Expert Antibody Side Chain Placement

Dave Ritchie

Department of Computing Science
University of Aberdeen
King’s College
Aberdeen AB9 2UB

Abstract

Antibody modelling by side chain placement on an homologous backbone conformation is a combinatorially hard problem. This can be made tractable by using a knowledge-based approach in which a heuristic cost function is used to perform a structural search on a database of known structures in order to find re-usable clusters of side chains. The procedure described yields a plausible structure which would be a good starting point for a detailed molecular dynamics simulation or one which a human expert could manually refine.
Declaration

I declare that this thesis has been composed by myself and describes my own work. It has not been accepted in any previous application for a degree. Any verbatim extracts have been distinguished by quotation marks and referenced. All sources of information have been specifically acknowledged.

David W. Ritchie

1 September 1995

Department of Computing Science
University of Aberdeen
King’s College
Aberdeen AB9 2UB
Acknowledgements

I would like to thank my supervisor Dr. G.J.L. Kemp for the many helpful discussions and his continuous encouragement. I would also like to thank Suzanne Embury for her rapid response to the rare problems with P/FDM, whether real, or more usually, self-inflicted. I am also grateful to Prof. J. Fothergill for allowing me to use the computer facilities at Marischal College.
Contents

1 Introduction ................................. 2
  1.1 Project Motivation .......................... 2
  1.2 Project Objectives .......................... 4
  1.3 Report Structure ............................ 5

2 Background ................................. 6
  2.1 Protein Chemistry ............................ 6
  2.2 Antibody Domains ............................ 10
  2.3 Side Chain Placement ........................ 13
  2.4 Related Work ............................... 14
    2.4.1 Tertiary Templates ....................... 15
    2.4.2 Topological Equivalence ................... 16
    2.4.3 Backbone-dependent Rotamers .............. 16
    2.4.4 Dead-end Elimination ..................... 16
    2.4.5 Free Energy Minimisation .................. 18
    2.4.6 Three-dimensional Relationships ............ 18
    2.4.7 The AbM System ........................... 19
    2.4.8 Discussion ............................... 20
  2.5 The P/FDM Database ........................ 21
3 An Expert System 24

3.1 System Overview .............................. 24
3.2 The Cost Function ................................ 26
  3.2.1 Volumetric Packing .......................... 27
  3.2.2 Side Chain and Backbone Spheres .............. 29
  3.2.3 Steric Overlap ................................ 30
  3.2.4 The Chemical Bonus .......................... 32
3.3 The Rotamer Library ............................. 34
3.4 The Model Database ............................. 38
3.5 Backbone Construction ........................... 40
3.6 Residue Neighbours ............................. 41
3.7 Residue Clusters and the Model Mask ............... 43
3.8 Conserved Side Chains ........................... 47
3.9 Rotamer Placement ............................... 48
3.10 Refining The Model .............................. 48

4 Results 50

4.1 The P53 Model ................................ 50
4.2 The 6FAB Model ................................ 53

5 Conclusions 56

5.1 Evaluation ..................................... 56
  5.1.1 The System .................................. 56
  5.1.2 P/FDM ...................................... 57
  5.1.3 Prolog ...................................... 59
5.2 Future Work .................................... 59
5.3 Achievements ........................................... 61

A User Manual ........................................... 66

B Maintenance Manual ................................. 69

C Human Expert Protocol ......................... 72

D Bounding Sphere Algorithm .................. 75

E Spherical Overlap Volume ................. 77
Chapter 1

Introduction

1.1 Project Motivation

Antibodies are very large biologically active protein molecules. As with proteins in general, antibodies are composed of a combination of up to 20 different amino acid residues (or just residues) which are joined together sequentially to form amino acid chains. These chains often fold in 3-dimensional space to form globular domains. Thus antibodies are often referred to as immunoglobulins since they are immunologically active globular proteins.

Every protein is uniquely distinguished by the sequence of amino-acid residues from which it is composed. Although the number of possible combinations of orientations of such amino acid residues within a protein is theoretically enormous, in Nature each protein is found to consistently take on its own unique shape.

Given that proteins often contain sequences of hundreds of residues, there is an enormous number of potential protein sequences and hence protein structures. Indeed, we each have a library of millions of subtly different antibodies\(^1\) (or the potential to create them) each of which may be specific to a particular antigen, on account of subtly different chemical and geometrical properties.

Although the mechanism of the immune system is outwith the scope of this project, any increase in our ability to predict antibody protein structures would be a significant advance in the understanding of many biological phenomena, including the behaviour of the immune system.

---

\(^1\)Branden and Tooze [3], for example, give a good introduction to antibody biology.
It is a relatively routine task to determine experimentally the sequence of amino acid residues in a protein, however it is much more difficult to determine a protein’s 3-dimensional structure. Current methods using X-ray crystallography or Nuclear Magnetic Resonance (NMR) can take several months to determine the complete structure of a single protein molecule.

Certain proteins simply cannot be crystallised, thus ruling out crystallography; they may be too large for current NMR methods (as are antibodies), or the protein solution may coagulate at the concentrations required for NMR. In these cases, predictive methods are the only possible way to obtain a structural model. However, precise prediction of protein shape has been shown to be NP-hard [22] and thus will remain a numerically intractable problem. Approximate methods for structure prediction are an active area of current research.

One of the most commonly used approximations in structure prediction is homology modelling. This makes use of the property that biologically similar proteins often have similar sequences of residues and similar 3-dimensional structures. Antibody molecules are known to have highly similar sequences [16] (even across species such as human and mouse) and studies of known structures [7] have shown that substantial regions of their 3-dimensional structure are often conserved. Thus antibodies are particularly amenable to homology-based modelling. Given the sequence of some new protein, an homology-based model would assume that if this sequence is similar to the sequence of a protein whose structure is known, then the backbone conformation\(^2\) of the unknown structure would be likely to have a similar overall fold to the known structure and therefore could be copied from that structure.

This leads to the sub-problem of protein side chain placement. The remaining atoms of each of the amino acid residues in the model sequence must be placed onto the backbone in such a way as to satisfy volumetric requirements (such as tight packing) yet still honour chemical constraints on all inter-atomic distances and bond angles. This yields a combinatorially hard problem and a number of further approximations need to be made. Some of these approaches are reviewed in Section 2.4.

The purpose of the present project is to attempt to circumvent such combinatorial difficulties by applying a structural search to a database of known antibody structures in order to re-use suitable clusters of side chains.

\(^2\)An amino acid residue is often considered in terms of backbone and side chain components. This is discussed in further detail in Section 2.1 below.
1.2 Project Objectives

The preceding section indicated that side chain placement is very much a combinatorial problem. One way to attack a combinatorial problem is to assume that it can be broken up into smaller independent sub-problems. Ideally, solutions to these smaller sub-problems might already exist and be in a form that could be re-used.

An object oriented database, P/FDM [13], has been developed at Aberdeen which contains the experimentally determined structures of some 20 antibody proteins. This database could supply the backbone conformation for a model antibody and also probably very good candidate side chain conformations in those cases where the type of residue is preserved between the model and the backbone donor structure.

However, in regions where sequence similarity has been lost (even in just one or two residues), it might be reasonable to expect that suitable clusters of residues might exist in other antibody structures which could be copied en masse to the model. Such residue clusters would correspond to ‘pre-solved’ sub-problems of the combinatorial search space.

Thus the primary objective of this project is to develop a knowledge-based antibody side chain placement system which makes maximal use of known structures in the P/FDM database. Since P/FDM is implemented in Prolog, this essentially dictated that the final system would also be written in Prolog, although call-outs to C and Fortran could be used for numerical calculations.

It was expected that the use of database clusters would not be sufficient to complete the entire model. Thus it would be necessary to implement additional procedures based on a human expert’s approach (a listing of which is given in Appendix C), along with ideas taken from some of the published procedures for homology modelling.

From a personal perspective, secondary objectives were to explore the suitability of Prolog for scientific programming and to gain experience in functional data modelling.
1.3 Report Structure

The rest of this report is structured in the following way:

Chapter 2 describes the problem domain in more detail, giving a basic description of some of the aspects of protein chemistry and antibody domains. It also reviews some of the published approaches to side chain placement and discusses how these have influenced the current project. Finally, a brief overview of some of the features of P/FDM is also presented.

Chapter 3 discusses some of the design issues of the final system and describes some of the main concepts used, such as the cost function, packing clusters, and rotamer search.

In Chapter 4 some preliminary results are presented for the P53 Fv antibody fragment, which was used as a test case during development of the system. A model of the 6FAB Fv fragment is compared to the known crystallographic structure for this antibody.

Chapter 5 discusses the accuracy and reliability of the system and suggests possibilities for future enhancements. It also reviews the project in terms of major achievements and lessons learnt.

Appendix A gives details on where to find the system and how to run it.

Appendix B describes the software environment and the contents of the various source files and command scripts.

The text of a recent modelling exercise by a human expert is listed in Appendix C.

A description of the bounding sphere algorithm which is used widely in this project is described in Appendix D and the derivation of the overlap volume between two spheres is given in Appendix E.

The overall intention of this report is to give an outline of the project. Inevitably, much of the detail has been simplified or omitted. Fragments of code are used as illustrations but full program listings (with plenty of comments) and example output files are provided separately.
Chapter 2

Background

2.1 Protein Chemistry

Proteins are amongst the largest organic molecules found in nature and are an essential part of all biological processes. Although a typical protein molecule may consist of many hundreds of atoms, nearly all of these atoms derive from much simpler building blocks, known as amino acids. An amino acid is a relatively simple organic molecule which is characterised by an amino group (-NH$_2$), a carboxylic acid group (-CO$_2$H) and a hydrocarbon chain of zero or more carbon atoms. These groups are arranged around a central carbon atom, known as the $\alpha$-Carbon or $C_\alpha$, about which the other groups are arranged approximately tetrahedrally. This is shown in Figure 2.1.

![Amino acid functional groups](image)

Figure 2.1: Amino acid functional groups. Lines represent bonds in the plane. Triangles represent bonds out of the plane.

In nature, there are 20 biologically useful amino acids whose chemical structures are formed by varying the side chain, R, depending on the type of amino acid. For example, R is just H in the case of Glycine, and $C_\beta$-$C_\gamma$($C_\delta$)$_2$ in the
case of Leucine\textsuperscript{1}. Some amino acids have other atoms such as sulphur and oxygen, in addition to the hydrocarbon chain and these can affect the chemical and physical properties of the amino acid. For example if the side chain contains further amino or hydroxyl groups then it tends to be polar and hydrophilic. If it contains only hydrocarbon groups, then it tends to be nonpolar and hydrophobic.

![Diagram of amino acid structures](image)

**Figure 2.2:** Newman projections of phenylalanyl conformations, as viewed along the $C_\alpha$-$C_\beta$ bond axis. Actual conformations may be skewed from these idealised configurations.

Although any given type of side chain is chemically identical to all other side chains of that type, its known from NMR studies that the atoms of a side chain can often adopt a number of different *conformations*. Each conformation is a sterically distinct set of relative orientations of atoms which are formed by rotations about the bond axes of the side chain. This is illustrated in Figure 2.2, which shows the Newman projection of a Phenylalanyl\textsuperscript{2} side chain ($R$ is $C_\beta$-$C_6H_6$) when viewed along the $C_\alpha$-$C_\beta$ bond axis. The different staggered arrangements of functional groups are obtained by successive rotations of about $120^\circ$ about the $C_\alpha$-$C_\beta$ bond axis. These conformations can be described concisely by specifying the torsion angle, $\chi_1$, which is the angle subtended by the N atom and the $C_6H_6$ functional group. Side chains that have a $C_\beta$-$C_\gamma$ bond would have a $\chi_2$ torsion angle, and so on.

There is an energetic cost associated with $\chi$-angle torsions although usually thermal motion at room temperature is sufficient to allow the conformations to flip rapidly from one to the next. Often each conformation has a slightly different energy and a large sample of molecules will contain a distribution of

\textsuperscript{1}By convention, side chain atoms are often subscripted by Greek letters according to their bond separation from $C_\alpha$. The presence of Hydrogen atoms to complete the valency is implicit.

\textsuperscript{2}Phenylalanyl is the chemical name for the side chain that derives from the amino-acid Phenylalanine although this naming distinction is often ignored. Three-letter and one-letter abbreviations are also commonly used: Phe = F = Phenylalanine/yl.
conformations in proportion to these energies (the Boltzmann distribution). Clearly, side chains that have more rotational degrees of freedom can have more conformational isomers. In protein modelling, such conformational isomers are often referred to more simply as rotamers.

Figure 2.3: Amino acid condensation. Torsions are possible about the $C\alpha$-CO and $C\alpha$-N bonds, labelled $\psi$ and $\phi$ respectively. The amide bond N-CO has double-bond character and does not permit torsion. The first side chain torsion $\chi_1$ is also shown.

The amino and carboxylic acid functional groups are chemically relatively reactive and one of the reactions that amino acids can undergo is condensation, in which two amino acid molecules combine and a water molecule is eliminated to form a new peptide bond (N-CO), as shown in Figure 2.3. The result of this dimerisation is a new amino-acid that still has an amino and a carboxy functional group. Thus further condensations can take place to yield a polypeptide chain, better known simply as a protein.

Biological proteins are formed in a similar way, but the sequence of amino acids is strictly controlled by the genetic coding of the DNA sequence. Such proteins are formed in a largely aqueous environment, yet large sections of the protein are hydrocarbon-rich which are strongly hydrophobic. This means that as the protein forms, there is a strong tendency for it to curl up into a globular shape in which most of the hydrophobic residues are buried and most of the hydrophilic residues appear at the surface.

The side chains in a protein often adopt similar conformations to those found in free amino acids, however collisions with other parts of the protein may restrict the variety of conformations that are possible. This turns out to
be a useful constraint which can be applied during side chain modelling.

In nature, proteins are commonly found to adopt similar shaped regions, such as helices (α-helix) and linear or planar strands (β-strands, β-sheets). The formation of these secondary structures is primarily a consequence of Hydrogen bonding\(^3\) along and between sections of the backbone, although the effects of steric constraints and other chemical interactions of the backbone and side chain groups (and solvent) can also be important. However, prediction of these subtle, but vitally important secondary structures from knowledge of the primary protein structure (the residue sequence) is virtually impossible.

Current theories would suggest that the secondary structure forms very early in the birth of a new protein and that these larger units begin to interact with each other (and the solvent), rapidly equilibrating to the final tertiary globular structure. The whole process of protein formation is estimated to take about two seconds [24]. Figure 2.4 illustrates the complex 3–dimensional structure of a protein molecule.

---

Figure 2.4: Space-filling representation of the 6FAB Fv fragment. All backbone atoms are white. The side chain Carbon, Oxygen and Nitrogen atoms are yellow, red and blue respectively.

\(^3\)Hydrogen bonds are what make water wet and a good solvent.
2.2 Antibody Domains

Antibodies are usually made up of 4 protein chains, folded into 12 globular domains, each domain consisting of around 100 residues. The 12 domains are arranged approximately in a ‘Y’ shape. A schematic diagram is shown in Figure 2.5. There are two ‘heavy’ chains, consisting of about 400 residues each, and two ‘light’ chains, each of about 200 residues. The heavy chains make up the inner body of the ‘Y’ and the light chains are attached along its outer arms. There are sufficient attractive inter-molecular forces to bind the chains together so that they usually behave as a single molecule [9].

![Diagram of antibody domains](image)

**Figure 2.5:** Schematic arrangement of antibody domains.

As can be seen from Figure 2.5, the domains are labelled according to the type of chain to which they belong (‘L’ or ‘H’) and whether the domain varies or is conserved from one antibody to the next (‘V’ or ‘C’ respectively). The outer arms of the ‘Y’ shape (VL+CL+VH+CH1) make up what is known as the antibody binding fragment (Fab). Variations in the shape and residue composition in the Fab fragment are responsible for the different specificities of different antibodies. In fact, only the variable fragment (Fv = VL+VH) changes from one antibody to the next and it is just these two domains which need to be modelled since all of the conserved domains are essentially replicated across different antibodies.

Even the variable domains can show high degrees of similarity. When several sequences are examined together in a multiple alignment, it is frequently observed that different antibodies often have similar or identical sub-sequences of residues. This is particularly evident when gaps are introduced to improve
Figure 2.6: First 42 positions in the Kabat multiple alignment of VL domains. Dashes indicate artificial gaps. Each row is from a different antibody VL domain. The first row of numbers is purely for reference. Asterisks point out conserved residues in the alignment.

the sequence match. Figure 2.6 shows portions of a Kabat [16] multiple alignment for a section of the VL domain chain fragments. In such an alignment, the gaps are introduced so as to give the most statistically significant overall sequence match, although often this can be done ‘by eye’ when there is such a high similarity.

In general, Biologists would interpret the differences between such alignments in terms of insertions, deletions or mutations of residues from some (probably unknown) ancestral sequence. Thus the diversity observed today is evidence of evolutionary change from earlier simpler forms4.

The Kabat study found that certain sub-sequences (or regions) of the aligned residues correspond almost directly to specific portions of known structures, even though the alignments were originally produced statistically, without structural information. Generally a large proportion of a sequence is assigned to a framework region which mostly consists of β-strands, and smaller sub-sequences are assigned to hypervariable loops which connect the β-strands and contribute to holding the strands together as a β-sheet. These hypervariable loops are also known as Complementarity Determining Regions (CDRs)

4Antibodies, however, are a special case. In addition to evolution, the body continually produces new variants of antibodies as one of the defences against infection.
since it is these regions which are most important in antigen binding. Figure 2.7 illustrates the positions of the CDRs relative to the framework of \(\beta\)-strands in the 6FAB Fv fragment. This is the same molecule as shown in Figure 2.4, except that the backbone trace is plotted as a ‘ribbon’ to enhance the detail of the \(\beta\)-sheet and loop structures.

Figure 2.7: Ribbon plot representation of the 6FAB Fv fragment, highlighting the six hypervariable loops of the CDR.

Chothia and Lesk [7] have performed detailed studies of known antibody structures. Their interpretation gives slightly different assignments to the framework and CDR regions than would the Kabat assignment, but it is considered to be more accurate when modelling the detailed shapes of the hypervariable loops. Indeed, for most of the loops (H3 excluded) Chothia and Lesk have correlated the sequences and geometric shapes of the loops into what they term *canonical conformations*. Thus the conformation of a loop can often be predicted, given the sequence identity of certain key residues. For example the absence of Proline at VL position 95 would almost certainly specify a particular conformation for the L2 hypervariable loop [8].

It should be pointed out that the current project makes use of the Kabat sequencing scheme but not the Chothia and Lesk structural assignments. Thus the largest errors in the system might be expected to be in the predicted structure of the hypervariable loops.
2.3 Side Chain Placement

If a molecular graphics package, such as QUANTA\textsuperscript{5} is used to examine the 3-dimensional structure of a protein it becomes immediately obvious just how remarkable protein folding really is. The backbone appears to twist and turn through space and the side chains branch out from the backbone, filling virtually all of the interior space. Yet no side chain or backbone atom ever gets ‘too close’ to any of its neighbours (other than those to which it is formally bonded). If the atoms are colour-coded by their atom type, then its observed that those atoms that are known to form hydrogen bonds are usually at just the right separation for this to take place.

Although most side chains can each adopt several different conformations in space, the observed structure gives the appearance of having ‘frozen’ into precisely the correct combination of conformations. Given some homologous backbone shape as a starting point, the task of side chain placement is to find this correct set of orientations.

Since an antibody domain can contain around 100 residues and thousands of atoms, it would be impossible to simultaneously solve for all the atomic positions. Even if each residue is represented by a set of idealised rotamers then this still leaves a combinatorial problem over the number of rotamers. If each residue $i$ has $R_i$ rotamers and there are $n$ residues, then the number of possible combinations of rotamers $N$ is given by:

$$N = \prod_{i=1}^{n} R_i$$  \hspace{1cm} (2.1)

If each residue had 5 rotamers, then $N$ would be of the order of $5^{100} \approx 10^{70}$.

Fortunately, side chain conformations are not completely mutually dependent. For example, Hurley [15] describes how protein structures are quite tolerant to single and pairwise residue mutations, although the enzymatic effectiveness of the protein can often be reduced or lost in such cases.

This project attempts to exploit this behaviour by \textit{assuming} that there are discrete spatially related clusters of side chains within a protein which are mutually independent. Thus the combinatorial product in Equation 2.1 can be simplified (approximately) to:

\textsuperscript{5}\textit{QUANTA} is a commercial product of Molecular Simulations Inc.
\[ N = \sum_j^n \prod_i R_i(j) \] (2.2)

Where each residue \( i \) now belongs to some cluster \( j \). If each residue still had 5 rotamers but these were found in 10 independent clusters, then \( N \) would be of the order of \( 10 \times 5^{10} \approx 10^7 \).

Rather than simply attempting to solve the smaller combinatorial problem for each cluster, it might be the case that such clusters already exist in a database of known structures. In a sense, the database of antibody structures could be considered to contain a large but feasible number of clusters for which Nature has already solved the combinatorial problem. If a suitable search of the database can find such clusters, then a reasonable knowledge based solution might be obtained.

It should be stressed that a spatial search for side chain clusters differs substantially from a sequence search. A sequence search might find a re-usable secondary structure such as an \( \alpha \)-helix or a \( \beta \)-strand, but this would rely on there being a continuous sequence match for the extent of that structural feature. A spatial search relies on residue identity in a spatial region rather than along a sequence. This might be particularly appropriate for antibody modelling (in which the domains of interest have no \( \alpha \)-helices, but do have large extents of \( \beta \)-sheet). A spatial search might identify the good \( \beta \)-strand interactions which help to form the \( \beta \)-sheet, and it might still do well in the hypervariable regions that have lower sequence identity.

## 2.4 Related Work

Most side chain placement algorithms employ the concept of rotamers in some form. The purpose of this section is to review some of the approaches that appear in the literature.

It is only in the last few years that sufficient protein structures have been determined to allow Molecular Biologists to begin to search for structural patterns and rules. In Artificial Intelligence parlance, Molecular Biologists are still at the knowledge acquisition stage of this part of their science. This might explain why most of the papers are by Biologists who are taking on board ideas from Computer Science, rather than by Computer Scientists offering practical rule-based or knowledge-based solutions.
2.4.1 Tertiary Templates

Ponder and Richards [23] demonstrated that remarkably good results could be obtained simply by applying two packing criteria: avoidance of steric overlap and complete filling of available space. The recent availability of high resolution crystallographically determined structures meant that they could accurately represent 17 of the 20 amino-acids\(^6\) using just 67 rotameric conformations derived from these structures.

One objective of Ponder and Richards’ work was to assist Crystallographers by taking a partial crystallographic structure (the backbone trace) and enumerating all possible rotamers that were compatible with each residue site on the backbone, eliminating all rotamers whose side chains sterically clashed with the backbone. For the ‘interior’ residues, potential sidechain-sidechain clashes were detected by considering selected residues to be the centre of a packing site of some given volume (chosen by eye, so as to enclose from between 5 to 8 residues) and by noting in a matrix those pairs of rotamers in the site that would sterically clash. They called the resulting list of rotamers the tertiary template\(^7\) for the protein structure. This approach could also be used as a filter to determine whether a particular sequence could be said to be compatible with a given structural family (i.e. the tertiary template is ‘masked’ with the given sequence: if all rotamers are eliminated at one or more sites, then its unlikely that the sequence would have a structure suggested by the template).

This principle is also applicable to side chain placement. A given sequence can be used to select the appropriate rotamer(s) for each site from an homologous template list. Successively tightening a volumetric packing constraint could be used to select the most likely sequence of rotameric assignments. This is not a particularly efficient approach if the task is purely one of side chain placement, since the initial steps are very wasteful. However it did serve to demonstrate the utility of simple steric and packing criteria and it introduced the notion of packing sites as a means of dealing with sidechain-sidechain interactions.

\(^6\)Met, Arg and Lys have many conformations but these were not all known with certainty at that time.

\(^7\)It should be stressed that in the rest of this report, a template refers to the backbone conformation of a known structure when used as the starting point for side chain modelling.
2.4.2 Topological Equivalence

At about the same time, Sutcliffe et al. [28] analysed the distribution of side chain conformations in relation to their secondary structure environment (α-helix, β-sheets or loop). They concluded that structurally conserved regions are often topologically equivalent and that the conformations of side chains in equivalent positions are often conserved. Interestingly, the structures that they analysed included several antibody Fab domains, although they report results only for myoglobin.

They utilised these findings in a rule-based side chain placement procedure in which side chain torsion angles are transferred from the template residue to the corresponding torsion angles in the model side chain, whenever the two side chains are in the same structural environment. Otherwise, the model conformation is assigned the most probably rotameric conformation for its particular secondary structure. They report that this method works well particularly for the buried hydrophobic residues, but less well for the larger more flexible side chains (such as Arg and Lys).

2.4.3 Backbone-dependent Rotamers

As more high resolution structures have become available, more detailed analyses of side chain conformations have become possible. Dunbrack and Karplus [12] compiled a backbone-dependent rotamer library from 132 PDB\textsuperscript{8} protein structures. This lists preferred rotamer conformations for each pair of backbone φ and ψ angles in 20° intervals.

When combined with the CHARMM\textsuperscript{9} energy minimisation procedure, the use of these rotamers can correctly place side chains with an accuracy of from 61% for lysozyme to 81% for crambin (side chains whose \( \chi_1 \) and \( \chi_2 \) angles are predicted to within 40° of the crystallographically determined values were considered to be correct).

2.4.4 Dead-end Elimination

One of the more mathematical techniques is the Dead End Elimination (DEE) algorithm of Desmet et al. [10]. This also makes use of a rotamer library but in a somewhat novel way. The basic principle is to eliminate candidate rotamers

\textsuperscript{8}The Protein Data Bank [2] at Brookhaven is one of the main protein structure libraries.  
\textsuperscript{9}CHARMM [4] is the molecular dynamics module of the QUANTA package.

18
on energetic considerations. All energy-based approaches attempt to solve for the minimum energy conformation of the system. In terms of rotamer search, Desmet calls this the \textit{global minimum energy conformation} (GMEC).

The basic DEE algorithm applies its elimination procedure to pairs of interacting residues via the inequality:

\[
E(I_r) + \sum_j \min_s E(I_r, J_s) > E(I_i) + \sum_j \max_s E(I_i, J_s) \quad ; i \neq j
\]

where \(I_r\) represents a residue \(I\) in rotamer state \(r\); \(E(I_r)\) is the ‘inherent energy’ of the residue in this state; \(E(I_r, J_s)\) is the interaction energy between two residues in particular rotameric states; the min and max terms are applied over the possible rotameric states \(s\); the summations are applied over all possible neighbouring residues \(J\). Stated more simply, this says that for any residue in a given rotameric state, if the best energy obtained with the residue in this state and in the most favourable local environment of neighbouring rotamers is worse than the worst energy obtained by having the residue in any other state, then that state cannot contribute to the GMEC.

This principle can be applied repeatedly to eliminate successive rotamers from consideration. It is claimed to converge rapidly to the GMEC. However, this is not necessarily the true thermodynamic free energy minimum: its simply the best conformation available with the given library of rotamers. The algorithm does not allow for small adjustments to torsion angles nor for solvation effects which can be important.

Desmet \textit{et al}. (1994) [11], report a trial of the method on Insulin which has 102 residues of which 76 have rotatable side chains. They produced a single GMEC structure in just 14 minutes of CPU time on a VAX 8650 which correctly placed 72\% of the rotamers. However, it should be noted that this was using table lookups to perform the energy calculations. The execution time was 5.6 CPU hours when the energy minimisation was performed explicitly.

Tanimura \textit{et al}. [29] used the DEE algorithm in a side chain modelling exercise in which they factored out the relative contributions of sidechain-sidechain and sidechain-mainchain effects. They concluded:

\textit{“The sidechain-mainchain and sidechain-sidechain interactions work concurrently to favour the native conformations. This complementary nature would discriminate in favour of the unique and stable sidechain conformations found in native proteins.”}
2.4.5 Free Energy Minimisation

One of the more recent and probably more accurate protein modelling experiments is reported by Wilson et al. [30]. Their approach used the accurate rotamer library of Ponder and Richards [23] and applied an explicit inter-atomic energy minimisation calculation which included solvation terms.

Initially, arbitrary rotameric conformations are assigned at each residue site along the fixed backbone template. Residue sites are then selected randomly and a local free energy minimisation calculation is applied around the packing site (typically of 5 residues) centred on that residue, and a combinatorial search is used to select the energetically most favourable rotamer for that site. This is repeated until all residue sites have been visited. The overall cycle is then repeated until no new rotameric assignments are made or until some fixed limit on the number of outer cycles has been reached (typically 3).

This study also attempted to determine the relative importance of the various approximations that were being used. For example, they estimated that their force field expression was good enough to correctly locate 89% of the rotameric residues in alpha-lytic protease and that 7% of the placement errors were due to the approximations which are inherent in using a rotamer library. As usual, most of the errors were from hydrophilic residues at the protein surface which would indicate inaccuracies in the solvation terms.

Overall, the method accurately predicts about 80% of residue conformations when there is good homology with the backbone template but this falls to about 60% where the homology is poor. This demonstrates the impact of adopting an incorrect backbone template purely from homology and they propose incorporating relaxation of the backbone template once the side chain conformations have been sufficiently refined.

2.4.6 Three-dimensional Relationships

A recent homology modelling exercise by Laughton [19] has some interesting similarities to the current project, despite the fact that it does not use an explicit rotamer library.

The starting point in this method uses a database of representative known protein structures. The local environment of each residue in the database is estimated by deriving an approximate residue sphere from a matrix of $C_{\alpha}$ coordinates. These spheres are then used to locate spatially neighbouring residues for each residue in the database. Each residue was found to have an
average of around 2.5 neighbours.

For a given model residue, the database is searched for corresponding residues of that type. The database neighbours of that residue are located and scored according to the similarity between the database $C_\alpha-C_\alpha$ distances and the corresponding distances in the model, along with a bias term from the Dayhoff sequence similarity matrix. The top scoring neighbours are then collected for use as candidate conformations. When this is repeated for all model residues, a list of candidate side chain conformations for each model residue site is obtained. The combination of conformations that give the fewest steric clashes is then selected by a Monte Carlo simulated annealing algorithm, using a very simple potential energy term.

This was found to correctly place around 60% of all residues and 80% of buried residues. Here, the definition of correctness was that all $\chi$-angles be within ±30° of those of the crystallographic structure. Interestingly, subsequent energy minimisation lowered the energy of the structure but did not substantially improve the accuracy of the $\chi$-angles.

2.4.7 The AbM System

A good example of a commercial antibody modelling system is AbM\footnote{AbM is a commercial product of Oxford Molecular Ltd.} [20]. This probably provides the most accurate modelling to date of antibody variable domain structures, particularly in the hypervariable loop regions. However, it can involve some major computation. According to the User Manual, “total run times can range from hours to days”.

AbM uses a database of known antibody structures in combination with an ab initio conformational search using the CONGEN program [6]. It models the VL and VH framework residues by choosing the best individual domains from the database and by applying a least-squares fitting scheme to bring the two domains into the same coordinate frame.

The hypervariable loops (except for H3) are modelled by initially searching for a matching Chothia and Lesk canonical structure. If this fails to find a match, then depending on the number of residues in the loop, either a database search is used, a conformational search is applied, or a combination of the two is used.

The database search finds all segments of backbone sequences in the entire PDB database that have the required sequence for the loop and that have $C_\alpha$
geometry that is compatible with the loop site. This can yield a great many candidate loop fragments.

For each hypervariable site, every candidate loop is spliced into the structure and a conformational energy minimisation is applied to refine the structure of each candidate loop. The five lowest-energy candidates for each site are then reported as possible solutions and the one with the lowest exposed hydrophobic surface area is selected as the final solution.

Optionally, this modelling procedure can be applied in the presence of an antigen at the binding site.

2.4.8 Discussion

A trend that can be observed in the previous sections is an increasing awareness of the importance of each side chain’s interactions with its local environment. This is evident both in terms of steric and volumetric effects and, as more accuracy is required, also in terms of the detailed energetics (or chemistry) of those interactions.

This project tries to capture the importance and approximate behaviour of these interactions, but in a way which avoids the computational complexity of the ab initio and Monte Carlo types of approach.

Another trend is the increasing reliance on databases of known structures. However, there may be a tendency to replace combinatorial rotamer searches with database searches of comparable complexity. Bruccoleri [5] gives an interesting discussion of some of the problems of protein structure prediction in relation to Levitt’s paradox. This observes that proteins are known to fold into their final shape in only a few seconds; yet the rate constant for changes to torsion angles is in the region of $10^{-9}$ seconds. Thus in the time it takes to form, a protein can sample only around $10^{12}$ conformations which is far less than the theoretical maximum. He uses this as a strong argument in favour of directed search. By this, he means that the best conformations should be generated early on in a search and that information acquired during the search is used to guide subsequent choices.

I would put this more bluntly: Nature doesn’t sample the whole conformational search space, so neither should we.
2.5 The P/FDM Database

P/FDM [13] is an object-oriented database system which has been developed at Aberdeen. It is based on Shipman’s notion of a Functional Data Model [25] and is implemented in terms of a small set of access primitives, which support the associated DAPLEX query language or which can be used directly by application programs.

In P/FDM, objects are represented as members of entity classes and functions are defined on objects to access the attributes or properties of an object. Conceptually, an object can often be considered as a tuple of values rather like tuples in a relational database since, for example, all the attributes of an object can be deleted in a single operation.

Each object instance has both an external key (combination of attributes, as in a relational database) and an internal key (the object identifier). This has the advantage that an object instance can be located directly by either its internal object-id or by its external key. Object instances can be enumerated by their object identifiers. Object uniqueness constraints can be enforced easily by use of the external key.

However, the relational analogy should not be taken too far. For example the actual storage mechanism is based on triples of [object-function-value] rather than on tuples, although this is concealed by the interface. What distinguishes a functional model from a relational model is that the relationships between entity instances (objects) are also represented functionally. This means that queries to a functional database are essentially navigational or tuple-at-a-time, whereas queries to a relational database are inherently set-based.

Of course relational databases do offer a ‘cursor’ interface to give the appearance of tuple-at-a-time, but this is almost invariably implemented by memory-buffering the result of a set-based query. Often complicated joins are required to specify a query which could easily be expressed navigationally. Such joins can be expensive to interpret (not to mention hard for humans to formulate) and potentially wasteful if the query is satisfied or abandoned early, after the first few tuples have been returned.

P/FDM also supports method functions, which allow derived quantities to be calculated or, for example, sequences of relationships to be traversed. Often method functions combine database access with computation to support high-level queries. For example the Daplex query shown in Figure 2.8 prints information regarding the interaction between the two light chain variable domain Cysteine residues (which always appear at Kabat positions 23 and
program q7 is
  for each s in structure
    for each vl in domain_structure_inv(s) such that
      domain_type(vl)="variable" and chain_type(vl)="light"
      print(protein_code(s),
      packing_factor_events(kabat_residue(vl, "23"),
      kabat_residue(vl, "88"));

Figure 2.8: Example Daplex query.

88), as measured by the pairwise cost function (packing_factor_events), which
is used in the current project.

This demonstrates the equal status of a data extraction function such as
protein_code, a multi-valued relationship function such as domain_structure_inv
and a method function packing_factor_events. It also demonstrates enumeration of the structure entity class and the use of a multi-argument function
kabat_residue which maps a Kabat position code number to a residue object
as required for packing_factor_events. The cost function is discussed in more
detail in Section 3.2, in which it should become clear that this method function
invokes a large amount of both database access and computation.

The starting point for the current project is antibody sequence and structure
information, stored in two P/FDM database modules. These have been
built as a result of previous work at Aberdeen [17]. The sequence database is
used to find the best sequence to use for the model backbone. The antibody
database is the source of main chain atoms with which the model backbone is
built and of the side chain atoms with which it is populated.

The database schema diagrams for the sequence and antibody databases are
shown in Figure 2.9. In such schema diagrams, boxes represent entity classes
and relationships between entity classes are represented by arrows. The arrows
are single-headed if there is a one-to-one relationship between objects of the
related classes. A double-headed arrow represents a one-to-many relationship.
Relationships may be optional, in which case entity classes are connected by a
dotted line. Relationships between objects of different database modules may
also be defined (with certain restrictions) and this is illustrated in Figure 2.9.
Figure 2.9: Schema diagrams of the *sequence* and *antibody* database modules. Boxes represent entity classes. Labelled arrows represent relationships. Dotted lines represent optional relationships. Some functions require an additional argument, such as an *atom name* (atom + string), to uniquely specify a relationship.
Chapter 3

An Expert System

3.1 System Overview

Although the main objective of the project is to place side chains by database search, it is likely that not all residues could be placed in this way and so additional placement procedures will be needed. Indeed, the assumption of mutually independent residue clusters is inevitably incorrect and so will probably give rise to some bad side chain assignments which the system must be able to deal with.

Section 2.4 discusses a range of recent approaches to side chain modelling. Many of those approaches are computationally complex, although the database-based approaches lend weight to the validity to the ideas behind the current work. Additionally, the protocol from a recent modelling exercise was made available. This is listed in Appendix C. In brief, the procedure that the expert used can be summarised as follows:

1. The best backbone chains for modelling have the highest sequence identity and contain no insertions or deletions.

2. The most reliably placed residues are those which can be copied directly from the backbone donor structure.

3. Copying a side chain from a corresponding position of some other structure is the next best strategy.

4. If any unplaced residues remain, assign a default rotamer conformation. This is the least reliable operation, although necessary to complete the model.
5. Examine the resulting model and manually correct any badly placed side chains by bond axis rotations. Preferentially adjust conformations that were placed with least confidence.

Some Prolog code that implemented the first four steps was also available, although this was not used directly (other than as inspiration) since it was too specific to that particular modelling exercise.

The first three steps are essentially the same as those recommended by Summers et al. [27] as a result of a detailed analysis of residue conformations in homologous proteins. All five steps can be seen to encapsulate most of the concepts that were found to be significant in the papers of Section 2.4. The system essentially follows this strategy for any residues that the cluster search fails to place. The final step of bond axis rotation is approximated by a rotamer search of the rotamer library since this is much easier to program, even though its likely to be less accurate.

The main stages of this strategy translate almost directly into a top level Prolog goal, as shown in Figure 3.1. The following sections describe some of the issues involved in implementing this goal.

```prolog
model _ antibody :-

setup_target(ModelProteinCode),
new_entity(m_structure, [ModelProteinCode], ModelStructure),
model_backbones(ModelStructure),
model_kabat_clusters(ModelStructure),
model_conserved_singletons(ModelStructure),
model_kabat_singletons(ModelStructure),
model_singleton_rotamers(ModelStructure),
model_conformational_rotamers(ModelStructure),
refine_model(ModelStructure),
write_protein_score(ModelStructure),
write_model_summary(ModelStructure, '_model'),
write_structure_pdb(ModelStructure, '_model').
```

Figure 3.1: Top level Prolog goal.
3.2 The Cost Function

The preceding sections indicate the importance of steric, volumetric packing and chemical interactions such as Hydrogen bonding, in stabilising a protein structure. If it is hoped that re-usable clusters of residues can be found in a structural database then it would be expected that these clusters would also demonstrate such favourable interactions and good packing.

So how would these requirements be encapsulated for a database search?

A conceptually simple approach would be to devise a cost function that can discriminate (at least approximately) between successive pairs of residue-residue interactions. Thus when searching the database, the cost function must be able to answer the question:

“Given a particular residue in the database, with which of its neighbours does it interact most favourably? Which third residue interacts most favourably with this initial pair? ... And so on, for higher order groupings.”

When modelling a new structure, similar questions would be:

“Given a particular model residue site, what side chain substitution would give the most favourable overall interaction?”

“Given a model structure, which residue has the worst overall interaction with its neighbours?”

In order to answer these types of question, the cost function must have the following properties:

1. It rewards good volumetric packing.
2. It rewards favourable chemical interactions.
3. It penalises steric clashes (van der Waals overlap).
4. It is easy to calculate.
5. It generalises from pairwise interactions to higher-order clusters.

This suggests that the cost function should be composed of three main elements, which together define the packing factor:

\[
Packing \ Factor = \frac{\text{Packing Gain}}{\text{Overlap Penalty}} \times \text{Chemical Bonus}
\]
More specifically, the packing factor between a pair of interacting residues \( i \) and \( j \) is defined as:

\[
F_{ij} = \frac{G_{ij} * B_{ij}}{P_{ij}}
\]  

(3.1)

Equation 3.1 increases with good volumetric packing and good chemical interactions but decreases in the presence of bad steric overlaps. As a rule of thumb, the chemical bonus \( B_{ij} \) is the most important, followed by the overlap penalty \( P_{ij} \). In the absence of chemical and steric effects the cost function is controlled by packing gain \( G_{ij} \). In order to get this behaviour, each individual term must be unity when it has no contribution to make. The next sections describe how these components of the cost function are defined.

### 3.2.1 Volumetric Packing

Each amino-acid residue has its own unique arrangement of side chain atoms. Although these atoms are joined by covalent bonds, they do have some restricted degrees of freedom in which they can move relative to each other by rotations about various bond axes within the side chain. Although in principle, such rotations can produce an infinite number of relative orientations, in practice we know that each side chain can be represented by just a handful of preferred conformations. Thus, in a sense, each residue can be considered to be part of a 3-dimensional jigsaw puzzle in which there are a small number of possible ‘side chain pieces’ that can fit at each particular residue site.

It would be extremely difficult to model the exact geometric packing of such complicated jigsaw pieces. Fortunately, the requirement of the cost function is that it should *discriminate* rather than *quantify* residue interactions. A good first approximation of how well any pair of residues pack together is to measure the amount by which the volume of the pair changes on going from residues considered separately to residues considered as a pair. This is illustrated in Figure 3.2.

In this project, the *pairwise packing gain*, \( G \), between a pair of volumes \( V_1 \) and \( V_2 \) which overlap with a volume of \( S_{12} \) is defined as:

\[
G_{12} = \frac{V_1 + V_2}{V_1 + V_2 - S_{12}}
\]  

(3.2)

The volumes \( V_1 \) and \( V_2 \) are determined by finding the smallest spheres
that just fully enclose all of the atoms of each residue. This is hard to solve analytically (if not impossible) although a *numerically exact* solution can be obtained iteratively. The procedure that does this is described in Appendix D and is implemented as a Fortran routine which can be called from Prolog. The overlap volume \( S_{12} \) can be obtained analytically, the derivation of which is given in Appendix E. Equation 3.2 has the property that it is unity if there is no overlap and increases to some well-behaved maximum (of around 2.0) as the spheres get closer together, which is precisely the behaviour required for the packing factor equation in 3.1.

Once residue bounding spheres have been calculated for Equation 3.2 they can be put to further use to quickly find atomic interactions. If the atoms of a residue were to interact simply as hard spheres, then two residues can only ever interact when their bounding spheres overlap. Also, any pair of atoms from these residues can only ever interact when an atom of one residue intersects the bounding sphere of the other residue. This is demonstrated in Figure 3.2. Thus pairwise atomic interactions can be found very easily without having to enumerate all possible atom pairs and without the need for a coordinate grid to help locate neighbouring atoms. If there are \( N \) atoms in one residue and \( M \) atoms in the other, then the small set of potentially interacting atoms can be found in \( O(N + M) \) distance comparisons. The few atoms in this set can then be checked explicitly for atom-atom overlap.

Of course, atoms are not simply hard spheres but can interact over short distances. However, if an additional capture range is added to each distance comparison, then the above arguments can still be used as a very fast way to search for interacting atoms. In fact, this principle is used throughout the system to detect, for example:
• Inter-atomic interactions between residues.

• Model residue neighbours.

• Inter-domain residue neighbours.

• Clusters of interacting residues.

3.2.2 Side Chain and Backbone Spheres

Rather than always treat the residue volume as a single unit, a residue is divided into a backbone volume and a side chain volume. This has a number of benefits. Since we are modelling side chain placement on a fixed backbone, sequentially adjacent backbone-backbone interactions will be invariant and so can be eliminated from the calculation.

![Diagram of side chain and backbone overlap volumes](image)

Figure 3.3: Schematic side chain and backbone overlap volumes of two residues, Tyrosine and Tryptophan. The shaded region indicates a small volume error.

Another reason for working with separate backbone and side chain volumes is that an overall spherical volume is quite a poor approximation of the actual volume of an amino-acid residue. When a side chain is substituted at a backbone location (either as a rotamer or as copy from a template), it is
the change in packing that is of interest and only the side chain conformation contributes to this change. Figure 3.3 illustrates pairwise residue overlap when distinct side chain and backbone spheres are used. The components of the residue-residue packing gain are given by:

\[
G_{SS} = \frac{S_1 + S_2}{S_1 + S_2 - S_1 S_2}
\]
\[
G_{SB} = \frac{S_1 + B_2}{S_1 + B_2 - S_1 B_2}
\]
\[
G_{BS} = \frac{B_1 + S_2}{B_1 + S_2 - B_1 S_2}
\]

and the overall packing gain of a pair of residues is defined as:

\[
G_{12} = 1 + (G_{SS} - 1) + (G_{SB} - 1) + (G_{BS} - 1)
\]  

Equation 3.3 demonstrates how all pairwise interactions in this system are added. Since each sub-term is 1 if there is no interaction, then subtracting 1 from each contribution allows the non-trivial components to be added; adding 1 to the result restores the original behaviour.

It should be noted that adding pairwise contributions in this way can introduce a small geometric error in the overlap volume. The central shaded volume in Figure 3.3 in which 3 spheres intersect is counted twice; once from each of the two pairwise overlaps. Although pairwise volumetric overlaps are geometrically exact, this is no longer the case for higher-order interactions. Thus the concept of Equation 3.2 has been converted from an exact volume ratio into Equation 3.3, which is a simple (but very reasonable) volumetric packing score.

The additional cost of calculating both backbone and side chain bounding spheres can be offset in the case of rotamers, since rotamer spheres need only be calculated once and saved in the rotamer library. Whatever coordinate transformation that would be required to copy a rotamer to the model could equally be applied to any spherical volumes associated with that rotamer. This is discussed further in Section 3.10.
3.2.3 Steric Overlap

Although there are many types of inter-atomic interactions at play within a large protein molecule, a good way to visualise these interactions is to think of the atoms rather like billiard balls with a sticky surface. Interatomic interactions are described to a very good approximation by the Lennard-Jones\(^1\) (or “12−6”) potential energy equation. The stickiness is due to a short-range attractive term of \(O(r^{-6})\), which tends to pull atoms together toward their equilibrium separation which is usually referred to as the effective von der Waals radius. Any attempt to compress atoms beyond this distance is countered by the very strongly repulsive term of \(O(r^{-12})\), which is responsible for the incompressibility of all solid matter. In protein modelling, atoms that are placed in such a way as to violate this incompressibility are usually referred to as causing **steric overlap**.

Any molecular modelling system that is to be thermodynamically correct must explicitly solve for the Lennard-Jones type of potential between interacting atoms. However, as with the volumetric packing score, the requirement of the cost function is to selectively differentiate between pairs of interactions rather than to accurately calculate a precise physical value.

The repulsive term of the Lennard-Jones potential is a very strong on-off potential. If this were to be used directly, it could differentiate between good and bad interactions, but not in the smooth way in which the packing gain varies. Instead, a softer parametric form was chosen to score the overlap penalty, \(P_{rs}\), between two atoms \(r\) and \(s\):

\[
P_{rs} = \begin{cases} 
1 & \text{if } B_{rs} \neq 1 \\
(R_r + R_s)^3/R_{rs}^3 & \text{if } R_{rs} < R_r + R_s \\
1 & \text{otherwise}
\end{cases} \tag{3.4}
\]

Here \(R_r\) and \(R_s\) are the effective van der Waals radii of the two atoms and \(R_{rs}\) is the inter-atomic separation of their nuclei. \(P_{rs}\) is 1 if there is no interaction and increases fairly rapidly as the two atoms are brought together. The overlap penalty is only calculated if there is no chemical bonus, \(B_{rs}\), as explained below.

If a pair of residues clash sterically, it is likely that more than just one pair of atoms in those residues will overlap. Following the approach used for summing volumetric packing factors, the overlap penalty between two interacting residues is defined as:

\(^1\)A good introduction to inter-molecular forces can be found in Atkins [1], for example.
\[ P_{12} = 1 + \sum_{r,s} (P_{rs} - 1) \] (3.5)

Where the sum is over all pairs of interacting atoms \( r \) and \( s \) in the residues. As before, the overall overlap penalty is 1 if there is no steric overlap and smoothly increases as the degree of steric overlap increases. The exponent of 3 was chosen so as to make Equation 3.5 vary more strongly than the corresponding packing gain (Equation 3.3), as any two residues are brought together.

Since residues are divided into side chain and backbone spheres, it is not necessary to enumerate all pairs \( r, s \) in Equation 3.5 because an atom can be eliminated if it does not intersect either of the other residue’s side chain or backbone spheres. Figure 3.2 illustrates this principle. Eliminating non-interacting atom pairs in this way is even more of a saving when chemical interactions are included, as described in the next section.

### 3.2.4 The Chemical Bonus

As mentioned in Section 2.1, Hydrogen bonds play an important role in stabilising the secondary structures of protein molecules. A Hydrogen bond can form between a proton donor, such as N-H or O-H and a proton acceptor, such as Oxygen (atomic orbital theory is required for a proper explanation) and this can lower the energy of the protein molecule by about 4 Kcal/mol, which is about 5% of the strength of a typical covalent bond. Since the cost function is intended to find stable clusters of residues, it should reward favourable interactions such as Hydrogen bonds in a similar way that it penalises bad steric clashes. In fact the presence of a Hydrogen bond can often involve bringing two atoms closer together than would normally be allowed by steric considerations, so that the bonus of finding a good chemical interaction must make a greater contribution to the cost function than the overlap penalty. In this system, if a chemical bonus is found for a pair of atoms, no overlap term is calculated.

Another important type of interaction within antibody molecules is the disulphide bridge. This is the formation of a covalent bond between the Sulphur atoms of two Cysteine residues which, for example, always appear at Kabat positions 23 and 88 in the VL domain. This bond plays an important part in stabilising the \( \beta \)-sheet of the framework region. As with Hydrogen bonds, the cost function should reward these disulphide bonds. This reward should be proportionately strong enough such that when searching for residue clusters,
any cluster that contains a Cysteine residue must also contain its disulphide partner.

Chemical interactions such as Hydrogen bonds and disulphide bridges have an equilibrium separation, analogous to the equilibrium separation implied by the effective van der Waals radii. The form of the chemical bonus term should try to honour this behaviour. However, if Hydrogen-bonding atoms are brought too close together, then this should be treated as a steric overlap.

With these ideas in mind, the chemical bonus, $B_{rs}$, between two atoms $r$ and $s$ is defined as:

$$B_{rs} = \begin{cases} 
(R_r + R_s)^4 / R_{rs}^4 & \text{if } D_{\text{min}} < R_{rs} \leq D_{\text{max}} \\
1 & \text{otherwise}
\end{cases} \tag{3.6}$$

Where $D_{\text{min}}$ and $D_{\text{max}}$ define the distance over which the bonus term is applied. $B_{rs}$ is 1 if there is no interaction and increases fairly rapidly as the two atoms are brought together. The exponent of 4 was chosen to be larger than the steric exponent in Equation 3.4, but otherwise its value is arbitrary. An exponent of 6 is used to reflect the additional importance of disulphide bonds.

Equation 3.6 is only applied to appropriate pairs of atoms that can participate in Hydrogen bonds or disulphide bridges. Similar but weaker parameters are also defined for certain aromatic atoms, since it is known that favourable ‘π-interactions’ can occur between aromatic ring systems [14]. All such interacting atoms are listed in a look-up table that tabulates residue atoms against their allowed type(s) of chemical interaction. It also lists the inter-atomic range over which the bonus can apply. Most of these ranges were taken from tables of the QUANTA package.

The table look-up for chemical interactions is probably the most expensive part of calculating the chemical bonus term. This search benefits by having applied the $O(N+M)$ search of atoms against side chain and backbone spheres to eliminate all non-interacting atoms.

As in the case of the volumetric packing term, chemical interactions between adjacent backbone atoms will be invariant. However, these backbone-backbone interactions are, in a sense, in competition with any favourable backbone-sidechain interactions that might be present. Thus all inter-atomic interactions are included in the chemical bonus and steric overlap calculations, except between the backbone atoms of adjacent residues.
3.3 The Rotamer Library

The rotamer library that is used in this project was donated by Desmet et al. [10]. This consists of the Ponder and Richards [23] rotamer library with corrections and extensions for the Arg, Lys and Met rotamers. The original library was given as a table of torsion angles. In his 1991 MSc. project, Georgios Lappas [18] converted the library into a list of explicit atom coordinates, formatted as a P/FDM load file. He produced these coordinates by back-calculating from the Desmet $\chi$-angles and some bond lengths ‘taken from the PDB’ [2], although the precise details were not given.

![Diagram of rotamer library](image)

Figure 3.4: The $C_\alpha$-C-N coordinate system. (a), (b) and (c) represent useful transformations.

This P/FDM load file is also used in the current project. However, each rotamer was observed to be defined in a somewhat arbitrary coordinate system. Since it was planned to make extensive use of pre-calculated rotamer spheres, it became clear that a consistent coordinate system would greatly simplify the transformations that would be required to map rotamers, residues and their spheres from one coordinate frame to another. Thus the atom coordinates of each rotamer were transformed to a consistent reference frame in which the $C_\alpha$ atom is at the origin, the carbonyl Carbon atom is on the negative $z$ axis and the Nitrogen atom makes an angle of 0° with the positive $x$ axis. This is illustrated in Figure 3.4.

Without going into geometric detail, this scheme allows three simple Prolog
predicates to perform every coordinate transformation that is needed in the current system (except for backbone domain fitting):

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_residue_transform(Ra,Ta)</td>
<td>% (a)</td>
</tr>
<tr>
<td>get_residue_transform(Rb,Tb)</td>
<td>% (b)</td>
</tr>
<tr>
<td>get_rotamer_transform(Ra,Ta_inv)</td>
<td>% (a)^{-1}</td>
</tr>
<tr>
<td>get_rotamer_transform(Rb,Tb_inv)</td>
<td>% (b)^{-1}</td>
</tr>
<tr>
<td>get_residue_residue_transform(Ra,Rb,Tab)</td>
<td>% (a) * (b)^{-1} = (c)</td>
</tr>
<tr>
<td>get_residue_residue_transform(Rb,Ra,Tba)</td>
<td>% (b) * (a)^{-1} = (c)^{-1}</td>
</tr>
</tbody>
</table>

Figure 3.5: Three Prolog predicates for (nearly) all coordinate transformations. Ra and Rb can be residue or rotamer objects. The output is a pointer to a C transformation matrix. The transformations (a, b, and c) refer to Figure 3.4.

The predicates have been written to work with both residue and rotamer objects and the residues need not belong to the same protein structure (i.e. they have atom coordinates with a different origin). In the rest of this report, it will be implicitly assumed that coordinate transformations are applied as necessary, to bring an object into the appropriate coordinate system.

Figure 3.6: Schema diagram for the rotamer database module.

The modified rotamer library holds all atomic rotamer coordinates in this ‘standard’ Cα-C-N coordinate system. In addition, a side chain sphere, a backbone sphere and the all-atom rotamer sphere (somewhat misleadingly
\begin{verbatim}
declare rotamer_sphere  
declare x(rotamer_sphere)  -> float  
declare y(rotamer_sphere)  -> float  
declare z(rotamer_sphere)  -> float  
declare r(rotamer_sphere)  -> float  

declare rotamer_atom     
declare x(rotamer_atom)  -> float  
declare y(rotamer_atom)  -> float  
declare z(rotamer_atom)  -> float  
declare accessibility(rotamer_atom) -> float  

declare amino_acid      
declare name(amino_acid) -> string  
declare bounding_sphere(amino_acid) -> rotamer_sphere  
key_of amino_acid is name  

declare rotamer         
declare conformer_of(rotamer) -> amino_acid  
declare number(rotamer)  -> integer  
declare sidechain_sphere(rotamer) -> rotamer_sphere  
declare backbone_sphere(rotamer) -> rotamer_sphere  
declare residue_sphere(rotamer) -> rotamer_sphere  
key_of rotamer is key_of(conformer_of), number  

declare rotamer_atom(rotamer,string) -> rotamer_atom  

declare atom_type       
declare part_of(atom_type) -> amino_acid  
declare name(atom_type)  -> string  
key_of atom_type is key_of(part_of), name  

declare quanta_atom_type 
declare quanta_code(quanta_atom_type) -> integer  

declare quanta_parameters 
declare quanta_code(quanta_parameters) -> integer  
declare vdw_radius(quanta_parameters) -> float  
key_of quanta_parameters is quanta_code  
\end{verbatim}

Figure 3.7: Daplex schema definition of the \textit{rotamer} database module.
atom(rotamer,string) — >> atom
name(rotamer) — >> string
num_rotamers(amino_acid) — >> integer
atom_vdw_radius(string,string) — >> float

Figure 3.8: Method function signatures for the rotamer database module.

called the residue sphere) are also saved for each rotamer. For each amino_acid instance an overall bounding sphere is saved. This bounding sphere encloses the space that is swept out by any of the rotamers for a given amino-acid and hence defines the volume in which the atoms of a residue are virtually always going to be found. A schema diagram for the rotamer module is shown in Figure 3.6. The Daplex declarations that define the schema are shown in Figures 3.7 and 3.8.

The rotamer module is also a convenient place to store information about the atoms of each amino-acid. The atom_type class contains instances of each type of atom that is to be found in amino-acids. Atom names follow the PDB convention, thus $C\alpha$ becomes ‘CA’, etc. Most atom types have an associated parameter, the van der Waals radius, which is taken from a QUANTA parameter file. Thus the class quanta_atom_type forms a sub-class of atom_type. The QUANTA package uses the notion of an atom code number. Rather than laboriously edit the QUANTA data to implement an explicit relationship function, the value of quanta_code is used to implement the relationship rather like in a traditional relational database. A method function supplies a default value if no stored value is available. Since atomic parameters are referenced frequently, they are read from the database just once and cached for subsequent use.

The rotamer database module also has several method functions associated with it. These are shown in Figure 3.8. It can be seen from the schema that the rotamer class does not have a name attribute. This is supplied by the name method function, which locates the required name from the appropriate amino_acid instance. The atom method function is used to locate atomic coordinates, given a rotamer object and the name of an atom type. Both the name and atom method functions have been deliberately chosen to overload the corresponding function names which are defined over residue objects in the antibody database. This overloading allows a great deal of code re-use. Any predicates that are designed to operate on the coordinates of residue atoms, such as the transformation predicates in Figure 3.5, will work equally well

### 3.4 The Model Database

In principle it would be possible to represent a model protein structure entirely using ‘in-memory’ Prolog predicates. However, it was expected that the some of the proposed modelling steps might be time-consuming and that debugging and development would be faster if the system were to be built and tested incrementally. Saving intermediate modelling results to a permanent database, similar to the original `antibody` database, is the simplest way to do this. This also has the advantage that the resulting model can be queried using most of the existing `antibody` Daplex queries.

![Diagram](image)

Figure 3.9: Schema diagram for the `model` database module.

The schema for the modelling database is shown in Figure 3.9 and the Daplex schema declarations are shown in Figure 3.10. It can be seen that this is very similar to the `antibody` database, although the `m_residue` class has some additional properties, including the recursive `neighbour` relationship. As with the `rotamer` database, the `atom` method function allows the atoms of `model` residues to treated in the same way as atoms of the `antibody` database.
declare m_structure  
    declare protein_code(m_structure)  
    key_of m_structure is protein_code

declare m_chain  
    declare component_protein(m_chain)  
    declare component_id(m_chain)  
    key_of m_chain is key_of(component_protein), component_id

declare m_residue  
    declare pos(m_residue)  
    declare name(m_residue)  
    declare has_component(m_residue)  
    declare kabat_position(m_residue)  
    declare source(m_residue)  
    declare status(m_residue)  
    declare confidence(m_residue)  
    declare neighbour(m_residue)  
    key_of m_residue is key_of(has_component), pos

declare m_atom  
    declare x(m_atom)  
    declare y(m_atom)  
    declare z(m_atom)  

declare accessibility(m_atom)  

declare m_sphere  
    declare x(m_sphere)  
    declare y(m_sphere)  
    declare z(m_sphere)  
    declare r(m_sphere)  

declare m_ig_domain  
    declare domain_structure(m_ig_domain)  
    declare domain_type(m_ig_domain)  
    declare name(m_ig_domain)  
    declare ig_domain_chain(m_ig_domain)  
    declare start(m_ig_domain)  
    declare end(m_ig_domain)  
    declare source_protein(m_ig_domain)  
    declare source_domain(m_ig_domain)  
    declare domain_sphere(m_ig_domain)  
    key_of m_ig_domain is key_of(domain_structure), name

Figure 3.10: Daplex schema definition of the model database module.
3.5 Backbone Construction

The backbone of an antibody structure is modelled in a relatively simple way.

Given the Kabat sequences of the VH and VL domains to be modelled, the antibody database is searched for protein structures whose Kabat sequences give the highest sequence match with the model sequences. The sequences are matched by counting the number of residues that are identical (or conserved) between the model sequence and the database sequences. A sequence that contains insertions or deletions, relative to the model sequence, may be used but only if there is no other choice. It should be noted that this approach ignores the structural assignment work of Chothia and Lesk. However, it is sufficient for the present purpose of exploring a knowledge-based approach to side chain modelling.

The best sequence matches for the two domains are often to be found in different protein structures. In this report, these will be referred to as the backbone donors. Since atomic coordinates are normally given relative to the coordinate frame in which the protein structure was originally determined, this means that coordinates of the backbone atoms of the two domains will be incompatible. This can be resolved by applying McLachlan’s least squares fitting algorithm [21] to selected atoms of the two protein structures. The coordinates of the $C_\alpha$ atoms of those residues that are conserved across the two donor VH domains are used to define this transformation. The fitting algorithm finds the coordinate transformation (a translation and a rotation) that would superpose the VH $C_\alpha$ atoms of one domain onto those of the other domain. This transformation will also take the atoms of the VL donor domain into the correct position relative to the VH domain of the other donor. This is illustrated in Figure 3.12.

Thus the model structure is initialised by copying the VH backbone atom coordinates from the VH donor and by copying the transformed VL backbone
atoms from the VL donor. Model residue neighbours are found by applying a modified version of the *neighbours search* (described in the next Section), in which bounding amino-acid spheres are placed at each model residue site in order to record pairs of residues which *could* interact, once their side chains are in place.

### 3.6 Residue Neighbours

In order to find clusters of interacting residues, it is necessary to determine the *neighbours* of each residue in the database of known antibody structures. Since the *antibody* database is essentially static, the search for residue neighbours is done only once and the result is saved in a separate permanent database module. This database stores only the minimum amount of information needed to record those neighbouring residue relationships.

---

**Figure 3.13:** Schema diagram for the residue *neighbours* database module.

The schema diagram for the *neighbours* database is shown in Figure 3.13.
and the Daplex schema definition is listed in Figure 3.14. It should be noted that unlike the main *antibody* database, the key of the *n_residue* class has been changed to [protein_code, domain_name, kabat_position], since the Kabat numbering scheme is more useful than absolute chain positions when searching for clusters of neighbouring residues. The *neighbour* function on *n_residue* defines a recursive relationship since a residue can have many neighbours, each of which is also a residue.

As in the *rotamer* module, the *neighbours* database module also defines name and atom method functions on *n_residue* objects. These functions access the original *antibody* database via the *residue* relationship to retrieve the appropriate values. This greatly simplifies the process of copying clusters of neighbouring residues into the model structure.

```plaintext
declare n_sphere
    declare x(n_sphere) -> float
    declare y(n_sphere) -> float
    declare z(n_sphere) -> float
    declare r(n_sphere) -> float

declare n_structure
    declare protein_code(n_structure) -> string
    declare num_units(n_structure) -> integer
key_of n_structure is protein_code

declare n_ig_domain
    declare name(n_ig_domain) -> string
    declare domain_sphere(n_ig_domain) -> n_sphere
    declare domain_structure(n_ig_domain) -> n_structure
key_of n_ig_domain is key_of(domain_structure), name

declare n_residue
    declare kabat_position(n_residue) -> string
    declare has_component(n_residue) -> n_ig_domain
    declare neighbour(n_residue) -> n_residue
    declare residue(n_residue) -> residue
key_of n_residue is key_of(has_component), kabat_position

declare pairwise_factor(n_residue, n_residue) -> float
```

Figure 3.14: Daplex schema definition of the *neighbours* database module.

Residue neighbours are found by examining the residues in each antibody VL and VH domain in turn. Within each domain, the appropriate amino-acid
bounding sphere is placed at each backbone residue site. Since an amino-acid bounding sphere encloses the volume swept out by all possible rotamers of that type, these spheres define the steric interaction range of each residue. A small additional capture range of 0.75Å is added to define the steric and chemical interaction range. This list of amino-acid spheres is searched for pairs of spheres that intersect\(^2\) (a coordinate grid would help reduce this search, but this was not implemented, since this search is only applied once for each domain), resulting in a list of potentially interacting residue pairs. The \textit{packing factor} is evaluated for each pair of residues and those pairs that have a non-trivial packing score (anything other than 1) are saved as neighbours under the \textit{n_residue} class.

In an antibody Fab fragment quite large surface areas of the VL and VH domains can come into contact at their interface. Any residues that interact across this interface are also saved in the \textit{neighbours} database. Candidate interface residues are found by calculating a \textit{domain sphere} that bounds all of the residue spheres in each domain and by eliminating those residues that do not intersect the other domain’s bounding sphere. The remaining cross-domain residue neighbours are located and saved, as described above.

It takes around 1 hour to find and load all of the neighbouring residues in each protein structure. Most of this time is spent evaluating the \textit{packing factor} cost function. Consequently the packing scores for residue pairs are also saved to eliminate the overhead of ever having to recalculate them. These are used in the subsequent search for residue clusters.

### 3.7 Residue Clusters and the Model Mask

The \textit{neighbours} database contains details of all pairwise residue interactions in the \textit{antibody} database, as measured by the cost function. This section explains how such pairwise interactions are resolved into clusters of residues and how these clusters are placed in the model.

In this system, a \textit{cluster} of residues is defined as a group of \(N\) or more residues from the \textit{antibody} database which have a pairwise interaction with at least one other member of the cluster for which the \textit{packing factor} is greater than some threshold and where each member of the cluster corresponds to an unplaced residue in the model. \(N\) is typically 2 or 3. This definition suggests a way to search efficiently a database of many thousands of neighbouring residue

\(^2\)Searching for intersecting spheres is the only search in the system that is \(O(N^2)\), where \(N\) is appreciably large (around 100).
interactions. If the interaction threshold is high, then many residues are immediately eliminated as candidate members of a cluster. If many residues have already been placed in the model, then the search is restricted to just those that correspond to the unplaced residues.

A convenient way to keep track of which model residues still need to be placed is with the *model_mask*. Initially, the model mask is simply the Kabat sequence for the model structure (in fact, the model mask has two sequences since there are two domains in the model structure). When a residue is placed in the model, the corresponding element of the model mask is replaced with a dash. Thus the non-dash elements of the mask represent unplaced model residues. Since the elements of the model mask correspond to residue locations under the Kabat numbering scheme, they can be used to restrict or *mask* the search for antibody database residues that could fill an unplaced model site.

The search for clusters of neighbouring residues proceeds iteratively. To begin with, there are many unplaced residues in the model mask, so a high interaction threshold is specified in order to restrict the number of residues that fulfill the cluster membership criteria. This corresponds to requesting only the most favourable residue interactions, such as those of disulphide bridges and strong Hydrogen bonds. As placement proceeds, the threshold is reduced to select the more weakly interacting residues, although it is the model mask that now restricts the search. This scheme works even better when the minimum cluster size $N$ is also gradually reduced from 4 to 2. Conceptually, this corresponds to building a network of strongly interacting residues and then filling the remaining spaces with tightly packing groups, although in practise the behaviour is less clear-cut.

Clusters are found by searching the *neighbours* database, one protein at a time, for pairs of residue neighbours in each protein that satisfy the cluster criteria. This produces a list of interacting residue pairs. If the head of this list is treated as the beginning of a cluster then the rest of the list can be searched for residues which are neighbours of some member of that cluster. This is repeated recursively until all clusters have been found. The net *packing gain* of a cluster is calculated by summing the pairwise contributions in the usual way. This is saved along with the cluster *members* and the sphere that just bounds the $C_\alpha$ atoms of each of the members (the *alpha_sphere*).

It is convenient to save clusters as they are found, in the *clusters* database module. This is because each protein structure can contribute many candidate clusters and it becomes burdensome to manipulate complex data structures in the argument lists of Prolog predicates (or such argument lists of any other language, for that matter). However, this database becomes obsolete at the
Figure 3.15: Schema diagram for the temporary clusters module. The dotted line represents a relationship with an external database.

---

```
declare alpha_sphere
  declare x(alpha_sphere) - > float
  declare y(alpha_sphere) - > float
  declare z(alpha_sphere) - > float
  declare r(alpha_sphere) - > float

declare protein_cluster
  declare protein_code(protein_cluster) - > string
  declare num_clusters(protein_cluster) - > integer
key_of protein_cluster is protein_code

declare cluster
  declare source_structure(cluster) - > protein_cluster
  declare num_members(cluster) - > integer
  declare number(cluster) - > integer
  declare gain(cluster) - > float
  declare alpha_sphere(cluster) - > alpha_sphere
  declare member(cluster) - > n_residue
key_of cluster is key_of(source_structure), number
```

---

Figure 3.16: Daplex schema definition of the clusters database module.
end of each cycle of the cluster search and so it is defined as a temporary module. The clusters database schema is shown in Figure 3.15 and the Daplex schema declarations are shown in Figure 3.16.

The cluster size (num
members), gain and alpha
sphere are used to select the best clusters to place in the model. As would be expected, the general principle is to place the largest and highest-scoring clusters first. However, an explanation of the algorithm is easier if the following two concepts are defined:

- A cluster is compatible if all of its residues correspond to unplaced residues in the model. Initially, all clusters are compatible.

- A cluster is viable if all of the distances between the Cα atoms of its residues and the centre of its alpha
sphere are within some tolerance, say 0.5Å, of the corresponding distances in the model (when the alpha
sphere is translated to the model coordinate system)\(^3\). This does not imply that the cluster residues would fit the model with the same precision, but with the high homology of antibody structures, this is a reasonable (and easy) test to perform. The root-mean-square (RMS) of the distance differences is a measure of the relative viability a cluster.

Clusters are sorted by size and, within groups of the same size, by each cluster’s gain. The largest cluster with the highest gain becomes the favourite to place in the model. If the favourite is not viable (i.e. if it does not fit the model), the search re-starts on the rest of the cluster list.

If the favourite is the single largest cluster, then all of its residues are copied to the model. Otherwise, all of the other clusters of the same size become contenders,\(^4\) if they correspond to the same cluster site as the favourite and if they are viable. The favourite and contenders are scored according to their gain and relative viability:

\[
\text{Score} = \frac{\text{Gain}}{1 + \text{RMSviability}}
\]  

(3.7)

The residues of the highest-scoring cluster are copied to the model and the model mask is updated to record this new state of the model.

\(^3\)On reflection, it might be wise to include the Carbonyl Oxygen atoms in this test to exclude backbone peptides that may be ‘flipped’ relative to the model backbone.

\(^4\)Please excuse the boxing metaphor!
Whenever a residue is copied to the model in this way, the source protein and cluster size are also saved for future reference. The cluster size is used subsequently as an estimate of the confidence with which a residue was placed.

Any clusters which have become *incompatible* with the new model mask are discarded and the search re-starts on the remainder of the cluster list. When the cluster list is exhausted, a new cluster search starts with a lower *packing factor* threshold value. This process is repeated 5 times and can be expected to place around 60% of all model residues.

### 3.8 Conserved Side Chains

As indicated above, the cluster search may not find suitable residue substitutions for every residue site in the model. The next best strategy is to attempt to copy the side chains of *conserved* residues from the antibody database.

For any given residue in a multiple sequence alignment, the residues at the corresponding position in other sequences of the alignment are said to be *conserved* if they are of the same residue type. By homology, these residues are very likely to occupy structurally comparable positions.

Copying conserved residues happens in two steps. Firstly, any unplaced model residue that is conserved in the corresponding residue position of the *backbone donor* protein domain is copied to the model. Since the model backbone is a copy of the donor's backbone, this *assumes* that the backbone donor continues to offer the most topologically equivalent environment, even after some side chains have been placed by the cluster search.

Secondly, all other protein structures are searched for conserved residues at unplaced model residue sites. If this finds only one residue for a given model site, then that residue is copied unconditionally. If several candidates are found, then the *local packing gain* is evaluated as if each candidate in turn had been placed at that site (only residue coordinates need to be transformed to do this). The *local packing gain* is the sum of all pairwise packing factors between that residue and its immediate neighbours in the model. The residue substitution that would cause the highest local packing gain is copied to the model. At this stage, it’s quite possible that not all of the side chains of the immediate neighbours are in place. Thus the use of the local packing gain could cause the wrong choice to be made. Nonetheless, this is using whatever partial information that is available to guide the choice at every step, following Brucoleri’s *directed search* principle.
This procedure is found to place around 30% of model residues.

The confidence factor for each model side chain which is placed by homology is set to a value that is lower than that used for any cluster, but higher than any of the values that are assigned for side chains whose conformations must be derived from the rotamer library.

3.9 Rotamer Placement

Following the database placement of side chains, any model residues that do not yet have a side chain are assigned a default rotamer conformation from the rotamer library. Since rotamers are numbered in the approximate order of the frequency with which they occur in known structures, the first rotamer of each amino acid is copied to every unplaced model residue site.

This completes the initialisation of the model.

The confidence factor of a model residue whose side chain derives from the rotamer library is set according to the number of possible rotameric conformations for that residue (many rotamers implies low confidence). These values are usually less than the values assigned to residues that were copied from the antibody database. The exceptions are Gly, Ala and Pro. Since these have only one conformation, they are assigned a high confidence factor in order to exclude them from the model refinement step which is described in the next section.

3.10 Refining The Model

Once all side chains have been placed on the model backbone, the quality of the side chain assignments is determined by calculating all pairwise residue packing factors for all neighbouring model residues. This is essentially the same procedure as is used when building the neighbours database.

Those pairs of interacting residues whose packing score is less than some threshold are collected as candidates for refinement. The threshold is chosen to be less than any of the values observed for the pairwise interactions of the residues of real antibodies. Currently this value is 0.3. Each residue in the candidate refinement list is scored according to the frequency with which it appears, the confidence factor of its initial placement and the number of
rotameric conformations that exist for that type of residue. Frequently appearing residues with few rotamers and a low confidence factor are considered to be the best candidates to improve the model with least effort.

Each of these candidate refinement residues is examined in turn by successively placing each of its rotamers at that residue site (similar to the selection of conserved side chains) and evaluating the local packing gain that results from that residue's interactions with its immediate neighbours. The side chain conformation of a residue is replaced by its highest-scoring rotamer if this would cause an improvement over its current side chain assignment.

Repeating this cycle causes further refinements, although currently the procedure is applied just twice. This is the most time-consuming part of the system and its observed that relatively few (often around 20) residue conformations are refined in this way. There are a number of possible reasons why so few refinements are applied:

- The quality of the initial data-based placements leave little scope for improvement.
- The initial data-based placement was incorrect and led to a “dead-end”.
- The rotamer search is too crude: there is no opportunity to fine-tune the torsion angles.
- The fixed backbone assumption constrains the quality of the solution.

These are issues which the current system cannot assess and so it must now hand control to a human expert (it stops!). It does, however, produce some output which may be of use to a human modeller. A summary file lists the source of each residue and its placement confidence factor, the warnings file lists those residues and their atoms which are involved in steric clashes and the pdb file may be imported into a molecular graphics package to visualise the model. A display selection file is also written, which may be used in QUANTA to display only the sterically clashing residues.

The next section looks at some of the results that are obtained with this system.
Chapter 4

Results

4.1 The P53 Model

The P53 antibody Fv fragment\(^1\) was modelled using the current system. This consists of a light chain of 113 residues and a heavy chain of 116 residues. The backbone modelling procedure selected the VH chain from 2HFL as the best VH donor, with no insertions or deletions in the sequence match. The best VL domain donor was the VL domain of 2IGF (114 residues), although this had an insertion, relative to P53, at the end of the sequence match, so the final donor residue was ignored to give a chain of the correct length.

The cluster placement algorithm placed 154 residues (67\%) in 5 iterations of the cluster search. The largest cluster in the VL domain contained 10 residues and the largest VH cluster contained 8 residues. Most clusters contained 3 or 4 residues. Two clusters spanned the VL and VH domain interface and 8 of the clusters spanned the framework and CDR regions (as defined by the Kabat assignments).

Every VH domain residue that is conserved relative to the backbone donor was found to have been placed by the cluster search. The 11 VL domain residues which were not placed as clusters but which were conserved in the backbone donor were copied to the model. The search for conserved residues in the remaining antibody structures located a further 46 side chains, leaving just 18 side chains (8\%) which were assigned default rotamer conformations.

The refinement procedure found 78 side chains which were involved in at

\(^1\)The sequence of the P53 antibody was provided by Prof. David Lane, University of Dundee.
least one bad steric contact. The first refinement cycle replaced 21 side chains by a rotameric conformation from the rotamer library. It replaced 4 side chains in the final second cycle. Of the 81 residues which were originally placed by clusters of 4 or more residues, only 3 were replaced by refinement, whereas 8 of the 18 initially placed rotamers were replaced in this way.

The system found that there remained 21 bad pairwise residue interactions in the final model, 12 of which were between adjacent residues. There were just 3 close atomic contacts of less than 1Å, the worst being 0.72Å between $C_\gamma$ of Leucine at chain position 41 in the VL domain and the backbone Nitrogen of Leucine at position 42 in the same chain. There were around 30 atomic contacts of between 1Å and 2Å, around 80 contacts of between 2Å and 3Å and about 45 contacts of over 3Å but less than the van der Waals equilibrium separation. Adjacent residues were responsible for the majority of these bad contacts.

As might have been expected, there are fewer suitable database candidates for the CDR than for the framework region (around 30% of the 37 residues in the structural CDR had to be built from rotamers, compared to around 15% for the model as a whole, after rotameric refinement). However, the locations of bad residue contacts are very evenly distributed throughout the structure.

Figure 4.1: Ball and stick representation of the model P53 Fv fragment. Atoms are colour-coded by cluster size. All rotamer atoms are in orange.
A ball and stick representation of the model is shown in Figure 4.1, in which all atomic radii are artificially reduced to allow a better view of the interior of the molecule. Each residue is colour-coded according to how it was initially placed. Clusters of less than 4 residues (including conserved singletons) are coloured in blue. Rotamers are coloured in orange. It can be seen that most of the rotamers appear on the surface the antibody fragment.

Figure 4.2: Space-filling representation of the model P53 Fv fragment. Backbone atoms are in white, atoms from clusters are in yellow, atoms placed by homology are in blue and rotamers are in red.

The same view is repeated in Figure 4.2 except that the atoms fill space according to their van der Waals radii and are colour-coded by the type of side chain assigned after refinement. Since there are relatively few yellow atoms (clusters) on the surface, yet 68% of residues derive from clusters, this would indicate that the majority of the cluster residues are at the interiors of the domains.

The protocol in Appendix C notes that the Tryptophan at Kabat position 89 in the light chain caused some problems. This is a bulky side chain, located at the interface between the two domains. It would appear that this system finds a reasonable set of conformations for nearly all of the residues at the domain interface, although visual inspection (not shown) does show some crowding between Tryptophan and the Asparagine at Kabat position 34.
Figure 4.3: Quality-checking the worst pairwise interactions in the P53 model, using the same colours as in Figure 4.2. The clash of Leucine’s $C_\gamma$ with the backbone is indicated.

Although the predicted structure of P53 would appear to be plausible, a crystallographic structure has not been determined for this antibody and so it is difficult to determine the accuracy of the model. Figure 4.3 shows just those residues which the system has flagged as being involved in bad steric contacts. This can be a useful way to quality-check the model. Close contacts such as these can only be judged properly when the model is interactively rotated in space. Most apparent contacts are an artifice of the current viewing angle and are seen to be relatively moderate upon rotation.

4.2 The 6FAB Model

A good test of the side chain placement algorithm is to place side chains onto the backbone of a known structure from the *antibody* database in the usual way, but to *exclude* all of the backbone donor side chains from being used in the model. The 6FAB Fv fragment was modelled in this way. In this exercise, 110 side chains (48%) were placed as clusters and 105 side chains (46%) were copied from conserved residue positions. 14 side chains (6%) were copied from the rotamer library. Many large clusters were found, but were eliminated due
to a poor match with the backbone geometry.

On refining the model, 61 bad pairwise residue contacts were reduced to 19 pairs after 15 rotamer replacements. This gives just 4 atomic contacts of less than 1Å and a spread of other near contacts which is similar to the P53 model. However, in contrast to the P53 model (since the 6FAB model uses the true backbone conformation), 36% of the reported residue contacts involved the 16% of residues that make up the loops of the structural CDR, despite the fact that fewer rotamers were used to build the 6FAB CDR loops (25%, after refinement).

Figure 4.4: 6FAB side chain skeleton comparison. Model atoms are in yellow and database atoms are in blue.

A comparison between the 6FAB model and the known PDB structure is shown in Figure 4.4. This is in the same orientation as the space-filled image shown in Figure 2.4. Backbone atoms are not shown since they are identical. As can be seen, there tends to be good agreement towards the interior of the molecule, with more variability in the conformations of the surface residues. Figure 4.5 zooms in on a selected portion of the VH domain, to examine the match in more detail. This shows that there is often very close agreement between the model and the real structure, which is quite remarkable considering that most side chain atoms were copied unchanged from the *antibody* database.
Figure 4.5: Zoomed portion of Figure 4.4.
Chapter 5

Conclusions

5.1 Evaluation

5.1.1 The System

The system appears to give very plausible results in a reliable way. The P53 and 6FAB models give comparable results in terms of the system's quality checks. Visual inspection of the modelled structures shows that most of the contacts that are reported are not particularly severe and that probably only a few side chain conformations would need to be resolved by human intervention. Certainly, there are no gross errors, but it's clear from the 6FAB trial that there is scope for further automatic refinement.

As with other approaches (e.g. Sutcliffe et al. [28], Laughton [19]), this system is less accurate at the surfaces of the domains. The tight packing bias of the cost function tends to pull surface side chains towards the body of the domain, whereas in the crystal structure the surface residues are solvated and can adopt a more open conformation.

The refinement procedure is rather simplistic: lack of time meant that only a small amount of effort went into this part of the system. For example, it looks only for substitutions that give an immediate improvement in badly strained interactions. It does not try to improve poor but acceptable interactions nor does it distinguish between sidechain-sidechain and sidechain-backbone interactions. This could account for some of the clashes observed. For example in the P53 model more than half of the steric clashes were between the side chain and backbone atoms of adjacent residues. Increasing the penalty for
sidechain-backbone clashes would probably help to improve the model.

Many authors in the literature measure the accuracy of their models in terms of RMS distance deviations from the crystal structure (e.g. Wilson et al. [30]) or as the percentage of $\chi$ angles within 30–40% of the known values (e.g. Dunbrack and Karplus [12], Laughton [19]). Both of these measures are simple and easy to apply, but they are not necessarily a strong indication of the quality of the model. However, this project has focussed on producing a plausible solution, along with a suggestion of where there may be badly-solved regions. Consequently, numerical comparisons are probably not appropriate (although tempting!). Plausibility, however, is a valuable quality in an expert system. Biologists often think of particular antibodies as being ‘human’ or ‘murine’, for example. The knowledge that a particular side chain was copied from a specific source may be more important to the Biologist than some numerical measure of the model’s ‘accuracy’. On the other hand, having a list of potential problem residues should greatly assist any manual refinement that may be necessary and would provide a good starting point for any further automatic procedures.

Although around 20 ‘bad’ pairwise interactions remained in each of the model structures, this is actually a very encouraging result. In an antibody Fv fragment there are around 230 residues and, using Laughton’s figure [19] of 2.5 neighbours per residue, this means that there are about 600 residue pairs to be considered. On this basis, it could be argued that the system successfully places a very large proportion of side chains in a way that is highly compatible with the given backbone.

5.1.2 P/FDM

P/FDM was found to be very easy to use. The data-independent functional form of P/FDM queries meant that there was often a tendency to forget that physical databases were actually involved. Good analysis is still required in data modelling, although its much easier to make subsequent schema changes than is the case with the relational model. For example the temporary clusters module was originally a part of the private neighbours database. Part of the way into the project there was a change of plan so as to find candidate clusters ‘on the fly’. This was easily implemented by changing the storage status of the appropriate entities.

The notion of method functions in P/FDM was also found to be a powerful feature. For example an atom method function in each database concealed the details of where residue and rotamer atom coordinates were physically located.
Even with several different database schemas in operation, it was often possible to use the same generic code to operate on atomic coordinates, regardless of design choices for the data model.

The hierarchical nature of protein structure can be represented easily in the functional model, whereas the set-based relational model offers no explicit mechanism with which to represent such hierarchies. I personally found that the functional form of database queries was extremely natural and appropriate for this type of project: It would be absurd to have to formulate a relational query simply to find the coordinates of a given atom, or to find the next residue in a sequence.

Although P/FDM is very good for representing relationships and queries on those relationships, the support for aggregate data types is rather weak (as it is in traditional relational databases). One possible extension to P/FDM might be support for a list type, similar to the existing value_entity. I frequently found that I wanted to write 'plural' calls such as

\[
\text{get_fnval} (\text{residue\_atoms}, [\text{Residue}], \text{AllAtoms})
\]

Indeed, one can do so, if the type-checking in P/FDM is bypassed by declaring residue\_atoms as a method function that delivers a string result. Of course, it is always possible to express this as:

\[
\text{findall} (\text{Atom}, \text{get_fnval} (\text{atom}, [\text{Residue}], \text{Atom}), \text{AllAtoms})
\]

but this would be difficult for P/FDM to optimise, since get_fnval cannot see the context from which it is being repeatedly called. But this is a rather weak example compared to what might be required if P/FDM ever had to host a graphical interface for protein structure visualisation. This must call for a much tighter coupling between the database and the application\footnote{For example, the database might pass to the application an aggregate identifier, which would grant the application code direct memory access to an array of atom coordinates.}, if interactive graphics performance is required.

Nonetheless, I am convinced that the functional data model is conceptually superior to the traditional relational model and it should be only matter of time before we see commercial implementations which are competitive alternatives to the current relational systems.
5.1.3 Prolog

Prolog's most remarkable features are its backtracking and clause unification mechanisms. It takes some time to get used to these, although once mastered its no less powerful than conventional procedural languages. Its declarative natures means that a problem can be expressed clearly without the syntactic baggage of control statements and type declarations.

Although Prolog can be 'compiled', it still needs a Prolog kernel to interpret it and it still has the performance of an interpreted language. Even with call-outs to Fortran and C, it was clear in this project that the system would have run much more quickly had the cost function been coded entirely procedurally. On the other hand, if a complex system such as this one can be developed in about 10 weeks and takes 10 hours to run, this is more time-efficient than a system that runs in 10 minutes but took 6 months to code. People's time is usually more valuable than CPU time and bottlenecks can always be hand-optimised. Thus, in my opinion, Prolog can be a surprisingly appropriate language, even for computational problems.

5.2 Future Work

Probably the most obvious weakness of the current system is in the way it creates the backbone structure. Although this was beyond the scope of the current project, it would be essential for a real antibody modelling system to choose the best possible backbone conformation prior to side chain placement. In particular it would allow splicing of the most appropriate conformation for each of the the hypervariable loops\(^2\). These are the regions of most interest to a practising Biologist, since it is these regions which confer antigen specificity.

Another weakness is in the final rotamer refinement, which essentially uses a blind conformational search. Although many side chains are well placed by the database search, using rotamers to refine the badly placed side chains takes about 80% of the system's execution time. Although it gives an indication of the benefit that has been obtained by using the database, this rotamer search could be substantially improved. For example, applying small changes to the \(\chi_1\) and \(\chi_2\) torsion angles of crowded residues, could give significant movements of the 'tail-ends' of residues, particularly in the case of the large aromatics such as Phenylalanine and Tyrosine. These movements would be more selective, however, than the very large change implicit in some of the

\(^2\)Dr. Kemp is currently working on such a modification.
rotameric representations of these conformations (e.g. \( \chi \)-angles that vary by \( \pm 180^\circ \)). It might also be possible to make use of Singh and Thornton's collection of pairwise residue orientations [26] so as to try preferentially the most probable conformations during the search.

Although many side chains are placed by virtue of sequence identity with a known structure, the rotamer library is used whenever this identity is lost. In these cases, it might be worthwhile to explore Sutcliffe's rule-based approach [28] for conservative replacement of similar residues in topologically similar environments in order to further reduce the dependence on the rotamer library.

As mentioned in the Evaluation section, adding a bias to the cost function to increase the penalty for sidechain-backbone steric clashes might be a simple way to improve the accuracy of the system. A better model of the surface residues might be obtained by including a surface accessibility factor in the cost function in order to reduce the effect of the tight packing bias in cases where polar side chains can be stabilised by the solvent.

Given that the overall approach of the system has been shown to work very well, it may be worthwhile trying to improve its performance. The cost function, for example, was originally devised from naïve reasoning about geometric packing and the observation that a 'simple' way to represent chemical interactions would be desirable. Perhaps the cost function is unnecessarily complicated? Perhaps Nature has found a complementarity between hydrogen-bonding patterns and tight packing? Would solving for one implicitly give a good solution for the other? I suspect not, but it might be interesting to experiment.

Nonetheless, there are several adjustable parameters in the system, which were assigned somewhat intuitively. It might be worthwhile to perform some systematic sensitivity studies to find better values for such parameters. Also, it might be worthwhile using more realistic energy equations for the initial database search for residue neighbours. Since this search needs to be done only once, in a sense it doesn't much matter what the computational expense would be. Certainly, it would be interesting to compare the performance of the cost function against real energy equations.

This project has focussed on modelling antibody structures, which are known to be highly homologous. The notion of database search for structural patterns relied very heavily on sequence similarity, exploiting the fact that high sequence similarity suggests good structural similarity. It would be interesting to see whether this approach would still work on other protein families. It might also be worthwhile investigating if there are better ways to represent structural clusters of interacting residues and to search for them efficiently.
5.3 Achievements

I believe that in this project I have quickly learnt how to use Prolog and P/FDM in an effective way and that I have applied this to a significant and substantial problem. I have built a working expert system which largely mimics a human expert's approach and which honours many of the important issues that are to be found in the literature.

This system is qualitatively as good as the manual approach that was used formerly by the expert (see Appendix C), who now plans to use this system in future modelling exercises. It removes much of the routine tedium that was previously involved and it replaces some of the arbitrary decisions that had to be made by a balanced judgement of appropriate alternatives. Indeed, the cluster placement algorithm goes beyond what any human expert could be reasonably expected to perform. Although not a panacea, this system provides a good initial model which should allow the user to focus his attention on some of the other interesting and difficult aspects of antibody modelling in general.

Additionally, I have shown that residue clusters can indeed be found in an efficient way and that this eliminates much of the conformational search (approximately 80% of residues are placed in just 20% of the execution time). The fact that the cluster search algorithm often finds residues from other structures in preference to those from the backbone donor (i.e. the structure that offered the highest sequence identity with the model) lends support to the idea that spatial and chemical interactions are at least as important as sequence identity, if not more so.

I also believe that I have devised and implemented some novel ideas which are of significant value in helping to solve a difficult problem. The notion of exact bounding spheres, for example, is so powerful yet so simple that I was surprised not to have seen it mentioned in the literature.

In many ways, the thrust of this project has been to demonstrate that intelligent database search can replace much of the brute-force search and computation that is often advocated in the literature. To use knowledge effectively, it is necessary to choose an appropriate granularity of knowledge representation and to encapsulate the key concepts of the problem domain as rules which can be applied to that representation. If this can be done in a systematic and structured way, then often the search can be applied in a tractable manner.

I believe that this has indeed been demonstrated here. Given some of the enhancements discussed in the previous sections, I would even suggest that
the system described here could even rival a ‘state of the art’ system such as *AbM*.

As a result of this project, I can now see that intelligent knowledge-based heuristics may have much to offer in helping to solve complex scientific problems, particularly in an information-rich yet computationally difficult domain such as Molecular Biology.

"*What’s past is Prologue.*"³


³Or is it *Prolog*?
References


Appendix A

User Manual

This section describes how to run the system. It is assumed that you have a Kabat sequence of an antibody structure that you wish to model. If this is not the case, you can still try running the system by simulating a mutation of an existing model sequence.

The code and data for the system are spread across a number of Unix directories in the Computing Science Department filesystem. These directories are located by reference to some Unix environment variables. These may be defined in the .cshrc file in your login directory:

\%
\% setenv SIDECHAIN_CODE /home/msc/ritchie/protein/progs
\% setenv SIDECHAIN_DATA /home/msc/ritchie/protein/data
\% setenv SIDECHAIN_DATABASES /home/swallow/ritchie

Here, the ‘%’ signs indicate c-shell command lines and should not be typed.

Note that the database files are located on swallow. The directories on this machine are not NFS-mounted to other systems. Thus the system will only work if you are logged in to swallow.

Before you can run the system, you will need to create a small file that contains Prolog facts that define the names and sequences of the VL and VH domains of the antibody to be modelled (manually typing such sequences would be tedious and error-prone). The simplest approach is to copy and edit an existing file (the new prompt assumes you have logged in to swallow):

```
swallow% cp $SIDECHAIN_CODE/target_p53.pl target Xxx.pl
```

68
where ‘XXX’ would be replaced by the code letters of the new model. The file that you have copied looks something like this:

```prolog
%-----------------------------------------------
% target.pl
%        
% Dave Ritchie, 04/07/95, Aberdeen AAI MSc Project.
%        
% Prolog predicates defining the model protein structure to be built.
% These must be specified as a Kabat Sequence for both the VL and VH antibody chains.
%        
% A = ALA = Alanine
% R = ARG = Arginine
% N = ASN = Asparagine
% D = ASP = Aspartic acid
% C = CYS = Cysteine
% Q = GLN = Gultamine
% E = GLU = Glutamic acid
% G = GLY = Glycine
% H = HIS = Histidine
% I = ILE = Isoleucine
% L = LEU = Leucine
% K = LYS = Lysine
% M = MET = Methionine
% F = PHE = Phenylalanine
% P = PRO = Proline
% S = SER = Serine
% T = THR = Threonine
% W = TRP = Tryptophan
% Y = TYR = Tyrosine
% V = VAL = Valine
% X = UNK = Unknown
% - = GAP = Inserted to improve the alignment
%-----------------------------------------------

target(p53_scFv).

% 123456789 123456789 123456789 123456789 12

target(p53_scFv_VL,light, '—DIELTQSPLTLSVTIQRASCKSQQSLQD—DGKTYLNW

target(p53_scFv_VH,heavy, '—QVKLQESGAILVRSVGASVKLSCATA3GFNIKDYYMH—WVKQ
```

69
Some of the lines may appear to be wrapped or truncated, so it’s better to use a 132-column window for editing. When you have replaced the domain names (p53...) and sequence letters you can now run the program by typing:

```
swallow% $SIDECHAIN_CODE/run model XXX
```

This generates instructions to read `target_XXX.pl` and to start the program. The program can take around 8 hours to complete so you may wish to start a background job instead:

```
swallow% $SIDECHAIN_CODE/run_batch model XXX
```

When the program has finished, it will send a mail message to that effect and it will have written the following files:

- `ZZZ.refined.sum`: summarises the source and status of model residues
- `ZZZ.refined.pdb`: model atom coordinates in PDB format
- `ZZZ.log`: log file from batch run
- `ZZZ.wrn`: lists unresolved bad inter-atomic contacts
- `ZZZ.contacts.def`: bad contact display selection commands for QUANTA
- `ZZZ.clusters.caf`: cluster colour selection commands for QUANTA
- `ZZZ.events`: lists all atomic pairwise events in the model
- `ZZZ_model.sum`: summary file prior to refinement
- `ZZZ_model.pdb`: PDB file prior to refinement

where `ZZZ` is replaced by the name that you gave for the protein when editing the `target/1` predicate.

The two ‘refined’ files (.sum and .pdb) represent the final state of the model. The .pdb file can be imported into a molecular graphics package such as QUANTA.

The ‘model’ files are produced after the initial side chain placement steps but before rotameric refinement, so that the output of the cluster algorithm can be examined in isolation.

Examples of the contents of these files (except the .pdb files) are given with the program listings.
Appendix B

Maintenance Manual

The Side Chain Placement System was developed on a Sun 4 Unix workstation.

The availability of Quintus Prolog and P/FDM are prerequisites. There are some known references to Quintus library predicates, and no doubt a few unknown ones. Fortran and C compilers would also be required to port the system to another platform.

The source code and binary files are located in the directory:

/home/msc/ritchie/protein/progs

on the Computing Science Department Unix file server. There is a Makefile for development and debugging. There are two Unix command scripts which are used to run the program either interactively (run) or in the background (batch_run). These command files ensure that the required Prolog source files are loaded and that the appropriate database directories are accessed. Some of the source files originated from Dr. Kemp. These are marked ‘GJLK’.

Makefile
batch_run
run
cacn_system.pl
chemical.pl
clusters.pl
foreign_fit.pl
geom.pl
model.pl
model_mask.pl
model_methods.pl

To make sphere.so and transform.so.
Run the system in batch.
Run the system in the foreground.
Define the Cα-C-N coordinate transformations.
Predicates for chemical interactions.
Predicates to find and operate on residue clusters.
Updated version of fitting code (GJLK).
Updated geometric transformation code (GJLK).
Main modelling predicates.
Predicates to operate on the model code mask.
Methods functions for the model database.
<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>model_neighbours.pl</td>
<td>Locates neighbouring model residues.</td>
</tr>
<tr>
<td>neighbour_methods.pl</td>
<td>Method functions for the neighbours database.</td>
</tr>
<tr>
<td>neighbours.pl</td>
<td>Locates neighbouring antibody residues.</td>
</tr>
<tr>
<td>output.pl</td>
<td>Miscellaneous output routines.</td>
</tr>
<tr>
<td>packing_methods.pl</td>
<td>Predicates to calculate the packing factor.</td>
</tr>
<tr>
<td>parameters.pl</td>
<td>Adjustable system parameters.</td>
</tr>
<tr>
<td>refine.pl</td>
<td>Predicates to refine the model.</td>
</tr>
<tr>
<td>rotlib.pl</td>
<td>Predicates to create the rotamer library.</td>
</tr>
<tr>
<td>rotlib_methods.pl</td>
<td>Method functions for the rotamer library.</td>
</tr>
<tr>
<td>rotmod_methods.pl</td>
<td>Method functions for the original Desmet rotamer library.</td>
</tr>
<tr>
<td>runit.pl</td>
<td>Predicates to access the antibody and sequence databases (GJLK).</td>
</tr>
<tr>
<td>runmodel.pl</td>
<td>Similarly, to build a model structure.</td>
</tr>
<tr>
<td>sphere.pl</td>
<td>Calculate/call-out spherical quantities.</td>
</tr>
<tr>
<td>target_ofab.pl</td>
<td>Model definitions to build ‘OFAB’.</td>
</tr>
<tr>
<td>target_p53.pl</td>
<td>Model definitions to build ‘P53’.</td>
</tr>
<tr>
<td>transform.pl</td>
<td>Updated call-outs to C transformation code (GJLK).</td>
</tr>
<tr>
<td>utilities.pl</td>
<td>Generic Prolog utilities (once, not, etc.).</td>
</tr>
<tr>
<td>clusters.sch</td>
<td>Schema definition for the clusters database.</td>
</tr>
<tr>
<td>model.sch</td>
<td>Schema definition for the model database.</td>
</tr>
<tr>
<td>model_methods.sch</td>
<td>Method signatures for the model database.</td>
</tr>
<tr>
<td>neighbour_methods.sch</td>
<td>Method signatures for the neighbours database.</td>
</tr>
<tr>
<td>neighbours.sch</td>
<td>Schema definition for the neighbours database.</td>
</tr>
<tr>
<td>packing.sch</td>
<td>Schema file for the temporary packing_methods module.</td>
</tr>
<tr>
<td>rotlib.sch</td>
<td>Schema definition for the rotamer database.</td>
</tr>
<tr>
<td>rotlib_methods.sch</td>
<td>Method signatures for the rotamer database.</td>
</tr>
<tr>
<td>rotmod.sch</td>
<td>Schema definition for the original Desmet rotamer library.</td>
</tr>
<tr>
<td>rotmod_methods.sch</td>
<td>Method signatures for the original Desmet rotamer library.</td>
</tr>
<tr>
<td>query.d</td>
<td>Daplex test queries and their output.</td>
</tr>
<tr>
<td>sphere.f</td>
<td>Fortran bounding sphere and overlap solver.</td>
</tr>
<tr>
<td>c_fit_v4.c</td>
<td>Updated C call-outs; extended for spheres (GJLK).</td>
</tr>
<tr>
<td>c_word_alloc.c</td>
<td>Memory allocation for Fortran.</td>
</tr>
<tr>
<td>c_sphere.c</td>
<td>Fortran to C interface for sphere.f.</td>
</tr>
<tr>
<td>transform.c</td>
<td>C coordinate transformation code (GJLK).</td>
</tr>
<tr>
<td>transform_pl.c</td>
<td>Updated Prolog interface to C coordinate transformations (GJLK).</td>
</tr>
<tr>
<td>transform.h</td>
<td>Header file for coordinate transformation code (GJLK).</td>
</tr>
</tbody>
</table>
sphered.so Binary version (‘shared object’) of sphere.f.
transform.so Binary version (‘shared object’) of transform.c.

Copies of the Desmet rotamer library and QUANTA parameters are in:

/home/msc/ritchie/protein/data

protein.rud van der Waals atomic radii from QUANTA.
protein.typ Residue atom names table from QUANTA.
rotlib.loa QUANTA parameters in P/FDM load file format.
rotmod.loa Desmet rotamer library in P/FDM load file for-
mat.

The database files (and any large output files) are located on swallow, where there is more disc space:

/home/swallow/ritchie/*

neighbours* The neighbours database files.
rotlib* The rotamer database files.
rotmod* The original Desmet rotamer database files.
*.events Residue interaction events for all antibody pro-
teins.

Additionally, there is a subdirectory for each antibody model (not shown). The above directories may also contain a few temporary files which have not been listed here as they are not necessary parts of the system.

This report was typeset using LaTeX. Showcase was used to produce the figures and the colour plates were produced from photographs of QUANTA displays.
Appendix C

Human Expert Protocol

The following text is a protocol (somewhat abridged) of a recent modelling exercise by a human expert\textsuperscript{1}.

\textit{I have now built a model of the p53 antibody Fv. The model’s VL domain was based on the VL domain of the structure with PDB code 2IGF and the model’s VH domain was based on the VH domain of the structure with PDB code 2HFL, since these were the domains with greatest sequence identity. The main chains from these domains were copied from the known structures. The two chains in the model are the same length as those on which the model was based (i.e. no “insertions” or “deletions” needed to be modelled). Side chains were then added using the following three methods.}

\begin{itemize}
\item[(a)] Where the model was to have a side chain that was identical to the one at the corresponding position in the known structure on which the model was based, that side chain conformation was used in the model. Most of the side chains were modelled in this way (i.e. 2IGF VL side chains and 2HFL VH side chain were copied into the model wherever possible).
\item[(b)] Wherever a side chain of the desired type was present at the “corresponding position” in one of the other known structures, then the side chain conformation was taken from that other known structure. Often, more than one structure would have the desired residue at the corresponding position. In these cases, one of these conformations with “typical” side chain torsion angles was selected (“arbitrary” selection).
\end{itemize}

\textsuperscript{1}Dr. G. Kemp, Aberdeen Computing Science Department, modelled the P53 Fv domain.
(c) Where a side chain of the desired type is not present at the corresponding position in any of the known structures, the most common conformation for a residue of the desired type was chosen from a rotamer library (I have least confidence in side chains placed in this way).

I list below the method used for modelling each side chain (method (a) was used, unless otherwise stated). For residues modelled using method (b), I give also the PDB code of the structure from which the side chain conformation was taken.

<table>
<thead>
<tr>
<th>Residue</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>L:Ile2</td>
<td>(b) 3HFM</td>
</tr>
<tr>
<td>L:Glu3</td>
<td>(b) 8FAB</td>
</tr>
<tr>
<td>L:Leu4</td>
<td>(b) 3HFM</td>
</tr>
<tr>
<td>L:Ser7</td>
<td>(b) 3HFM</td>
</tr>
<tr>
<td>L:Thr10</td>
<td>(b) 3HFM</td>
</tr>
<tr>
<td>L:Ser12</td>
<td>(b) 3HFM</td>
</tr>
<tr>
<td>L:Thr14</td>
<td>(b) 3HFM</td>
</tr>
<tr>
<td>L:Ile15</td>
<td>(c)</td>
</tr>
<tr>
<td>L:Gln17</td>
<td>(b) 8FAB</td>
</tr>
</tbody>
</table>

With all side chains added, the VH and VL domains were brought into the same frame of reference. I first found the transformation that maps the 2HFL VH framework onto that of 2IGF by using McLachlan’s fitting algorithm to superpose all CA atoms of residues in the VH framework as defined by Chothia and Lesk (1987). I then applied this transformation to all of the atoms of the modelled VH domain. I then inspected the initial model using the QUANTA molecular graphics, making the following observations.

<table>
<thead>
<tr>
<th>Residue</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>L:Ile2</td>
<td>OK, no clashes, packs against CB of L:Ser25</td>
</tr>
<tr>
<td>L:Glu3</td>
<td>OK, side chain h-bond to L:Thr5</td>
</tr>
<tr>
<td>L:Leu4</td>
<td>OK, packed in core</td>
</tr>
<tr>
<td>L:Ser7</td>
<td>OK, few constraints, could rotate OG away from own C</td>
</tr>
<tr>
<td>L:Thr10</td>
<td>OK, (could rotate OG1 away from own N if desired)</td>
</tr>
<tr>
<td>L:Ser12</td>
<td>OK, side chain h-bond to L:Glu105?</td>
</tr>
<tr>
<td>L:Thr14</td>
<td>Close to L:Gln17, could rotate to suggest h-bond</td>
</tr>
<tr>
<td>L:Ile15</td>
<td>OK, default rotamer</td>
</tr>
<tr>
<td>L:Gln17</td>
<td>(See L:Thr14)</td>
</tr>
</tbody>
</table>

...
I then adjusted some side chain torsion angles interactively to remove clashes. Details of this are listed below. Note that none of the side chains placed using method (a) were adjusted in this way. Many of the clashes involved side chains modelled using method (c). Where there was a choice, side chains modelled using method (c) were adjusted by hand in preference to those modelled using method (b).

Rotated L:Thr14 to remove clash and suggest h-bond.
Rotated L:Asp27D to remove clash and suggest h-bond.
Rotated L:Lys30 away from L:Tyr32.
Rotated L:Arg39 away from L:Glu42.
Rotated L:Arg46 to remove clashes.
Rotated L:Leu50 away from L:Tyr49.
Rotated L:Asp55 to remove clashes.
Rotated L:Leu54 away from L:Leu48 to improve packing.

... 
L:Trp89 difficult!

There are other mutations nearby. First, I moved H:Tyr95 and L:Asn34 to free up some space, then rotated L:Trp89 to remove clashes. (Rotating L:Arg46 might also help, but this was not done.)

The coordinates for the model, in PDB format, are in the following message.
Appendix D

Bounding Sphere Algorithm

The bounding sphere algorithm is most easily thought of as a problem in which the solution is to find the smallest sphere that bounds a given set of points in space.

Figure D.1: The 4 steps of the sphere solving algorithm.
The algorithm has four main steps as follows:

1. Guess a centroid of all points. This is shown in D.1(a). Since 4 points in space are required to solve for the surface of a sphere, find the 4 furthest points from this centroid. Each of these points defines a radius from the centroid. The length of the radius of the solution sphere must fall between the lengths of the 1st and 4th largest radii. The difference between the 1st and 4th radii, \( D_r \), is the maximum amount by which the centroid can move in any direction for this to continue to hold. This is shown in D.1(b).

2. Move the centroid towards the furthest point (1), by up to \( D_r \) until some other point is touched by the new (reduced) radius of the bounding sphere. This gives two points on the surface of the sphere and a smaller \( D_r \), as shown in D.1(c).

3. Find the mid-point of the two surface points (1) and (2) and move the centroid towards that point by up to \( D_r \) until some third point is touched by the new (reduced) radius of the bounding sphere. If the centroid becomes co-linear with (1) and (2), then the problem is solved. Otherwise the difference between the radius of this sphere and the next furthest point defines a new \( D_r \).

4. The three points on the surface of the bounding sphere define a plane which makes a circular cut through the sphere, as in D.1(d). The origin of the bounding sphere forms a perpendicular to the circular plane at a point which intersects the centre of that circle. Move the centroid towards that point by up to \( D_r \) until some fourth point is touched by the new reduced sphere. The problem is solved.

This can also be applied to equally sized spheres instead of points, by adding the radius of the object spheres to each of the radial distances from the centroid.

If the object spheres have different sizes (for example in the case of atoms), then the appropriate object radius should be added to each radial distance from the centroid. This causes a slight error as the iteration proceeds, which can be corrected by a second iteration of the final three steps. In the case of approximately equal sized spheres, such as atoms, the error will be very small and probably can be ignored, although the current implementation finds the numerically exact solution.
Appendix E

Spherical Overlap Volume

The overlap volume $S_{12}$ of two spheres of radius $r_1$ and $r_2$ can be found by considering the volume as being composed of two spherical caps $C_1$ and $C_2$. This is shown in Figure E.1.

![Diagram of spherical overlap volume]

Figure E.1: Overlap volume as two spherical caps $C_1$ and $C_2$.

The radius $u$ of the circular cut can be found by Pythagoras:
\[
h_1 = \frac{\frac{r_1^2}{2} - r_2 + r_{12}^2}{2r_{12}} \tag{E.1}
\]

\[
h_2 = \frac{\frac{r_2^2}{2} - r_1 + r_{12}^2}{2r_{12}} \tag{E.2}
\]

\[
u = \sqrt{\frac{r_1^2}{2} - \left(\frac{r_1^2 - r_2^2 + r_{12}^2}{2r_{12}}\right)^2} \tag{E.3}
\]

The parametric equation for the surface of a sphere of radius \( a \) is given by:

\[
z = (a^2 - x^2 - y^2)^{1/2} \tag{E.4}
\]

If a sphere is translated down the \( z \) axis by \( h \), then the volume of the cap under the positive \( z \) axis is found by integrating under the surface:

\[
C = \int \int (z - h) \, dx \, dy \tag{E.5}
\]

Converting to polar coordinates \((x = r \cos \theta, y = r \sin \theta)\) gives:

\[
C = \int_0^{2\pi} \, d\theta \int_0^u z \, r \, dr \tag{E.6}
\]

\[
C = 2\pi \int_0^u ((a^2 - r^2)^{1/2} - h) \, r \, dr \tag{E.7}
\]

\[
C = 2\pi \left[ \frac{1}{3}(a^2 - r^2)^{3/2} \Big|_0^u - \frac{h}{2}r^2 \right] \tag{E.8}
\]

\[
C = 2\pi \left( \frac{a^3}{3} - \frac{h^3}{3} - \frac{hu^2}{2} \right) \tag{E.9}
\]

And so by appropriate substitutions for \( a \) and \( h \):

\[
S_{12} = C_1 + C_2 \tag{E.10}
\]

A slight modification (not shown) is required when \( r_{12} < r_2 \), i.e. when the smaller sphere has passed more than half way into the larger sphere.