly l children; and (d) leaves correspond to the amino acids. The function $\delta : \Sigma^+ \rightarrow D$ of the ACFG that maps a set of sub-sequences of proteins $\alpha = \sigma[i \ldots i+k] = a_1 \ldots a_k$ to one of a set of $|D|$ pre-defined domains $D = \{d_1, d_2, \ldots, d_{|D|}\}$ is produced by a n-gram Bayesian text classifier. A supervised learning framework provided by the classifier is used to train the text classifier on a set of labelled training examples $(\mathfrak{D}^i, d_i) : i = 1, \ldots, |D|$. Here $\mathfrak{D}^i$ denotes the i-th training domain model containing a set of protein subsequences of a family belonging to a specific domain and d_i is the corresponding domain. The principle for using a n-gram Bayesian text classifier is to determine the category/domain that makes a given amino acid sequence most likely to have been generated by a domain model $\mathfrak{D}$. Thus, we train a separate language model for each domain, and classify a protein sub-sequence by evaluating its likelihood under each domain, choosing the category according to it.

Our methodology was applied to a class of Antimicrobial Peptides (AmPs): the Frog antimicrobial peptides family. This family consists of the major classes of antimicrobial peptides secreted from the skin of frogs that protect the frogs against invading microbes. They are typically 10-50 amino acids long and are derived from proteolytic cleavage of larger precursors. Major classes of peptides such esculentin, gaegurin, brevinin, rugosin and ranatuerin are included in this family [3]. According to the PFAM database, the domain architecture in which

the 68% of proteins of this domain is found implies a presence of another domain located in the first half of the sequence: the Brevenin family. This family contains a number of defence peptides secreted from the skin of amphibians.

We built the ACFG for the frog AMP family, G_{FROG}, considering the two domain identifiers $D = \{\text{AMP}, \text{BREVENIN}\}$. We set the length l of the most frequent motifs to 3. Finally, the classifier for the function mapping the subsequences of amino acids into the set of domain identifiers was trained. A compact notation for the derivation annotated tree is shown above where $\{\ldots\}$ represents a "BREVENIN" node while $(\ldots)$ represents an "AMP" node:

{ { { { (M F T) M (K K S)} L L} F { L F { F (L G T) I}}} { { ((S L S) L C) { E E E} R} { S { { A D E} D D} { G G { E M T}}} ({ { E E E} V { K R G}} (I L { D T L}) ((K Q (F { A K G} V)) G (K D L)))} { V K G}} ((A A Q) ((G V L) S T) (V S ((C K L) (A K T) C)))

ACFG allow us to analyse the functional conserved regions, where they are located and how they are related each other inside the primary structure of the protein. In this case we notice that some "AMP" nodes are contained by "BREVENIN" nodes, this result lead us to suppose that a common ancestor belonging to the domain "AMP" conserved that motifs to evolve the "BREVENIN" domain.

Through this case study, our approach pointed out some fundamental aspects regarding the relationship between sequence and functional domains of proteins and how protein domain motifs are preserved by natural evolution in the amino acid sequences. The showed results are comparable with the state of art of sequence alignment based frameworks (such as the PFAM tool) and give a structured information on the protein sequences more powerful than the current annotation systems producing regular grammars. A framework based on ACFGs structuring suggests also a methodology for synthesis of new proteins in order to design undiscovered multi-functional proteins.

For future works, we intend to further our research optimizing our grammatical structures. We are also planning to implement grammatical inference (GI) algorithms for our ACFG in order to obtain grammatical models to check for the presence or absence of a domain in a protein sequence.

References

[1] M. Davis, R. Sigal, and E.J. Weyuker, *Computability, complexity, and languages: fundamentals of theoretical computer science*, Morgan Kaufmann Pub, 1994.

[2] J.M. Otaki, S. Ienaka, T. Gotoh, and H. Yamamoto, *Availability of short amino acid sequences in proteins*, Protein Science: A Publication of the Protein Society **14** (2005), no. 3, 617.

[3] A.C. Rinaldi, *Antimicrobial peptides from amphibian skin: an expanding scenario: Commentary*, Current opinion in chemical biology **6** (2002), no. 6, 799–804.

[4] D.B. Searls, *The computational linguistics of biological sequences*, Artificial Intelligence and Molecular Biology (1993), 47–120.

[5] Searls, D.B., *The language of genes*, Nature **420** (2002), no. 6912, 211–217.

[6] J. Waldispühl and J.M. Steyaert, *Modeling and predicting all-α transmembrane proteins including helix-helix pairing*, Theoretical Computer Science **335** (2005), no. 1, 67–92.

[7] J. Ziv and A. Lempel, *A universal algorithm for sequential data compression*, IEEE transactions on Information Theory **23** (1977), no. 3, 337–343.