

1 **Regular Architecture (RegArch): A standard expression** 2 **language for describing protein architectures**

3
4
5 Davi R. Ortega^{1,*} and Grant J. Jensen^{1,2}

6
7 1) Division of Biology and Biological Engineering, California Institute of Technology, 1200 E.
8 California Blvd., Pasadena, CA 91125, USA.

9 2) Howard Hughes Medical Institute, 1200 E. California Blvd., Pasadena, CA 91125, USA.

10
11 * Correspondence: ortegad@caltech.edu

12 13 **Abstract**

14 Domain architecture – the arrangement of features in a protein – exhibits syntactic patterns
15 similar to the grammar of a language. This feature enables pattern mining for protein function
16 prediction, comparative genomics, and studies of molecular evolution and complexity. To
17 facilitate such work, here we propose Regular Architecture (RegArch), an expression language to
18 describe syntactic patterns in protein architectures. Like the well-known Regular Expressions for
19 text, RegArchs codify positional and non-positional patterns of elements into nested JSON
20 objects. We describe the standard and provide a reference implementation in JavaScript to parse
21 RegArchs and match annotated proteins.

22 **Introduction**

23 Protein sequences can be compared to sentences in a natural language: each amino acid
24 corresponding to a letter, and domains to words. Recently it was shown that, just as with
25 language, evolutionary pressures on proteins' biological roles give rise to a grammar structure in
26 the protein universe (Yu *et al.*, 2019). In other words, only a subset of all possible arrangements
27 of words (domains) form meaningful (useful) sentences (architectures). In the context of natural
28 languages, grammar structures can be described by syntactic patterns (Chomsky, 1957), which
29 can be used to mine large volumes of text for specific information. These syntactic patterns in
30 regular languages are commonly codified as Regular Expressions (Kleene, 1956). We propose an
31 analogous standard for describing syntactic patterns in proteins, enabling similar information
32 mining. We call this standard Regular Architecture (RegArch). Here, we briefly describe the
33 standard and give an example of its use.

35 **Concept**

36 The goal of RegArch is to standardize flexible expressions that can be used to define, and
37 therefore search, complex patterns of protein architecture. By architecture, we mean the
38 arrangement of functional features/domains in a protein. Standard BLAST searches reveal all
39 proteins that contain a particular domain, but RegArch allows one to search for proteins that
40 contain multiple domains, in particular orders. RegArch does not identify domains in sequences,
41 it just searches already-annotated sequences for specific domain patterns. Thus the protein
42 sequences have to have already been annotated, for instance by bioinformatics resources such as
43 PFAM (which identifies PFAM domains (Finn *et al.*, 2014)), DAS (which identifies
44 transmembrane regions (Cserzo *et al.*, 2004)), SignalP (which identifies signal peptides

45 (Armenteros *et al.*, 2019)), or some other method (Ulrich and Zhulin, 2014; UniProt Consortium,
46 2018).

47

48 Each RegArch expression is built of two types of patterns: positional and non-positional.

49 Positional patterns match features that occur in a specific order. This order can occur anywhere

50 in the sequence. As for Regular Expressions, RegArchs define the beginning and end of the

51 protein sequence, so a user can also specify that the sequence must start and/or end with

52 positional features. RegArch syntax contains a wild card, so a user can specify a pattern

53 consisting of any combination of defined and undefined (i.e. any domain in the PFAM database)

54 features. Finally, patterns can specify more than one requirement for a particular position,

55 allowing a user, for instance, to exclude specific domains. Non-positional patterns do not require

56 elements to appear in order, therefore allowing a user to search for a feature anywhere in the

57 protein sequence, or to require that a feature be absent. Multiple positional and non-positional

58 patterns can be combined in a single, intricate RegArch.

59

60 See the Supplementary Material for details of how to define, write and organize RegArch

61 expressions. We have also designed a reference implementation in JavaScript to search for

62 annotated proteins that match RegArchs. Details of this implementation can be found in the

63 Supplementary Material.

64

65 **Usage example**

66 Figure 1 illustrates an application of RegArchs to bacterial chemoreceptors, proteins that

67 modulate signaling networks involved in the control of various cellular functions including

68 motility (Porter *et al.*, 2011). In the PFAM database, bacterial chemoreceptors are characterized
69 by an MCPsignal protein domain (Zhulin, 2001). Therefore, to find all chemoreceptors, we
70 would simply define a RegArch consisting of a single non-positional pattern specifying the
71 presence of one or more MCPsignal domains (Fig 1A). Of course this could also be
72 accomplished with a simple BLAST search or PSI-BLAST. This may be perfectly fine for
73 certain purposes, but because of gene duplication, domain swapping, and other evolutionary
74 processes, this strategy is inefficient to find chemoreceptors with more specific domain
75 architecture requirements. RegArch allows more detailed and specific patterns to be searched for
76 as well. Chemoreceptors can be classified on the basis of their membrane topology (Zhulin,
77 2001), for instance. To select chemoreceptors of topological class I (crosses membrane twice
78 with both termini in the cytoplasm and a periplasmic domain), we would define a RegArch with
79 a positional pattern starting with a wild-card protein domain sandwiched by two transmembrane
80 regions (TMs), followed anywhere in the protein by an MCPsignal PFAM domain (Fig 1B).
81 Now imagine that we want to search for a very specific pattern: an initial TM, followed by any
82 PFAM domain except Cache_1, followed by another TM, any number of PFAM domains, and
83 ending with an MCPsignal PFAM domain. In addition, we want to exclude any protein with a
84 PAS domain anywhere in the sequence. The resulting RegArch is shown in Fig 1C. The result of
85 an example search for the RegArchs just described is shown in Fig 1D.

86

87 **Outlook**

88 We expect RegArchs to improve automatic annotation of genomes, providing highly specific and
89 efficient filters to identify multi-domain proteins with similar roles in biological pathways.
90 RegArchs may also help reveal patterns in the grammar of protein architecture in different

91 genomes, informing studies of molecular evolution. The quality of a dataset compiled by
92 RegArchs strongly depends on the accuracy and completeness of underlying protein feature
93 annotation. We therefore suggest tuning specificity, for instance by using highly specific patterns
94 to build an initial high-confidence dataset that can later be used to search for related sequences.
95 As the quality of computational predictions of protein features increases, so will the power of
96 RegArch search algorithms.

97

98 **Acknowledgments**

99 The authors would like to thank the developers of MiST3: Luke Ulrich, Vadim Gumerov, and
100 Ogun Adebali for helpful suggestions and comments on the manuscript. We also would like to
101 thank Igor Zhulin for discussions on protein domain architectures that led to the idea of Regular
102 Architecture. We also thank Dr. Catherine M. Oikonomou for helpful discussion and suggestions
103 on the manuscript. This work was made possible through the support of the National Institutes of
104 Health (grant R35 GM122588 to G.J.J.) and the John Templeton Foundation as part of the
105 Boundaries of Life Initiative (grants 51250 & 60973 to G.J.J.).

106

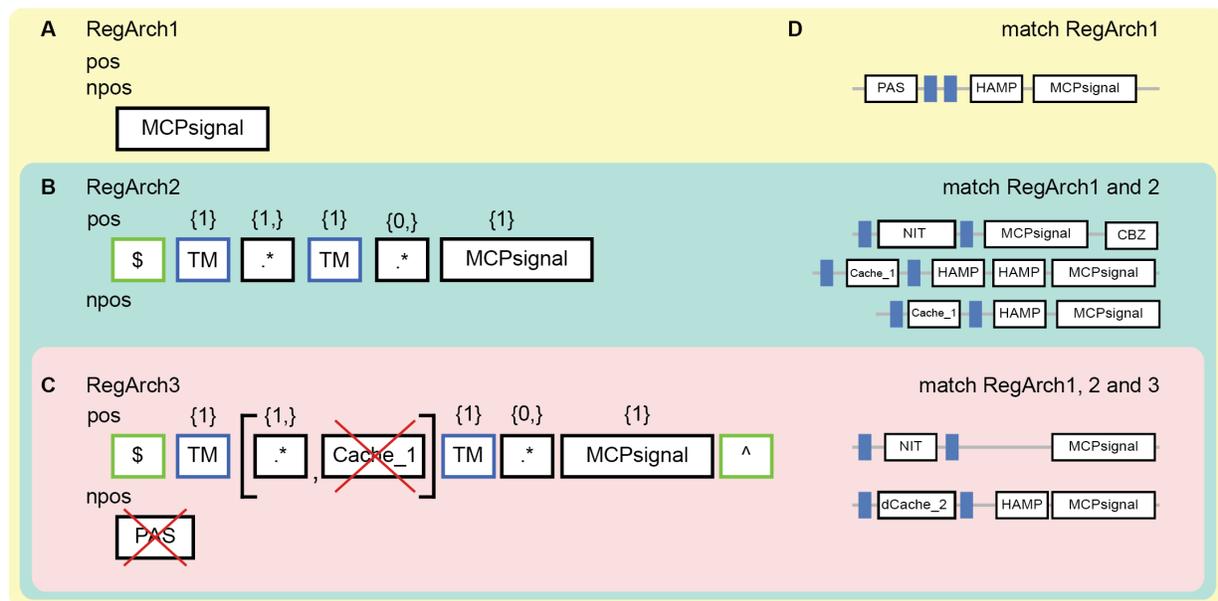
107 **References**

- 108 Armenteros,J.J.A. *et al.* (2019) SignalP 5.0 improves signal peptide predictions using deep
109 neural networks. *Nature Biotechnology*, **37**, 420.
- 110 Chomsky,N. (1957) Logical structures in language. *American Documentation*, **8**, 284–291.
- 111 Cserzo,M. *et al.* (2004) TM or not TM: transmembrane protein prediction with low false positive
112 rate using DAS-TMfilter. *Bioinformatics*, **20**, 136–137.
- 113 Finn,R.D. *et al.* (2014) Pfam: the protein families database. *Nucleic Acids Res*, **42**, D222–D230.
- 114 Kleene,S.C. (1956) Representation of Events in Nerve Nets and Finite Automata. In, *Automata*
115 *Studies. (AM-34)*. Princeton University Press, Princeton.
- 116 Porter,S.L. *et al.* (2011) Signal processing in complex chemotaxis pathways. *Nat Rev Micro*, **9**,
117 153–165.
- 118 Ulrich,L.E. and Zhulin,I.B. (2014) SeqDepot: streamlined database of biological sequences and
119 precomputed features. *Bioinformatics*, **30**, 295–297.

120 UniProt Consortium, T. (2018) UniProt: the universal protein knowledgebase. *Nucleic Acids Res*,
 121 **46**, 2699–2699.
 122 Yu, L. *et al.* (2019) Grammar of protein domain architectures. *PNAS*, **116**, 3636–3645.
 123 Zhulin, I. B. (2001) The superfamily of chemotaxis transducers: From physiology to genomics
 124 and back. In, *Advances in Microbial Physiology*. Academic Press, pp. 157–198.
 125

126 **Figures**

127 Figure 1



128
 129 **Figure 1: Example use of RegArchs to identify bacterial chemoreceptor proteins. (A-C)**
 130 Three RegArchs of increasing stringency, described in the text, are shown. Note that RegArch1
 131 contains only a non-positional pattern (npos), RegArch2 only a positional pattern (pos), and
 132 RegArch3 both. RegArch-specific features are denoted by a green border, PFAM features with
 133 black, and DAS features with blue. The symbols “\$”, “^” and “.*” represent “start”, “end” and
 134 “any” respectively. D) RegArchs were used to search a set of annotated protein sequences with
 135 PFAM domains in white boxes with black borders and DAS predictions of TMs in blue. A
 136 sample of the protein feature architecture that matches the RegArchs is shown.