Reasoning about protein topology using the logic programming language PROLOG

C J Rawlings*, W R Taylor†, J Nyakairu*†, J Fox* and M J E Sternberg†

*Biomedical Computing Unit, Imperial Cancer Research Fund, PO Box 123, Lincoln's Inn Fields, London WC2A 3PX, UK
†Laboratory of Molecular Biology, Department of Crystallography, Birkbeck College, Malet Street, London WC1E 7HX, UK

The logic programming language PROLOG was used to represent and reason about the topology of protein structures. PROLOG descriptions of the relative positions of protein secondary structural features (protein topology) were generated from information in the Brookhaven databank. β-structural motifs (hairpin, meander, Greek key and jelly roll) were then defined using PROLOG rules. The PROLOG program was able to infer the presence of these structures in the PROLOG representation of the protein. A key feature of this approach is the facility to reason about complex topological motifs in protein structure in terms of primitive spatial relationships. Through the ability to represent logical reasoning, PROLOG could be used to develop flexible, high-level methods for controlling a graphics facility or front ends to more conventional databases.

Keywords: proteins + chemical structure, topology, PROLOG, logic programming language, topological motifs

received 2 September 1985, accepted 17 September 1985

INTRODUCTION

The 3D structures of more than 150 proteins have been determined by X-ray crystallography. In most proteins, there are regular secondary structures: β-strands (which hydrogen bond to form a β-sheet) and/or α-helices. The relative positions of the secondary structures is called the protein topology. Over the last ten years, many topological motifs have been observed in proteins. For example, in proteins with β-sheets, four β-strands can be arranged so as to form a pattern known as a Greek key because of its resemblance to decorations on Greek vases[1,2].

An understanding of protein topology is central to various studies of protein conformation. The observation of similar topologies between two protein domains suggests that the protein domains might have evolved by divergence from a common ancestor. Topological motifs impose constraints on allowed conformations, which are exploited in algorithms to predict protein structure from their amino-acid sequence[3,4]. In addition, the presence of some topological motifs is best explained in terms of the pathway of protein folding. Identification of motifs thus provides clues to the nature of some protein-folding pathways.

The majority of work on the analysis of topological features has been performed by visual inspection of diagrams of structures by experts[1,3]. However, the number of known structures is increasing, and it is becoming harder to maintain a comprehensive list of topological motifs. Furthermore, with the widespread use of molecular graphics, a large community of workers is examining protein structure and require information about topological features. Topological motifs may also be important structural units that can be exploited in the design and engineering of synthetic proteins.

Conventional programming languages such as FORTRAN do not easily describe the complexity of protein topology. This paper reports the use of the logic programming language PROLOG to represent and consider the topology of β-sheet proteins. (A preliminary account of this work has been published as a conference abstract[5].) In outline, a PROLOG database of simple facts about the topology of protein structure, such as strand adjacency and relative orientation, is obtained from the Brookhaven protein structure databank[6]. PROLOG rules are derived to represent topological motifs such as Greek keys. Additional reasoning rules are introduced to relate the rules describing motifs to the stored facts. Thus, a database of protein topology is established that can be queried to establish the presence of a particular topological motif in a protein. The results of the query could be used either to collect the polypeptide sequence that comprises the motif, or as input to a graphics system capable of displaying or highlighting the motif.

PROLOG

An introduction to PROLOG programming and a discussion of the relationships between PROLOG and logic is given by Clocksin and Mellish[7]. Below, we outline some

features relevant to its use in considering protein topology. A helpful view of computation[8] is:

Algorithm = Logic + Control

The Logic of an algorithm is a declarative statement of what the algorithm is to do; its specification. The Control elements indicate how it is to be executed. Imperative programming languages (PASCAL, FORTRAN, etc.) require the user to mix Logic and Control together. This makes the program difficult to understand and also difficult to write. In logic programming, the user need only write the program specification, i.e. the Logic while the Control mechanism is automatically incorporated in the theorem prover. The logic programming language PROLOG (named after PROgramming in LOGic) was developed[9] from concepts used in programs that automatically constructed proofs to theorems stated in logic using a general proof procedure and axioms presented as data (e.g. Reference 10).

PROLOG stores facts (ground clauses) in its program database. For example, a database of PROLOG ground clauses that represents a part of the genealogy of the British royal family is:

```
father_of(charles,william).
father_of(charles,henry).
father_of(philip,charles).
father_of(philip,andrew).
```

where the first line states the fact that charles is the father of william. In addition, rules can be included in the database to infer new relationships from the ground clauses, e.g.

```
grandfather_of(GPA,GCHILD) :-
    father_of(GPA,X),
    father_of(X,GCHILD).
```

The symbol :- is read as 'if', and the comma indicates a conjunction (and).

This rule therefore reads:

GPA is the grandfather of GCHILD **if**
GPA is the father of (some person) X **and**
X is the father of GCHILD.

The symbols that start with an uppercase letter signify PROLOG variables (e.g. GPA) while symbols that are all lower case are constants (e.g. philip).

The contents of the PROLOG database can be used in a number of ways. It can be queried from the interpreter (?— indicates a query) e.g.

```
| ?— father_of(charles,WHO).          (Example 1)
WHO = william ;
WHO = henry ;
no
```

PROLOG responds to a query by searching the database to identify variables in the query with constants in the database. Formally, one states that a variable is unified with a constant. In Example 1, the variable 'WHO' is unified first with 'william'. Typing a semicolon ; instructs the interpreter to search for another solution to the query. Thus, 'henry' is also a satisfactory answer. Requesting a second resatisfaction results in the answer 'no' from the interpreter indicating that no more answers can be found.

The query could have been posed with variables in either or both arguments.

e.g.

```
| ?— father_of(DAD,CHILD).          (Example 2)
CHILD = william,
DAD = charles ;
CHILD = henry,
DAD = charles ;
CHILD = charles,
DAD = philip ;
CHILD = andrew,
DAD = philip ;
no
```

Alternatively, the PROLOG interpreter can be asked to prove a theorem (rule), which is equivalent to inferring a new relationship from the database facts.

e.g. 'is it true that henry is the grandchild of philip?'

```
| ?— grandfather_of(philip,henry).    (Example 3)
yes
```

or 'which persons exist that have philip as their grandfather?'

```
| ?— grandfather_of(philip,GCHILD).   (Example 4)
GCHILD = william ;
GCHILD = henry ;
no
```

Another feature is the use of recursive rules such as the definition

```
ancester_of(Parent,Child) :- father_of(Parent,Child).
ancester_of(Parent,Child) :- father_of(Parent,X),
                             ancestor_of(X,Child)
```

Although trivial in content, the above examples demonstrate many of the elegant features of PROLOG (logic programming) that have led to the current interest in PROLOG applications.

REPRESENTATIONS OF PROTEIN STRUCTURE

The description of the topology of protein structures uses a variety of simplified representations of the precise path of the polypeptide main chain. One form (Figure 1a) represents each β-strand as an arrow, α-helices as a spiral, and a thin wire for the connecting regions. The prealbumin monomer is represented in this form and the Figure shows the stacking of one β-sheet formed from strands 'f,e,b,c' on the second formed from strands 'h,g,a,d' together with a short α-helical region just after strand 'e'. The next level of abstraction (Figure 1b) is

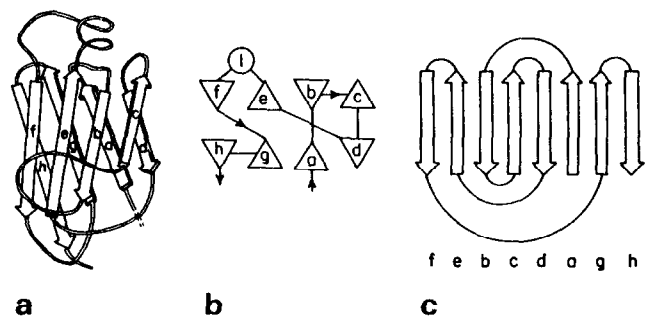

Figure 1. Three abstract representations of protein structure; (a) structure cartoon; (b) circle and triangle diagram; (c) 2D topological structure

the representation in which each β-strand is denoted by a triangle with the position of the apex indicating strand direction; α-helices are denoted by circles, e.g. Reference 11. In protein domains such as prealbumin, in which there is a stacked pair of β-sheets, the structure can be abstracted into a β-barrel. For example, edge strand 'f', which is adjacent to strand 'e' is abstracted as being adjacent to both strand 'h' and strand 'e'. Similarly strand 'c' can be considered adjacent to strands 'b' and 'd'. From such an abstraction, the relative positions of all the strands can be denoted in a planar form (see Figure 1c)[1].

The Brookhaven databank files contain information about the start and end residues of each β-strand, together with the relative positions and orientations of β-strands in a β-sheet (e.g. strand 'f' is next to and antiparallel to strand 'e'). The conversion from the original format in the Brookhaven data file(s) to a structure compatible with PROLOG was performed using a FORTRAN program. The principal aims in our choice of representation formalism were to make the conversion from the Brookhaven data structure to a PROLOG-compatible form, so that the PROLOG representation directly reflected the contents of the Brookhaven database; additional processing or inference would be executed by PROLOG rules. Furthermore, the relationships among structural features would be represented by binary relations (analogous to father_of(charles,william)). The binary relational model of data is both general and easy to understand when used in queries or rules.

The PROLOG clauses derived from the Brookhaven database are illustrated in Figure 2a, where the output from a conversion of the prealbumin entry (2PAB) is shown. This study was restricted to topological motifs formed from β-strands. The clauses used are:

is_part_of(STRUCTURE,SUPER_STRUCTURE).

to describe the structural organization hierarchy, the layers represented explicitly are β-strand, sheet and domain,

a

```
protein_name(p2pab,'PREALBUMIN (HUMAN PLASMA)').

is_part_of(strand(a),sheet('A')).
is_part_of(strand(d),sheet('A')).
is_part_of(strand(g),sheet('A')).
is_part_of(strand(h),sheet('A')).

is_part_of(sheet('A'),domain(a)).

follows(strand(b),strand(a)).
follows(strand(c),strand(b)).
follows(strand(d),strand(c)).
follows(strand(e),strand(d)).

is_antiparallel_to(strand(a),strand(d)).
is_parallel_to(  strand(g),strand(a)).
is_antiparallel_to(strand(h),strand(g)).
is_antiparallel_to(strand(b),strand(c)).
is_antiparallel_to(strand(e),strand(b)).
is_antiparallel_to(strand(f),strand(e)).

is_databank_neighbour_of(strand(a),strand(d)).
is_databank_neighbour_of(strand(g),strand(a)).
is_databank_neighbour_of(strand(h),strand(g)).
is_databank_neighbour_of(strand(b),strand(c)).
is_databank_neighbour_of(strand(e),strand(b)).
is_databank_neighbour_of(strand(f),strand(e)).

is_part_of(strand(b),sheet('B')).
is_part_of(strand(c),sheet('B')).
is_part_of(strand(e),sheet('B')).
is_part_of(strand(f),sheet('B')).

is_part_of(sheet('B'),domain(a)).

follows(helix(l),strand(e)).
follows(strand(f),helix(l)).
follows(strand(g),strand(f)).
follows(strand(h),strand(g)).
```

b

```
is_leftmost(strand(f),sheet('A')).
is_leftmost(strand(h),sheet('B')).

is_parallel_to(  strand(h),strand(f)).

is_oriented(strand(a),up).

is_located(sheet('A'),above).
```

Figure 2. (a) PROLOG clauses derived automatically from prealbumin (2PAB) entry in Brookhaven database. (b) PROLOG clauses added manually to complete the secondary structural description of prealbumin. Other protein structures converted to PROLOG are: carbonic anhydrase C (1CAC), alpha chymotrypsin (tosyl) (2CHA), concanavalin A (2CNA), immunoglobulin Fab V_H and C_{HI} domains (3FAB), Staphylococcal nuclease (2SNS), superoxide dismutase (2SOD) and tomato bushy stunt virus protein (2TBV).

follows(STRUCTURE2,STRUCTURE1).

to describe the path taken by the polypeptide chain,

is_parallel_to(strand(A),strand(B)).

and

is_antiparallel_to(strand(A),strand(B)).

to describe the relative orientation of strands, and

is_databank_neighbour_of(strand(A),strand(B)).

to describe the neighbour relationships between strands in each sheet.

In addition to secondary structural features, the polypeptide sequence data and the positions of the structural elements in the sequence are also converted to PROLOG format.

The PROLOG clauses at this stage partially describe a structure equivalent to the abstract circle and triangle diagrams drawn manually to illustrate features at the level of protein supersecondary structure[11] (see Figure 1b).

In order to complete the PROLOG representation of this Figure, additional clauses were added manually to compensate for the following difficulties discovered while processing the secondary structural assignment data in the Brookhaven database. In a structure such as prealbumin, which consists of a stacked pair of β-sheets, neither the relative position of the β-sheets, nor the absolute orientation of any strand is specified in the Brookhaven databank. This data was therefore added manually to reflect the published protein structure (see Figure 2b for clauses). PROLOG clauses describing the topology of representative proteins were constructed from the Brookhaven data bank (See caption of Figure 2).

The representation of protein supersecondary β-structural motifs was intended to be easy to construct and understand for a nonPROLOG user. It was also intended to be a declarative description of the structure and reflect the complexity hierarchy of β structures suggestive of a possible folding theory[12].

TOPOLOGICAL MOTIFS

The aim of this study is to extract topological motifs such as hairpins and Greek keys from the database. Motifs (Figures 3–7) are represented by PROLOG rules that in the general case are of the form:

motif_name(HANDEDNESS, TYPE, STRANDS)

All rules use the STRANDS argument that specifies which strands are involved in the motif. For the simplest motif, the β-hairpin (Figure 3) the HANDEDNESS and TYPE arguments need not be used. A protein can be said to contain a hairpin with two strands A and B if B succeeds A along the polypeptide, A is adjacent to B in a planar representation of the sheet and A and B are antiparallel. A meander can be considered present in three strands A, B and C if A and B can be shown to form a hairpin and B and C another hairpin (Figure 4). Figures 5, 6 and 7 show the definitions of four- and five-stranded Greek key motifs and a six-stranded jelly roll. These motifs use the Type argument which is a PROLOG list structure. Each element specifies the

```
hairpin (strands ( [ A,B ] )) :-
    succeeds (A,B),
    adjacent (A,B),
    are_antiparallel (A,B).
```

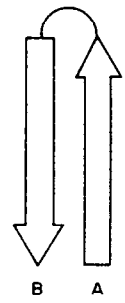

Figure 3. Two-stranded β-hairpin motif and the PROLOG definition

```
meander (strands ( [ A,B,C ] )) :-
    hairpin (strands ( [ C,B ] )),
    hairpin (strands ( [ B,A ] )).
```

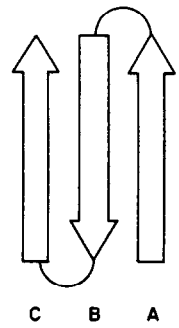

Figure 4. Three-stranded β-meander and the PROLOG definition

```
greek-key ( [ 3,1,1 ], strands ( [ A,B,C,D ] )) :-
        succeeds ( B,A ),
        adjacent ( A,D),
        are_antiparallel (A,D ),
        meander (strands ( [ B,C,D ] )),
        A\= = C.
```

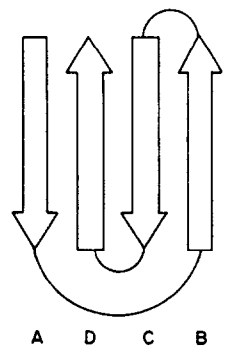

Figure 5. Four-stranded Greek key type [311] and the PROLOG definition

```
greek_key([3,1,1,3], strands([A,B,C,D,E])):-
    greek_key([3,1,1,], strands([A,B,C,D])),
    greek_key([1,1,3], strands([B,C,D,E])).
```

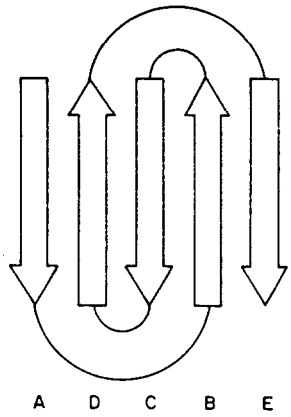

Figure 6. Five-stranded Greek key type [3113] and the PROLOG definition

```
jelly_roll([5,3,1,1,3], strands([A,B,C,D,E,F])):
  succeeds(B,A),
  adjacent(A,F),
  are_antiparallel(A,F),
  greek_key([3,1,1,3], strands([B,C,D,E,F])).
```

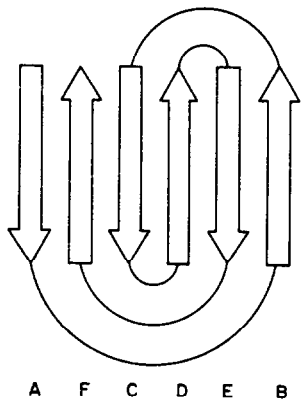

Figure 7. Six-stranded jelly roll type [53113] and the PROLOG definition

```
greek_key(Handed, Type, Strands):-
    greek_key(Type, Strands),
    is_handed(Handed, greek_key, Strands).
```

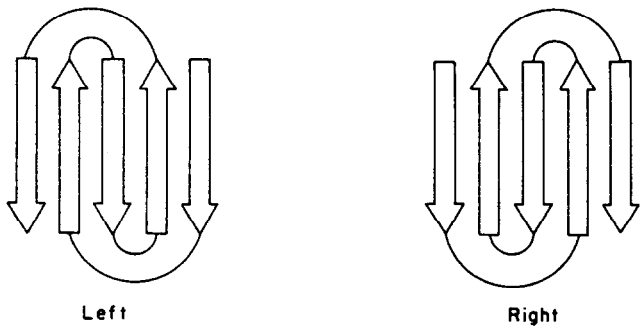

Figure 8. Classification of Greek keys according to their handedness

```
is_handed(right, greek_key, strands([A,B,Cl_])):-
        is_left_of(but(1), A, C),
        oriented(A, down).

is_handed(right, greek_key, strands([A,B,Cl_])):-
        is_right_of(but(1), A, C),
        oriented(A, up).
```

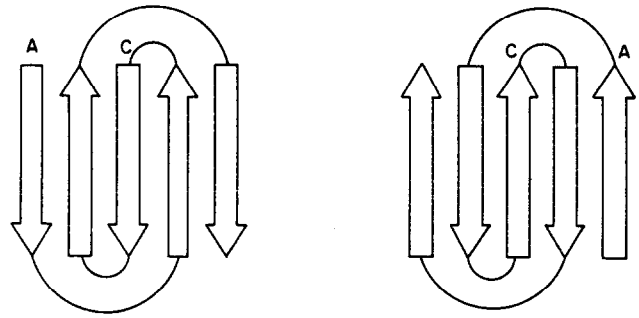

Figure 9. PROLOG rules that distinguish right handed Greek keys

number of strands crossed in moving from the current strand to the next sequential strand in the motif. Thus all instances of the motif may be found by leaving the Type unspecified, or it may be partially or fully specified to initiate the search for a subset of motifs.

Greek keys and jelly rolls can spiral in either a right- or left-handed sense. Accordingly, rules for determining handedness are used (Figure 8). The general definition of a Greek key is used, and then the handedness rules determine the handedness of the structure found. The same handedness rules (Figure 9) are used for all Greek keys, and thus a separate definition for each type of Greek key is avoided. This is both economical and, more importantly, knowledge about how the handedness of Greek keys is determined is kept independent of the motif definition.

TOPOLOGICAL REASONING

Supersecondary structural motifs are often distributed over more than one β-sheet and also occur in β-barrels. In previous studies where supersecondary structure has been identified by eye[1,11,12], the topological transformation from the structural 'cartoon' (Figure 1a) to a 2D representation (Figure 1c) has been achieved manually, sometimes using an intermediate representation (Figure 1b). In order for the PROLOG program to find all potential structures, rules were incorporated that enable it to perform the same topological transformations into a 2D representation.

RESULTS

A query requiring proof of the existence of a supersecondary structural motif (hairpin, meander, Greek key, jelly roll) in a particular protein, unifies the Strands argument of the motif rule with a list of strands that constitute the motif. Additional complexity can be added to the query for some motifs by the requirement (or specification) of Handedness or Type information (Figure 10). In addition, the amino-acid sequence of

```
QUERY   | ?— greek_key(Type,Strands).
ANS     Strands = strands([strand(a),strand(b),strand(c),strand(d)]),
        Type = [3,1,1] ;
ANS     Strands = strands([strand(b),strand(c),strand(d),strand(e)]),
        Type = [1,1,3] ;
ANS     Strands = strands([strand(a),strand(b),strand(c),strand(d),strand(e)]),
        Type = [3,1,1,3[ ;
ANS     no

QUERY   | ?— greek_key([3,1,1,3],Strands).
ANS     Strands = strands([strand(a),strand(b),strand(c),strand(d),strand(e)]) ;
ANS     no

QUERY   | ?— greek_key(Handedness,Type,Strands).
ANS     Strands = strands([strand(a),strand(b),strand(c),strand(d)]),
        Handedness = right,
        Type = [3,1,1] ;
ANS     Strands = strands([strand(b),strand(c),strand(d),strand(e)]),
        Handedness = right,
        Type = [1,1,3] ;
ANS     Strands = strands([strand(a),strand(b),strand(c),strand(d),strand(e)]),
        Handedness = right,
        Type = [3,1,1,3] ;
ANS     no
```

Figure 10. Example queries to the PROLOG interpreter with the clauses describing the topological structure of prealbumin in the PROLOG database

the entire motif, or the strands or the interstrand regions, can be extracted from the PROLOG database.

Enquiries of this kind were successfully performed on proteins selected as representing examples of all the β-structural motifs (see caption, Figure 2). Each enquiry requires PROLOG to search its current database for a small number of successful solutions from a very large search space. In order to achieve maximum efficiency, subgoals were ordered so that the simplest goals were executed first. Variables in the body of the rule are thus unified as soon as possible to reduce the number of resatisfactions (backtracking) by more complex (costly) goals. Table 1 shows the dependence of efficiency on the order of subgoals using an interpreted and compiled program on a Digital Equipment Corporation DECsystem 2060 running Edinburgh University DECsystem-10 PROLOG[13].

DISCUSSION AND CONCLUSION

This study demonstrates that PROLOG can provide a method of encoding abstract representations of protein topology into a computer database. PROLOG provides a mechanism for generating queries that can not only extract information from the database, but, more importantly, can perform topological reasoning. The reasoning is based on a hierarchical description of structural motifs that reflects the organization of protein structure (strands, hairpins, meanders, Greek keys, jelly rolls). It is proposed to extend this approach to proteins containing α-helices in addition to β-sheet proteins.

Table 1. Times required to find all 4 strand Greek keys in prealbumin (seconds)

	Optimum execution time	Worst execution time (reverse subgoal order)
Interpreted	1.8	10.0
Compiled	0.4	2.4

The facilities in PROLOG contrast with those provided by more conventional database systems such as the relational model (based on tables of information)[14]. From a relational database information can be extracted but the inference facilities are limited to operations such as Select, Project and Join. An extension of this work would be to combine the reasoning facilities of PROLOG with a relational database of protein structure. PROLOG would provide the framework for building an intelligent query language, while the relational database would manage the large structural database. A further extension would be to use PROLOG to establish the topological similarity between proteins using similar techniques to those employed in molecular sequence comparison algorithms. At present, measures of structural similarity are based on atomic coordinates.

The direct conversion of a relational database of protein structure to a PROLOG database of ground clauses has been demonstrated recently[15]. The PROLOG reimplementation was simple, and found to be more efficient than the relational database. A possible explanation for this is that most PROLOG systems rely on virtual memory management to manipulate large amounts of information. A conventional relational database system based on disc files would be slower than a virtual memory system but would have a greater data capacity.

Developing from a small number of simple definitions to complex rules defining topological motifs was found to be simple in PROLOG, because the rules were comprehensible. Furthermore, the flexibility of logic programming methods meant that all the subordinate rules could be used in a variety of ways to extend the query language. These features also contributed to the economy of program code. The topological reasoning occupied 143 lines of executable PROLOG containing 49 rules in 21 defini-

tions. The definitions of motifs occupied 49 lines containing 12 rules in six definitions.

The use of PROLOG to describe molecular structures at the topological level is not confined to proteins. The same methods could be applied to databases of smaller molecular structures.

AVAILABILITY

The protein structural data converted to PROLOG, the conversion program and associated PROLOG programs are available from C J Rawlings at ICRF.

ACKNOWLEDGEMENTS

M J E Sternberg wishes to acknowledge support from the Royal Society, W R Taylor and J Nyakairu from the Science and Engineering Research Council, and C J Rawlings and J Fox, from the Imperial Cancer Research Fund.

REFERENCES

1 **Richardson, J** 'β-Sheet topology and the relatedness of proteins' *Nature* Vol 268 (1977) pp 495–500

2 **Richardson, J** 'The anatomy and taxonomy of protein structure' *Adv. Prot. Chem.* Vol 34 (1981) pp 167–339

3 **Cohen, F E, Sternberg, M J E and Taylor, W R** 'Analysis and prediction of the packing of α-helices against a β-sheet in the tertiary structure of globular proteins' *J. Mol. Biol.* Vol 156 (1982) pp 821–862

4 **Taylor, W R and Thornton, J M** 'Prediction of super-secondary structure in proteins' *Nature* Vol 301 (1983) pp 540–542

5 **Sternberg, M J E, et al.** 'Reasoning about protein topology using the logic programming language PROLOG, Abstract 11, *J. Mol. Graph. Soc.* Vol 3 No 3 (September 1985) pp 108–109

6 **Bernstein, F C, et al.** 'The protein data bank: a computer-based archival file for macromolecular structures' *J. Mol. Biol.* Vol 112 (1977) pp 535–542

7 **Clocksin, W F and Mellish, C S** *Programming in PROLOG Springer-Verlag, FRG (1981)*

8 **Kowalski, R** 'Algorithm = Logic + Control' *Commun. ACM* Vol 22 (1979) pp 424–436

9 **Colmerauer, A, et al.** 'Un systeme de communication homme-machine en Francais' Preliminary Report, Group de Reserche en Intelligence Artificielle, Université de Aix-Marseille, Luminy, France (1972)

10 **Newell, A and Simon, H** 'GPS a program that simulates human thought' in **Feigenbaum and Feldman (Eds)** *Computers and thought* McGraw Hill, New York, USA (1963) pp 279–296

11 **Sternberg, M J E and Thornton, J M** 'On the conformation of proteins: the handedness of the connection between parallel β-Strands' *J. Mol. Biol.* Vol 110 (1977) pp 269–283

12 **Ptitsyn, O B and Finkelstein, A V** 'Similarities of protein topologies: evolutionary divergence, functional convergence or principles of folding?' *Ann. Rev. Biophys.* Vol 13 (1980) pp 339–386

13 **Bowen, D L, et al.** 'DECsystem-10 PROLOG user's manual' Department of Artificial Intelligence, University of Edinburgh (1982)

14 **Codd, E F** 'A relational model of data for large shared data banks' *Commun. ACM* Vol 13 (1970) pp 377–387

15 **Burridge, J M, Morffew, A J and Todd, S J P** 'Experiments in the use of PROLOG for protein querying', Abstract 13, *J. Mol. Graph.* Vol 3 No 3 (September 1985) p 109