IOSR Journal of Engineering (IOSRJEN)
e-ISSN: 2250-3021, **p**-ISSN: 2278-8719
Vol. 3, Issue 12 (December. 2013), ||V2|| PP 46-51

On attribute context free grammar derived from Brix2 database to describe protein's fragment assembly

Prof. Dr. Nabeel Hashem Kaghed[1], Mohannad M. J. Al-Yasiry[2]
Ministry of Higher Education and Scientific Research, Iraq
University of Babylon College of Information Technology

Abstract: - *In this work the scenario of protein's fragment assembly will be adopted. So, it will for each step the basic rules that be describe how the protein's fragment assembly from Building blocks to secondary structure elements is founded by data mining method, these rules were given in format of formal language grammar in order to increase the ability to understand these steps (semantic) and facilitate handling it in terms of (addition, deletion, and modification), and be a solid basis to any current or future method to deal with folding proteins in dry lab(computational methods). In addition, it gives signs to indicate the progress in stages series for predicting a target protein structure is meaningful or not.*

Keywords: - *ACF-Grammar, Protein's Fragment assembly*

I. INTRODUCTION

In the field of proteins, there were several ways to predict a tertiary structure. One of the most important of these methods were computational methods that depend mainly on the method of modeling protein. For modeling proteins, there were several ways, like mathematical-based models and Physics-based models. In another hand, the current protein's modeling methods has many disadvantages: first, it's hard to fully understand how it derived by bioinformatics researchers from the basic physical chemistry rules, therefore it's a complex process to modify or enhance one by computer science specialist. Second, protein's modeling methods did not help with answer the question of how and why a protein adopts its specific structure or what its role in describing protein folding. Third, protein's modeling methods must applied on all input elements (conformations space) without considering the time and storage complexity, in addition, there is no indicator about the significant for each element.

The process of searching and assembling clues to the structure of a target sequence by finding similarities to known structures in different categories of biological databases could help in overcome some of complex models disadvantages. The molecular structure databases one of these biological databases, and depending on the kind of data included, two types of databases can be distinguished[1]. Primary databases contain primary sequence / structure information (protein) and accompanying annotation information regarding function, bibliographies, cross-references to other databases, …etc, examples (PDB,CATH,SCOP). Secondary databases, however, summarize the results from analyses of primary protein sequence / structure databases, examples (Brix2, Astral). Furthermore, the process of describing a complex problem as a hierarchical levels then adding specific knowledge for each level often led to better problem understanding and more efficient solutions. Since the formal grammar can fulfill these requirements due to some qualities like precision and understandability then a specific type of grammar can be used to map the process of protein folding based on "zipping and assembly hypothesis" local structuring happens first at independent sites along the chain, then those structures either grow (zip) or coalescence (assemble) with other structures"[2] (wet lab) to computer environment (dry lab).

From all above we can assume the following hypothesis:

1. Since there were some levels of protein's fragment structures, so we can assume that it is possible to identify a number of key features for each level can be put on the basis of which the rules are extracted from specific databases of known protein structure such as "Brix2".
2. These features and rules can be mapped into a formal language grammars provide us with a significant contribution to facilitate the understanding of the proteins folding process in addition to the possibility of combined these grammar with energy functions to reduce the time and storage complexities, where it will determine any of the elements are the most significant, depending on the appropriate level of grammars and then examined using an energy function.
3. The best type of grammars that can be used to represent the features and rules of protein's fragment is ACF-Grammar (Attribute Context-Free Grammar).

II. THE PROPOSED SYSTEM

The proposed system contain three stages based on ACF-Grammar requirements (Attributes, CF-Grammar, and Rules). First stage consider with protein fragment's Attributes that must founded. Second stage consider with protein fragment's CF-Grammar. Third stage consider with protein fragment's ACF-Grammar. The proposed system was implemented in C#. All analysis and control of the programs was achieved using sets of scripts written in SQL and driven from a relational database of the BriX2 data implemented using the freely available object-relational database MySQL workbench 6.0 (http://www.MySQL.com). Analysis was performed on (a core 2 due processor and 3 GB RAM) machine running windows vista.

The following algorithm describe the main steps of proposed system :

Input	*BriX2.*
Output	*Attributes tables and ACF-Grammar for protein's fragment assembly*
ALGORITHM :	
Step 1	*Determining the rules that govern the process of element's association with each other.*
Step 2	*Split elements into classes according to certain criteria.*
Step 3	*Find central element for each class.*
Step 4	*Identify previous and subsequent elements to all central fragment then extract the important attributes that will help determine the importance of resultant Building Block (BB).*
Step 5	*Write CF-Grammar that describe these steps .*
Step 6	*Generate ACF-Grammar that describe the process using above (CF-Grammar, Attributes, and Rules).*
Step 7	*END.*

The block diagram of proposed system as follow:

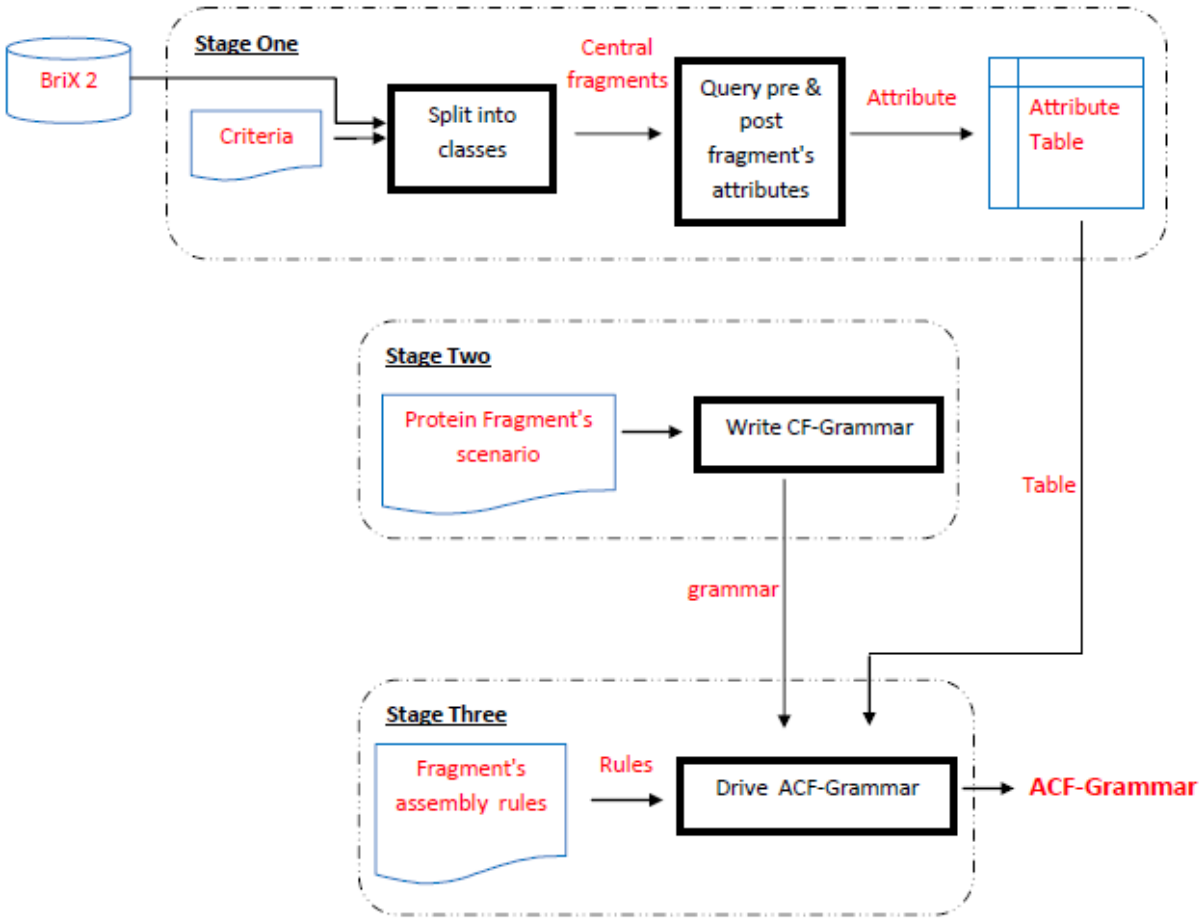


Figure [1]. The block diagram of proposed system

2.1 BriX 2 :[3]

It's a database of protein building blocks for structural analysis, modeling and design. It directed to identifying recurrent protein fragments, which are frequently reused as building blocks to construct proteins that were till now thought to be unrelated.

BriX contains two levels: the class level, and the fragment level. Classes can be sorted and filtered on (1) class size, (2) fragment length (from 4 to 14 residues), (3) clustering threshold describing the compactness of the classes, (4) minimum and maximum percentage of helix, loop, sheet and turn content and (5) regular expressions of the amino acid sequence and secondary structure as determined by DSSP (Dictionary of Protein Secondary Structure).

The BriX database contains fragments from over 7000 non-homologous proteins from the ASTRAL40 (set of 7290 proteins sharing <40% of sequence homology) and WHAT IF (set of 1259 non-redundant proteins) collection, segmented in lengths from 4 to 14 residues and clustered according to backbone similarity with a hierarchical clustering algorithm, summing up to a content of 2 million fragments per length. It available online (http://brix.crg.es) and can be accessed through a user-friendly web-interface or can be downloaded in the form of SQL file.

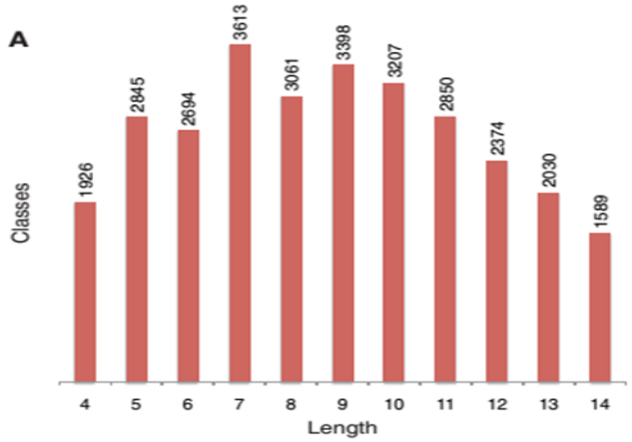
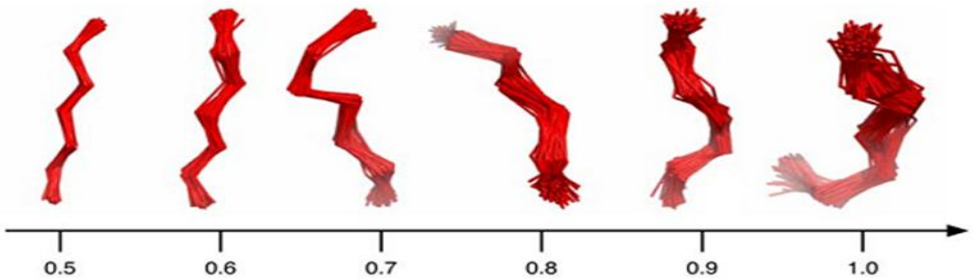


Figure [2]. Number of BriX classes for lowest class thresholds per length. A peak in the number of classes can be observed at fragment length 7 and class threshold 0.5 Å

2.2 Split into classes :

Depend on fragments' length [from 4 -14 residues] a classification criteria was used . Then for each fragment length, a centroid fragment is fetched based on RMSD thresholds from 0.6 to 1 Angstrom that describing the compactness of the classes[4]. Then saved in a table of central fragment for each length.



FFigure [3]. Thresholds (Å)

2.3 Query pre & post fragment's attributes :

Depend on central fragment's table for each length, a pre & post fragments are fetched. Then extract specific attributes like (FragmentID, aa_Sequence, Structure, Nex-f-ID, Nex-seq, Nex-stru, Pre-f-ID, Pre-seq, Pre-stru). All that done by using sub-query statements (it is a nested "select" statement in SQL).

2.4 Write a CF-Grammar :

To illustrate of how the CF-Grammar can be represent the protein's fragment assembly, a definition to the terminals and non-terminals in this CF-Grammar is needed. After that, a general template to represent protein's fragment assembly by the CF-Grammar is introduced based on a sound scenario.

The protein's fragment assembly can be classified into levels that most effectively indicate their stages in protein zipping and assembly mechanism. A protein's fragment can be part of multiple Building Block in these levels. Each class has its name indicating the general category; the set of classes will be denoted by Table[1]. These classes represent the non-terminals in the CF-Grammar whereas the segments represent the terminals of the CF-Grammar.

Classes	Abbreviations
Secondary Structure	Sec-Stru
Next Building Block	Nex-BBs
Previous Building Block	Pre-BBs
Building Block	BB

Table [1]. Abbreviations of some classes stages in protein's fragment assembly

The most two important key points in the protein's fragment assembly scenario are (orientation, and secondary structure information) of previous and subsequent to the central fragment. Based on the idea in figure[4] below that depict how to find the correct orientations and the expected secondary structure information (type, length) for each centroid fragment.

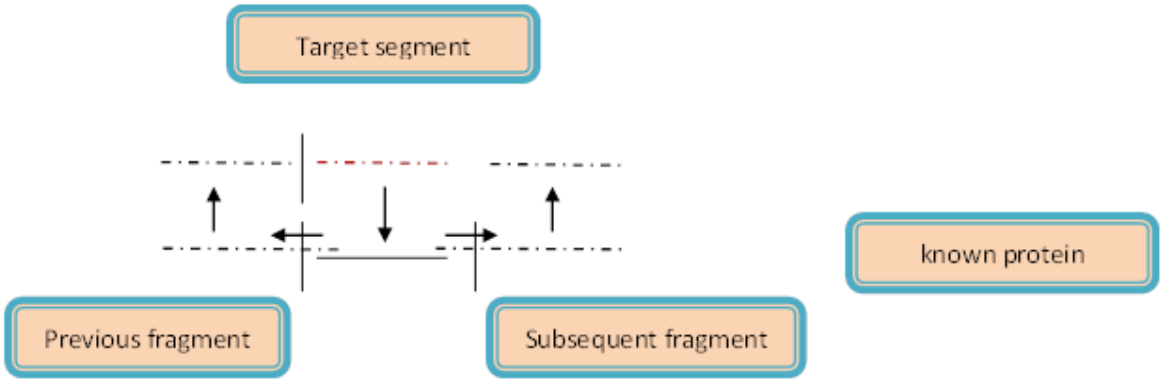


Figure [4]. The basic idea of protein's fragment assembly

The correct orientations can be preserved by find the previous fragment with last three amino acids identical to the first three amino acids in the center fragment and the subsequent fragment with first three amino acids identical to the last three amino acids in the center fragment.

The expected secondary structure can be found by select a previous and subsequent fragments with same length of center fragment then find how many amino acids needed in each direction to complete the current secondary structure.

2.5 Derive the ACF-Grammar :

To illustrate of how the ACF-Grammar can be derived to represent the protein's fragment assembly, An Attribute context-free grammar ACF-Grammar consists of three elements, a CF-Grammar, a finite set of attributes Att, and a finite set of semantic rules R[5]. The set of attributes Att includes knowledge about grammar symbols which produced in stage one. A set of terminal and non-terminal symbols in CF-Grammar production "p" represent the protein's fragment assembly scenario that produced in stage two. A finite set of semantic rules R is associated with each production "p". in the proposed system, a two types for these semantic rules was implemented: copy rules and check rules. The copy rule, as its name, copies the attribute value from Xi to Yj , where Xi, Yj belong to Non-terminal symbols. The check rule checks for some conditions to be satisfied. Finally, the ACF-Grammar that represent the protein's fragment assembly can be put in table that has three fields (levels, productions, and semantic rules).

III. EMPIRICAL RESULTS DISCUSSION

In this section the attribute table, CF-Grammar, and ACF-Grammar that were obtained from the proposed system must discusses. The table contain a number of attributes like (FragmentID, aaSequence,

Structure, Nex-f-ID, Nex-seq, Nex-stru, Pre-f-ID, Pre-seq, Pre-stru). While, there are 10 different lengths of segment (4 -14), so for each length there is a table hold above attributes. After analysis each table separately, a results about the count of all centroid fragments, with and without next fragment, and with and without previous fragment are gained. Example, segment length "14" contain 5488 centroid, 33 centroid that have no next fragments, 25 centroid that have no previous fragments, 9 centroid without next & previous fragments, and 5436 centroid that have next & previous fragments.

The above statistics can be used to evaluate the results from applying the proposed system on unknown protein structure to find out the expected fragments semantic with the same fragment residue number.

To understand Grammars G =(Σ,V,S,P), an introduction to few of their essentials terms was needed. An alphabet (let Σ be an alphabet) is a finite nonempty set of symbols. Symbols are assumed to be indivisible. "A" string over an alphabet Σ is a finite sequence of symbols of Σ. The number of symbols in a string x is called its length, denoted by |x|. It is convenient to introduce a notation ε for the empty string, which contains no symbols at all. The set of all strings over an alphabet Σ is denoted by Σ*. A sentential form is any string of terminals and non-terminals, i.e., a string over Σ U V.

Four classes of grammars are obtained by gradually increasing the restrictions that the production rules have to obey. This classification is due to Chomsky [6,7] and is called the Chomsky hierarchy. The following explanation [8] and graph briefly describes all grammar types.

Let G =(Σ,V,S,P) be a grammar.

1. G is called a Type-0 grammar or an unrestricted grammar.
2. G is a Type-1 or context-sensitive grammar if each production A → a in "P" satisfies |A| ≤ |a|. By "special dispensation," we also allow a Type-1 grammar to have the production S → ε , provided "S" does not appear on the right-hand side of any production.
3. G is a Type-2 or context-free grammar if each production A → a in "P" satisfies | A | = 1; i.e., "A" is a single non-terminal.
4. G is a Type-3 or right-linear or regular grammar if each production has one of the following three forms:

A → cB,
A → c,
A → ε
where "A", "B" are non-terminals (with B = A allowed) and "c" is a terminal

The language generated by a Type-i grammar is called a Type-i language, i = 0, 1, 2, 3. For each i = 0, 1, 2, the class of Type-i languages properly contains the class of Type-(i+1) languages 8 and this can clearly be seen in figure [5] below:

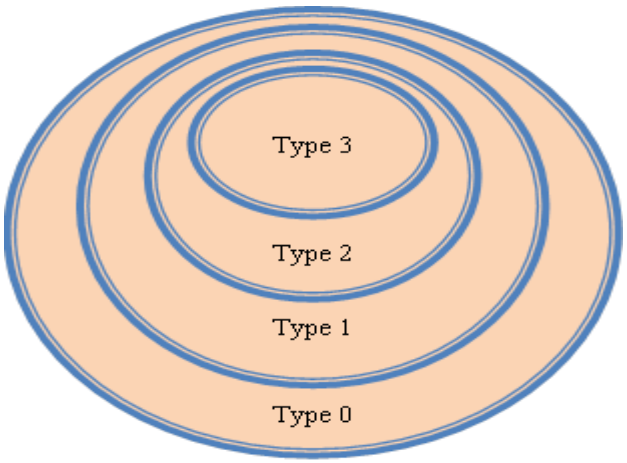


Figure [5]. Hierarchies of grammars

After this small introduction a general template to represent protein's fragment assembly by the CF-Grammar is introduced.

CF-Grammar of protein's fragment assembly is:

Sec-Stru → Pre-BBs BB Nex-BBs
Pre-BBs → Pre-BBs BB | ε
Nex-BBs → BB Nex-BBs | ε
BB → seg_4 | seg_5 | . . . seg_14

*Where seg_4 : It a fragment with four residues length.
And seg_5 : It a fragment with five residues length.
And so on.

Finally, the ACF-Grammar of protein's fragment assembly is:

Levels	Productions	Semantic Rules
1	Sec-Stru → Pre-BBs BB Nex-BBs	Sec-Stru.topology IN topology table
2	Pre-BBs → Pre-BBs BB | ε	Pre-BBs. SubString(length-3,end) = BB. Sub String(0,3) And Pre-BBs.secondary structure IN Att. Table
3	Nex-BBs → BB Nex-BBs | ε	Nex-BBs.SubString(0,3) = BB. SubString(length-3,end) And Nex-BBs.secondary structure IN Att. Table
4	BB → seg_4 | seg_5 | . . . seg_14	

Table [2]. Describe the ACF-Grammar of protein's fragment assembly

While ACF-Grammar = (CF-Grammar, Att, R), A finite set of attributes Att(X) is associated with each symbol X ϵ N. The set Att(X) is partitioned into two disjoints subsets, the inherited attributes and the synthesized attributes [9]. The synthesized attributes move the data flow upwards and the inherited attributes move the data flow downwards in the parse tree during the attribute evaluation process. In our model, we used the synthesized attributes.

The production p ϵ P, p : Yj → X1 . . . Xm (m ≥ 1), has an attribute occurrence Xi.a if "a" ϵ Att(Xi), 1 ≥ i ≥ m. A finite set of semantic rules is associated with each production p.

IV. CONCLUSIONS

Different from most other protein's model describing methods, our proposed system was treated the protein's fragment as biological language building blocks. It analyzed of a well-known protein database "BriX2" to got an appropriate attributes for each protein's fragment structure to be used as an assessment method. Then it convert the protein's fragment assembly scenario to formal grammar (CF-Grammar). Finally, by an appropriate attributes, some semantic rules, and CF-Grammar, a convenient ACF-Grammar were derived.

Form the results and discussions above, we can said that the hypothesis were correct and research objective accomplished. Also, someone can said that the ACF-Grammar it a good describing method and can be implemented on complex biological systems like protein folding process. In addition, it open new trends in structural bioinformatics filed to tackles hard problems.1

There are several ways in which the above method exposed in this research can be extended in the future; these ways are as follows:

1. Build from scratch a new ACF-Grammar to protein's fragment assembly based on other scenarios from different protein folding theories, then select the best according to simplicity and efficiency.
2. Suggest a similar idea for using ACF-Grammar to describe the protein folding scenario as a whole (for all levels) .

REFERENCES

[1] P.M. Selzer, R.J. Marhöfer, A. Rohwer, "Applied bioinformatics : an introduction" , Springer-Verlag Berlin Heidelberg, 2008.
[2] Ozkan SB, Wu GHA, Chodera JD, Dill KA. "Protein folding by zipping and assembly". Proc Natl Acad Sci USA 2007;104:11987–92.
[3] Vanhee P., Verschueren E. et al., "BriX: a database of protein building blocks for structural analysis, modeling and design", NAR Database Issue 2011.
[4] Baeten L. et al., "Reconstruction of Protein Backbones from the BriX Collection of Canonical Protein Fragments" PLOS Computational Biology, 2010.
[5] Wouter Swierstra,"Why Attribute Grammars Matter", The Monad.Reader Issue Four,2011.
[6] N. Chomsky. Three Models for the Description of Language. IRE Trans. On Information Theory. 1956, 2(2):113–124.
[7] N. Chomsky. Formal Properties of Grammars. In Handbook of Mathematical Psychology. 1963, 2:323–418.
[8] M. J. Atallah, S. Fox., "Algorithms and Theory of Computation Handbook", 1st edn. CRC Press, Inc., 1998:1266.
[9] D. E. Knuth, "Semantics of Context-free Languages. Theory of Computing Systems", 1968, 2(2):127–145.