Structural Identification of Immunoglobulin Variable Domains

Li Huihua

Supervisor: Graham J.L. Kemp

Chalmers University of Technology

Acknowledgements

This report is submitted as a Master's thesis in the International Master's Program of Bioinformatics at Chalmers University of Technology. The project described in this report was carried out at Chalmers University of Technology under the supervision of Graham J.L. Kemp, to whom I would like to express my deep gratitude for his advice, guidance and encouragement. I benefited a lot from close collaboration with him. I would also thank other teachers and students in our program for their help and encouragement.

Abstract

A database of antibody sequences and structures was built in an earlier project [Kemp et al., 1994]. That database contained indexes that made it easy to find residues at structurally equivalent positions in different domains, thus making it convenient to explore structural hypotheses. However, constructing these indexes by hand was a slow and inconvenient process, so we now seek to automate this task as we extend the database to include additional immunoglobulin domains.

Earlier structural studies of antibody VH and VL variable domains [Chothia and Lesk, 1987] and T-cell receptor variable domains [Chothia et al., 1988] identified template $\beta - sheet$ framework patterns based on conserved main chain hydrogen bond patterns. In this project we have adapted and improved some of these templates so that these can also recognize some of the new structures that have been determined experimentally since the earlier templates were derived. We have then used these templates to identify VH, VL, TCR V-α and TCR V-β domains in Protein Data Bank files automatically by computing main chain hydrogen bonds within the structures and then matching these computed patterns against the template patterns. The results show that the software developed here can help the process of finding structurally equivalent positions in different domains automatically.

List of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Immunoglobulin variable domains are present in many proteins. These domains have a compact globular structure containing two beta-sheets. Examples of immunoglobulin variable domains include antibody variable domains, and TCR α and β variable domains. These variable domains have hypervariable loops, which connect strands of this $\beta - sheet$ framework and are important for binding antigen. Immunoglobulins are of particular interest due to their high degree of specificity, which provides a wide range of therapeutic and other applications. Experimental developments over the past few years have led to new techniques for constructing artificial molecules based on natural immunoglobulin variable domains. Although a molecule's three-dimensional structure determines its biological function, many sequences do not have their structures determined experimentally. Known variable domain structures can be used as the basis for modelling others following a comparative modelling strategy.

In an earlier project at the University of Aberdeen, a database of antibody sequences and structures was constructed to support systematic structural studies and comparative modelling [Kemp et al., 1994]. However, it is not easy or convenient to maintain this database by adding more immunoglobulin variable domains since different numbering schemes are adopted in different structure files in the Protein Data Bank (PDB) [Bernstein et al., 1977], which is the main depository for protein structure data. We cannot easily make sure that residues with the same position index from corresponding PDB file are the residues at structurally equivalent positions, which contribute similar structure in different PDB files. Different position indexes at structurally equivalent positions made the maintenance of the database slow and awkward since these positions had to be identified and recorded by hand.

1.1 Objectives

This project aimed to automate the task of finding residues at structurally equivalent positions in different immunoglobulin variable domains. The main C program developed here can be used under Solaris 7 or any other Unix/Linux system.

1.2 Thesis overview

The background of this project including immunoglobulin variable domains, different numbering schemes used in PDB and the conserved main chain hydrogen bond patterns of different domains is introduced in Section 1.3. Chapter 2 describes the main methods used in this project to compare the hydrogen bond pattern of the examined chain to those of the templates. Chapter 3 gives the improved templates and summarises the performance of the software developed in this project. Finally, short conclusions are drawn in Chapter 4 to evaluate this software. The maintenance manual and user manual of the programs developed in this project are given in appendices.

1.3 Background

1.3.1 Immunoglobulin variable domain

In this project we are interested in structures that are classified in SCOP [Murzin et al., 1995] as belonging to the family "V set domains (antibody variable domain-like)" such as antibody variable domains, and TCR α and β variable domains. These structures have a $\beta - sandwich$ fold, with two antiparallel sheets. Within each immunoglobulin variable domain, the protein chain threads back and forth from one end of the domain to the other. Adjacent strands in the same sheet are held together by a regular hydrogen bond pattern to provide a stable framework. The heavy chain variable domain (VH domain) is one of the immunoglobulin variable domains, and the connectivity of its strands is shown in Figure 1.1.

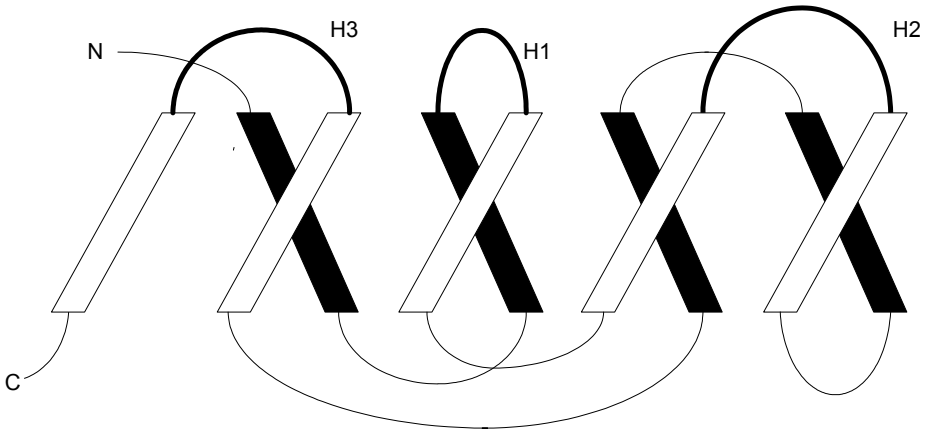


Figure 1.1 Connectivity of strands in a VH domain, adapted from [Kemp et al., 1994].

By "cutting" the loops towards the bottom of Figure 1.1 and "folding" the four black strands upwards, we can put all the nine strands on the same plane with the four black stands above the five white ones to show us the simplified structure of the VH domain (Figure 1.2). The connectivity of strands in light chain variable domain (VL domain), and T cell receptor (TCR) α and β domains is similar.

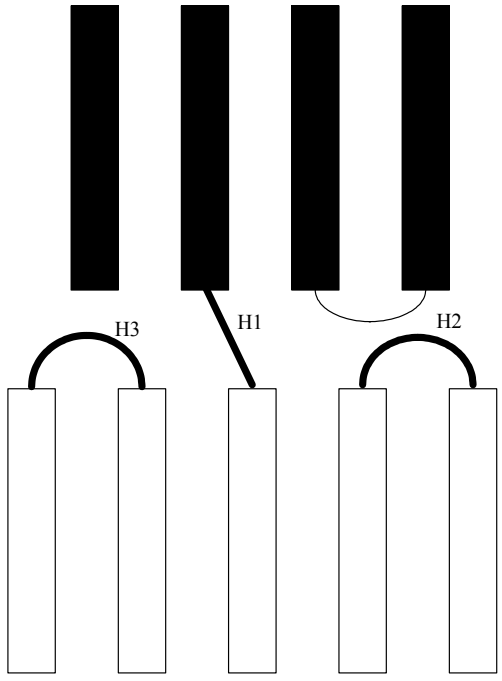


Figure 1.2: Simplified structure of a VH domain.

1.3.2 Numbering Scheme

So far, numbering schemes adopted in the Protein Data Bank include the Kabat Numbering Scheme, the Chothia Numbering Scheme and the Consecutive Numbering Scheme. This variety of numbering schemes makes it difficult to find structurally equivalent positions in different domains.

The Kabat Numbering Scheme[1]

The Kabat Numbering Scheme, developed from multiple sequence alignments, is the widely adopted standard for numbering the residues in an antibody in a consistent manner. However, the insertion positions in the complementarity determining regions (CDR) may not match the structural insertion positions.

The Chothia Numbering Scheme[1]

The Chothia Numbering Scheme is identical to the Kabat Numbering Scheme, but places the insertions in CDR regions at the structurally correct positions. However, some confusion may arise because the Kabat Numbering Scheme is so widely used.

Consecutive Numbering Scheme

Sometimes the Consecutive Numbering Scheme is adopted in PDB file to assign consecutive numbers to the residues starting with residue number one at the N-terminal of the chain and numbering residues according to their actual positions within the protein chain.

1.3.3 Variable domain templates

Earlier structural studies of antibody VH and VL variable domains [Chothia and Lesk, 1987] and T-cell receptor α and β domains [Chothia et al., 1988] have identified template $\beta - sheet$ framework patterns based on conserved main chain hydrogen bond patterns.

Figures 1.3 and 1.4 show a schematic representation of four different kinds of $\beta - sheet$ framework (VL, VH, TCR α and β variable domains) based on those earlier

[1] Adapted from descriptions of antibody residue numbering schemes by Andrew Martin: http://www.rubic.rdg.ac.uk/abs.

studies. Figure 1.5 shows the common schematic representation for all these kinds of domains. In each figure, one number is shown at each end of framework region, which is the position index using the Chothia Numbering Scheme. In these figures circles represent amino acid residue positions. Those circles with C or W inside indicate that the residue at these positions is usually Cysteine or Tryptophan. Thick black lines join residues that are adjacent in the sequence, and thin blue lines represent main chain hydrogen bonds. In Figure 1.3 the red lines represent an antibody's complementarity determining regions. The consecutive residues in one line connected by thick black lines correspond to one strand shown in Figure 1.2, except for the residues from position 9 to 13 in the VL domain and from position 8 to 12 in the VH domain. In Figure 1.5 four numbers are shown at the ends of each framework region. The upper pair, M_1/M_2, are the VL and VH sequence position codes proposed in [Chothia and Lesk, 1987]; the lower pair, N_1/N_2, are the TCR α and β domain sequence position codes proposed in [Chothia et al., 1988].

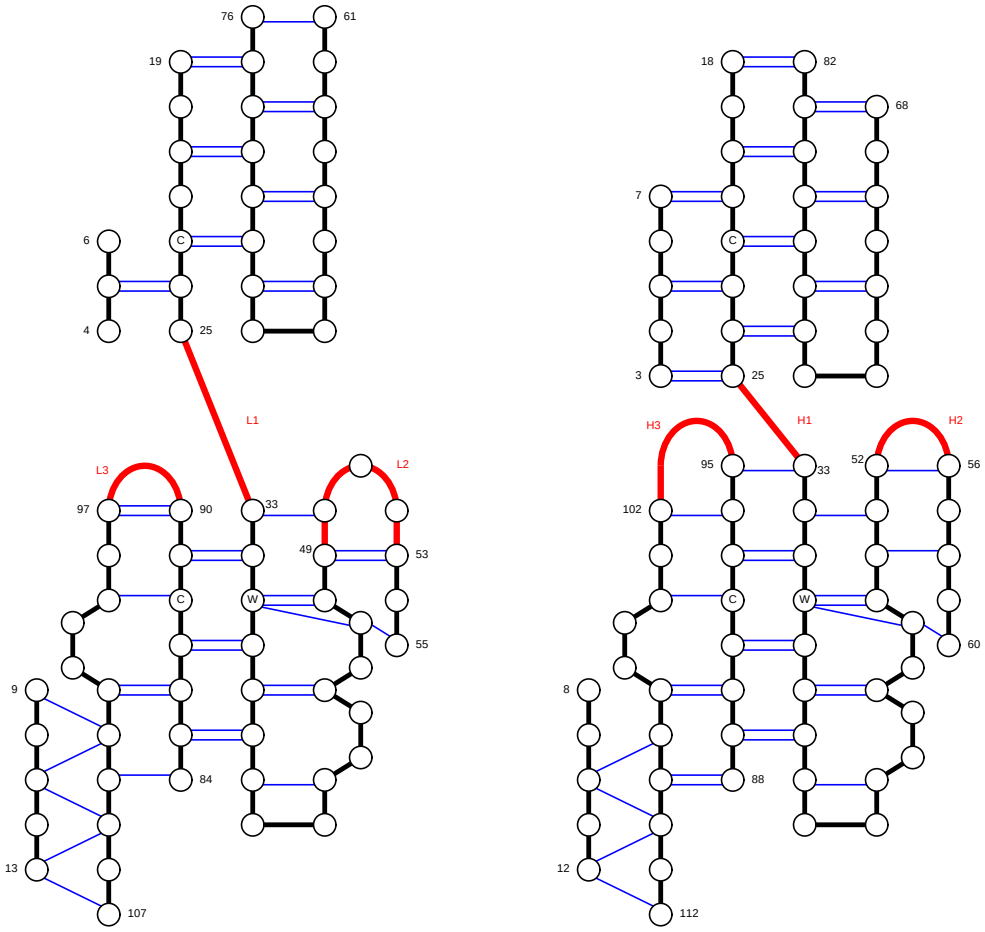


Figure 1.3: Plane of the $\beta - sheet$ framework conserved in VL (left) and VH (right) domains, adapted from [Chothia and Lesk, 1987].

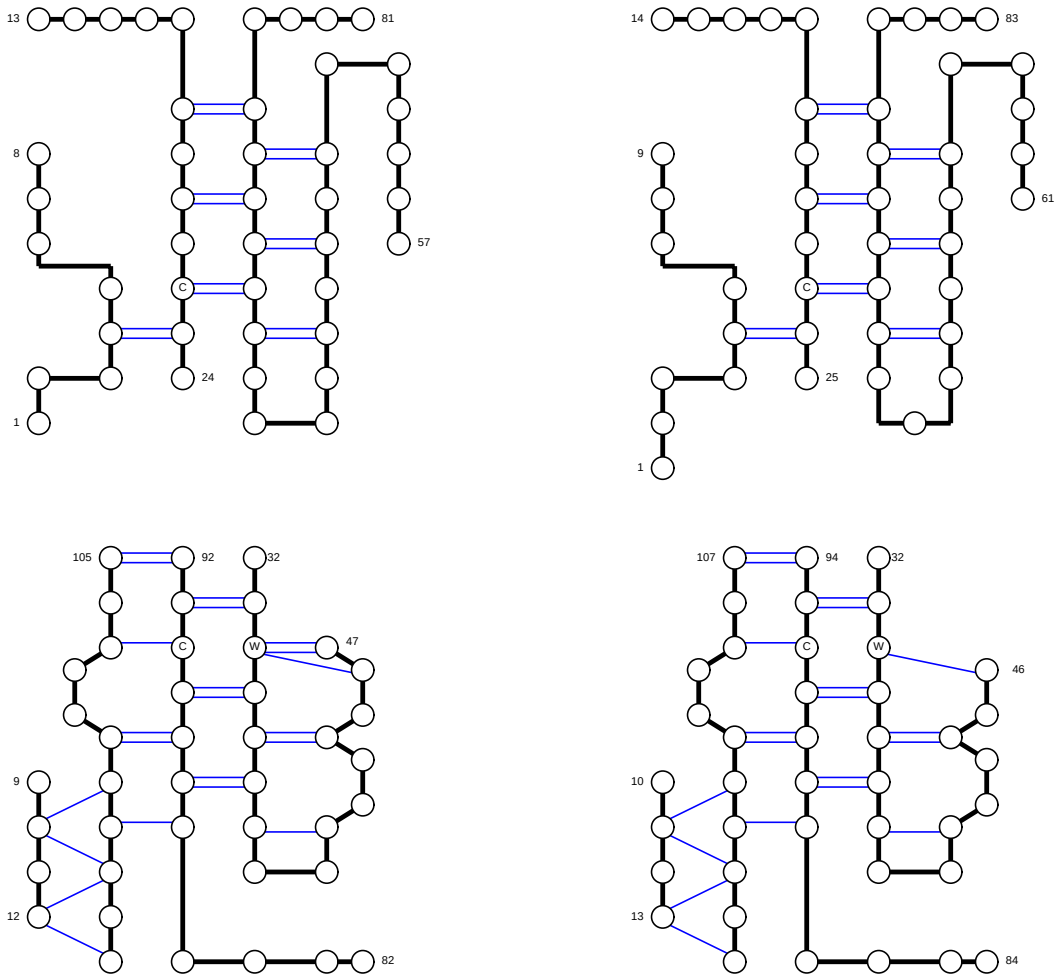


Figure 1.4: Plane of the $\beta - sheet$ framework conserved in T-cell receptor α (left) and β (right) variable domains, adapted from [Chothia et al., 1988].

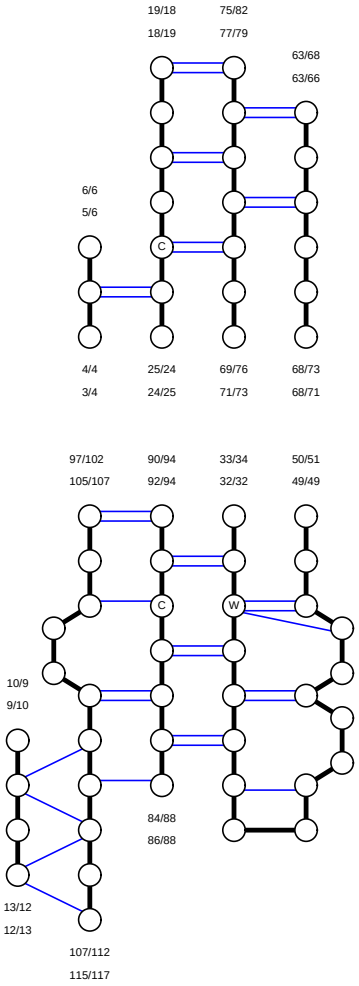


Figure 1.5: Plane of $\beta - sheet$ framework conserved in VL, VH, T-cell receptor α and β variable domains, adapted from [Chothia et al., 1988].

If we have a new structure file and we want to identify to which family it belongs, one possible approach would be to look for main chain hydrogen bond pattern in the new structure and then compare this pattern with the template hydrogen bond patterns shown in Figures 1.3, 1.4 and 1.5.

In Chapter 2, we shall introduce how to find the hydrogen bond patterns of the new structures and how to identify and classify these structures.

Chapter 2

Methods

In this chapter, the methods to identify and classify immunoglobulin variable domains are described. Figure 2.1 shows the whole procedure for identifying and classifying these domains. The main depository of immunoglobulin structure data is the Protein Data Bank [Bernstein et al., 1977]. In order to find all main chain hydrogen bonds, we need to know the coordinates of all main chain N, H, C and O atoms. Atom records in these PDB files contain the x, y and z coordinates of all of the "heavy" atoms (N, C, O, S, etc), but do not contain coordinates for hydrogen atoms. We need to get information of N, C, and O positions in main chain C=O and N-H group and C_α positions from PDB files and calculate positions of hydrogen bonded to main chain N. Then we find all the possible hydrogen bonds between each pair of main chain C=O group and N-H group of the examined chain [Section 2.2]. Then, we identify framework blocks [Section 2.3] corresponding to strands of each of the templates shown in Figures 1.3, 1.4 and 1.5. Finally, we compare the hydrogen bond pattern of the examined chain to that of the template, if they are similar, we assert that the examined chain matches the template. Otherwise, it doesn't match.

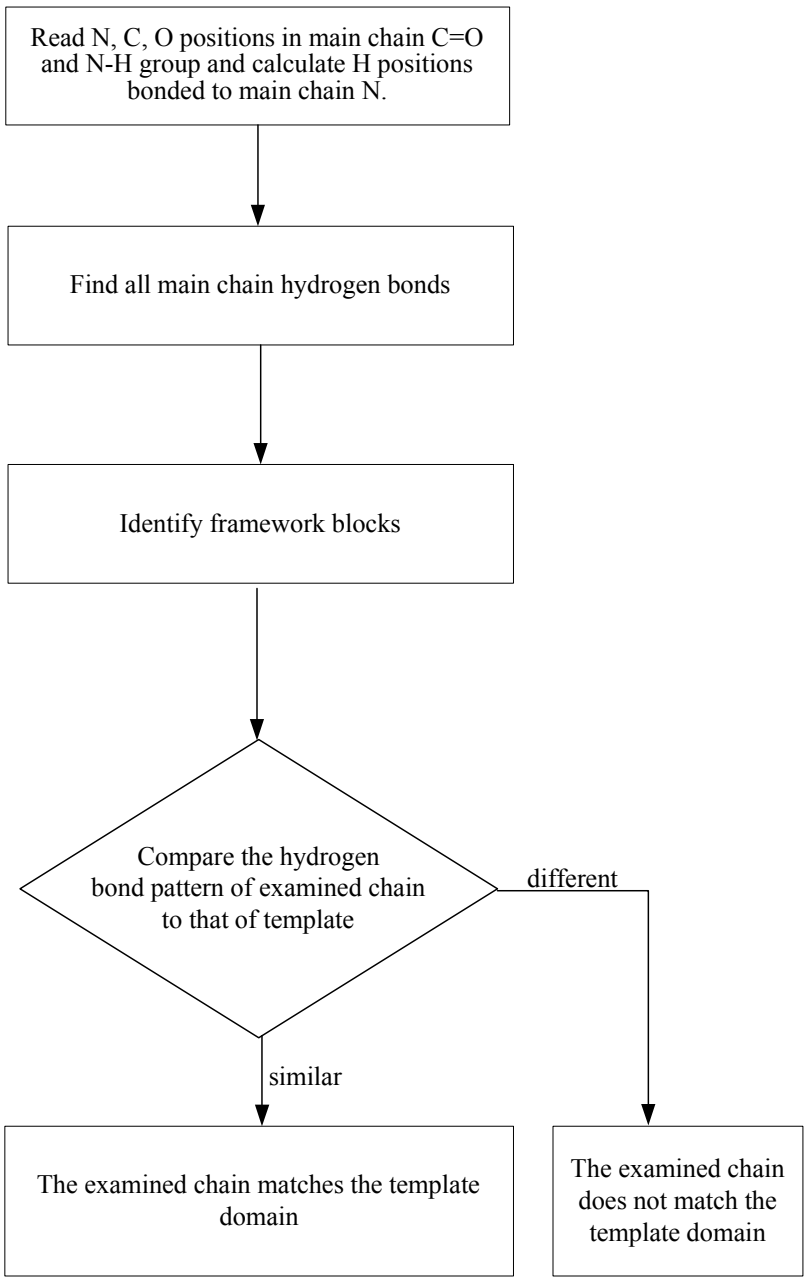


Figure 2.1: Flow chart for identifying and classifying immunoglobulin variable domains.

2.1 Get positions of main chain C=O and N-H groups

Since we need the positions of main chain atoms C, N, atom O bonded to main chain atom C and atom H bonded to main chain atom N to calculate the energy between main chain C=O group of residue i and N-H group of residue j, we read positions of main

chain atoms C, N, C_α and atom O bonded to main chain atom C at first by writing a program "getcoordinate.pl".

Since we cannot get the positions of atom H bonded to main chain atom N from PDB files, we use the positions of main chain atoms C, N and C_α to compute the positions of the hydrogen atom bonded to each main chain nitrogen atom. A polypeptide chain with three amino acid residues is shown in Figure 2.2. In this figure, the peptide bonds joining consecutive residues are represented by the thick black lines; R1, R2 and R3 are the side chains of different residues. The six atoms in the dotted rectangle are on the same plane (the peptide plane). The relative positions of main chain atoms C_α, N and H bonded to N of residue i, main chain atom C of residue i-1 are shown in Figure 2.3.

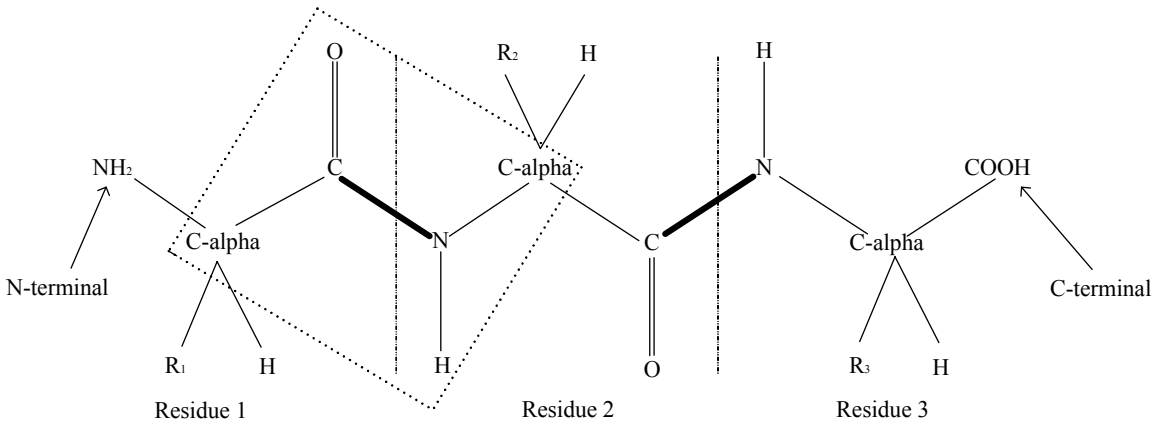


Figure 2.2: The polypeptide chain.

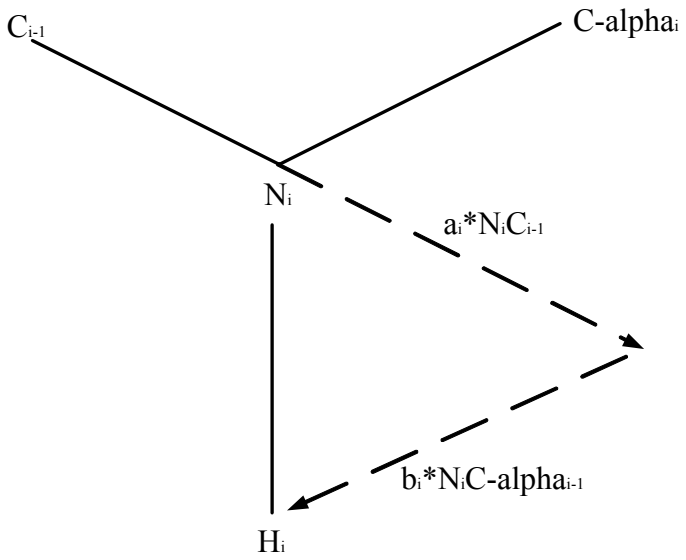


Figure 2.3: The relative positions of main chain atoms C_α, N and H bonded to N of residue i, main chain atom C of residue i-1.

As shown in Figure 2.3, for any two consecutive residues i-1 and i, the main chain atom C of residue i-1, the main chain atom C_α, N and H bonded to N of residue i are on the same plane. If we consider the distance from N to H of residue i, the distance from N to C_α of residue i, and the distance from N of residue i to the main chain atom C of residue i-1 as vectors, they have the following relationship:

$$\overrightarrow{N_i H_i} = a_i * \overrightarrow{N_i C_{i-1}} + b_i * \overrightarrow{N_i C_{\alpha i}} \tag{2.1}$$

where, a_i and b_i are coefficients.

For the triangle shown in Figure 2.3, the sides and angles have the following relationship:

$$\frac{|N_i H_i|}{\sin(180^0 - \angle C_{i-1} N_i C_{\alpha i})} = \frac{|a_i| * |N_i C_{i-1}|}{\sin(180^0 - \angle C_{\alpha i} N_i H_i)} = \frac{|b_i| * |N_i C_{\alpha i}|}{\sin(180^0 - \angle C_{i-1} N_i H_i)} \tag{2.2}$$

The coefficients can be found as follows:

$$a_i = -\frac{|N_i H_i|}{\sin(180^0 - \angle C_{i-1} N_i C_{\alpha i})} * \frac{\sin(180^0 - \angle C_{\alpha i} N_i H_i)}{|N_i C_{i-1}|} \tag{2.3}$$

$$b_i = -\frac{|N_i H_i|}{\sin(180^0 - \angle C_{i-1} N_i C_{\alpha i})} * \frac{\sin(180^0 - \angle C_{i-1} N_i H_i)}{|N_i C_{\alpha i}|} \tag{2.4}$$

Therefore, the positions of the hydrogen atom bonded to each main chain nitrogen atom should be:

$$Hx_i = Nx_i + a_i * (Cx_{i-1} - Nx_i) + b_i * (C_\alpha x_i - Nx_i) \tag{2.5}$$

$$Hy_i = Ny_i + a_i * (Cy_{i-1} - Ny_i) + b_i * (C_\alpha y_i - Ny_i) \tag{2.6}$$

$$Hz_i = Nz_i + a_i * (Cz_{i-1} - Nz_i) + b_i * (C_\alpha z_i - Nz_i) \tag{2.7}$$

where $\angle C_{\alpha i} N_i H_i = 118^0$, $\angle C_{i-1} N_i C_{\alpha i} = 122^0$, $\angle C_{i-1} N_i H_i = 120^0$,[2]

$|N_i H_i| = 1.02$ Å[3]

2.2 Hydrogen bond pattern of examined chain

Since the structural analysis to identify immunoglobulin variable domains is based on the conserved main chain hydrogen bonds, the second step is to find all main chain hydrogen bonds by calculating the energy between the main chain C=O group of residue i and the main chain N-H group of residue j using the method described in [Kabsch and Sander, 1983]:

$$E = q_1 q_2 \left(\frac{1}{r(O_i N_j)} + \frac{1}{r(C_i H_j)} - \frac{1}{r(O_i H_j)} - \frac{1}{r(C_i N_j)}\right) * f \tag{2.8}$$

with electric charge on the main chain C=O group $q_1 = 0.42e$ and electric charge on the main chain N-H group $q_2 = 0.20e$, e is the unit electron charge and r(AB) is the inter atomic distance from A to B. In chemical units, r is in Ångstroms, the dimensional factor $f = 332$, and E is in $kcal/mol$. A good hydrogen bond has about $-3 kcal/mol$

[2] http://www.pharmacology2000.com/physics/chemistry_physics/physics23.htm.
[3] http://www.cmbi.kun.nl/gv/service/counting/SET1/BNDLEH/.

binding energy. Here we choose a generous cutoff and assign a hydrogen bond between C=O group of residue i and N-H group of residue j if $E < -0.5\, kcal/mol$.

2.3 Identify framework blocks

As shown in Figure 1.1, each immunoglobulin variable domain consists of two $\beta - sheets$; each sheet is composed of several strands. In each strand between members of the same family, there are no insertions or deletions. However, insertions or deletions may occur in the loops between strands. In this case, we need to divide the whole sequence of the $\beta - sheet$ framework into several ***blocks***, corresponding to the strands in the immunoglobulin variable domains.

Figure 2.4 shows the blocks of residues in the conserved $\beta - sheet$ framework of immunoglobulin variable domains. In this figure, blocks are numbered consecutively from 1 to 8, with block 1 closest to the N-terminal while block 8 closest to the C-terminal. Here, each block represents one strand. Since there is no insertion or deletion between blocks 5 and 6 for some kinds of immunoglobulin variable domains, sometimes these two blocks are combined into one block. Therefore, for different kinds of immunoglobulin variable domains, the total numbers of blocks may be different, but they are partitioned in the same way.

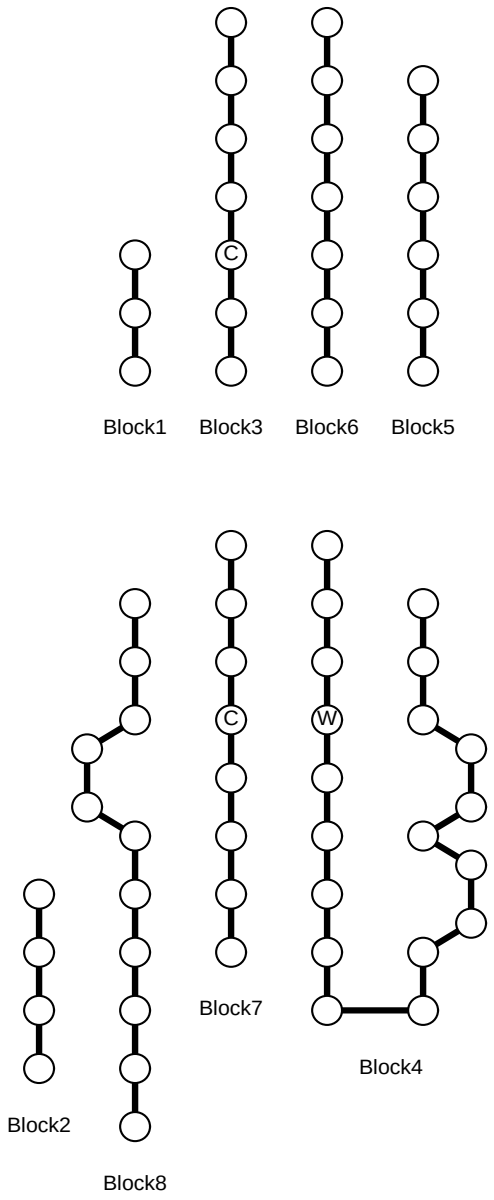


Figure 2.4: Blocks of residues in the conserved $\beta - sheet$ framework of variable domains.

The hydrogen bonds between the C=O groups of Block[i] and N-H groups of Block[j] is named as Pattern[i][j]. We call Pattern[i][j] intra-block hydrogen bond pattern if i is equal to j, otherwise, we call it inter-block hydrogen bond pattern. Tables 2.1, 2.2, 2.3, 2.4 list the summary of blocks and hydrogen bonds conserved in VL and VH domains [Chothia and Lesk, 1987] shown in Figure 1.3, and in TCR α and β domains [Chothia et al., 1988] shown in Figure 1.4. All the residue numbers in these tables follow the Chothia Numbering Scheme.

Table 2.1: Summary of blocks and hydrogen bonds conserved in VL domains.

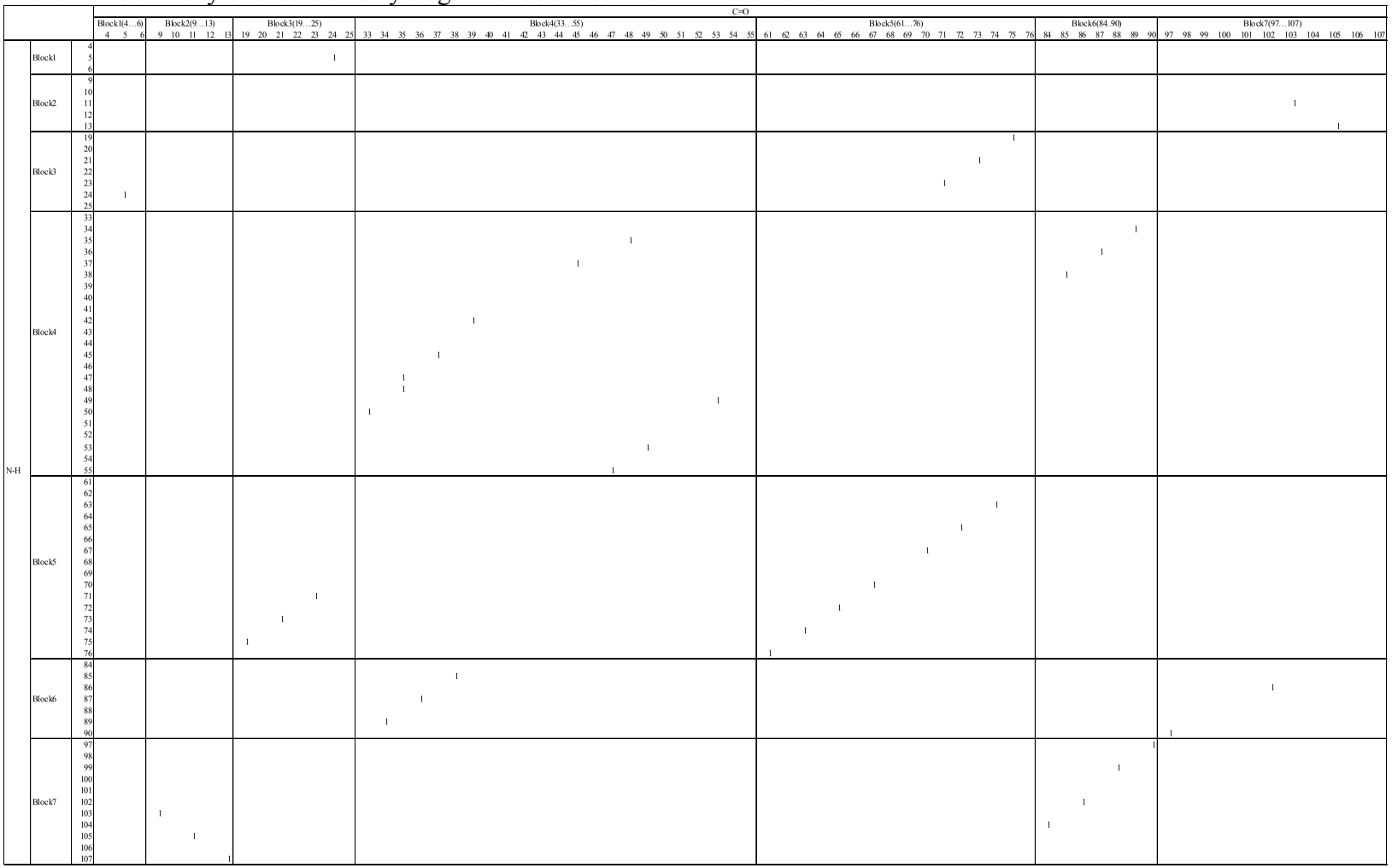


Table 2.2: Summary of blocks and hydrogen bonds conserved in VH domains.

			C=O																																																																										
			Block1(3?)					Block2(8? 2)					Block3(18? 5)								Block4(33? 2)																				Block5(56? 0)					Block6(68? 2)															Block7(88? 5)								Block8(102? 12)										
			3	4	5	6	7	8	9	10	11	12	18	19	20	21	22	23	24	25	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	56	57	58	59	60	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	88	89	90	91	92	93	94	95	102	103	104	105	106	107	108	109	110	111	112
		3																		1																																																												
		4																																																																												
	Block1	5																1																																																													
		6																																																																											
		7														1																																																														
		8																																																																											
		9																																																																											
	Block2	10																																																																								1				
		11																																																																											
		12																																																																										1		
		18																																																											1																			
		19																																																																											
		20																																																									1																					
	Block3	21					1																																																																									
		22																																																							1																							
		23			1																																																																										
		24																																																			1																									
		25	1																																																																												
		33																																																																											
		34																																																																												
		35																																																															1													
		36																																		1																																											
		37																																																													1															
		38																															1																																													
		39																																																													1																	
		40																																																																											
		41																																																																											
	Block4	42																																																																											
		43																									1																																																							
		44																																																																											
		45																																																																											
		46																							1																																																								
		47																																																																											
		48																					1																																																										
		49																					1																																																											
		50																																																																											
		51																			1																																																													
		52																																						1																																											
N-H		56																																																																												
		57																																																																												
	Block5	58																																			1																																													
		59																																																																											
		60																																	1																																																
		68																																																										1																				
		69																																																																											
		70																																																									1																						
		71																																																																										
		72																																																							1																								
		73																																																																										
		74																																																																											
	Block6	75																																																																											
		76																	1																																																													
		77																																													1																																
		78														1																																																																
		79																																											1																																		
		80												1																																																																			
		81																																										1																																				
		82										1																																																																						
		88																																																																									1			
		89																								1																																																									
		90																																																																							1					
	Block7	91																						1																																																												
		92																																																																											
		93																				1																																																														
		94																																																																		1										
		95																		1																																																															
		102																																																																												
		103																																																																												
		104																																																																1															
		105																																																																											
		106																																																																											
	Block8	107																																																													1																	
		108																																																																											
		109																																																												1																			
		110							1																																																																								
		111																																																																											
		112									1																																																																					

Table 2.3: Summary of blocks and hydrogen bonds conserved in TCR α domains.

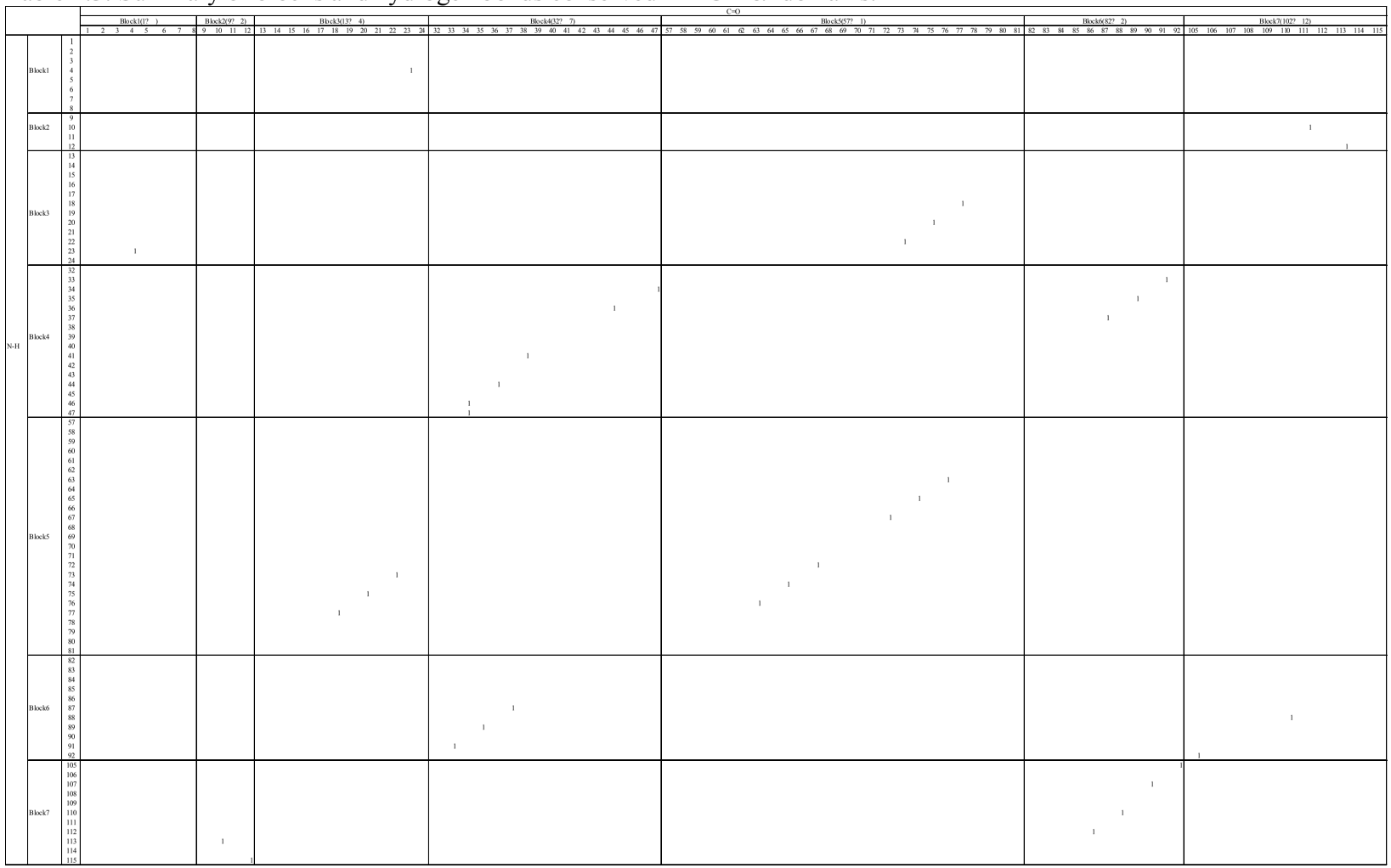
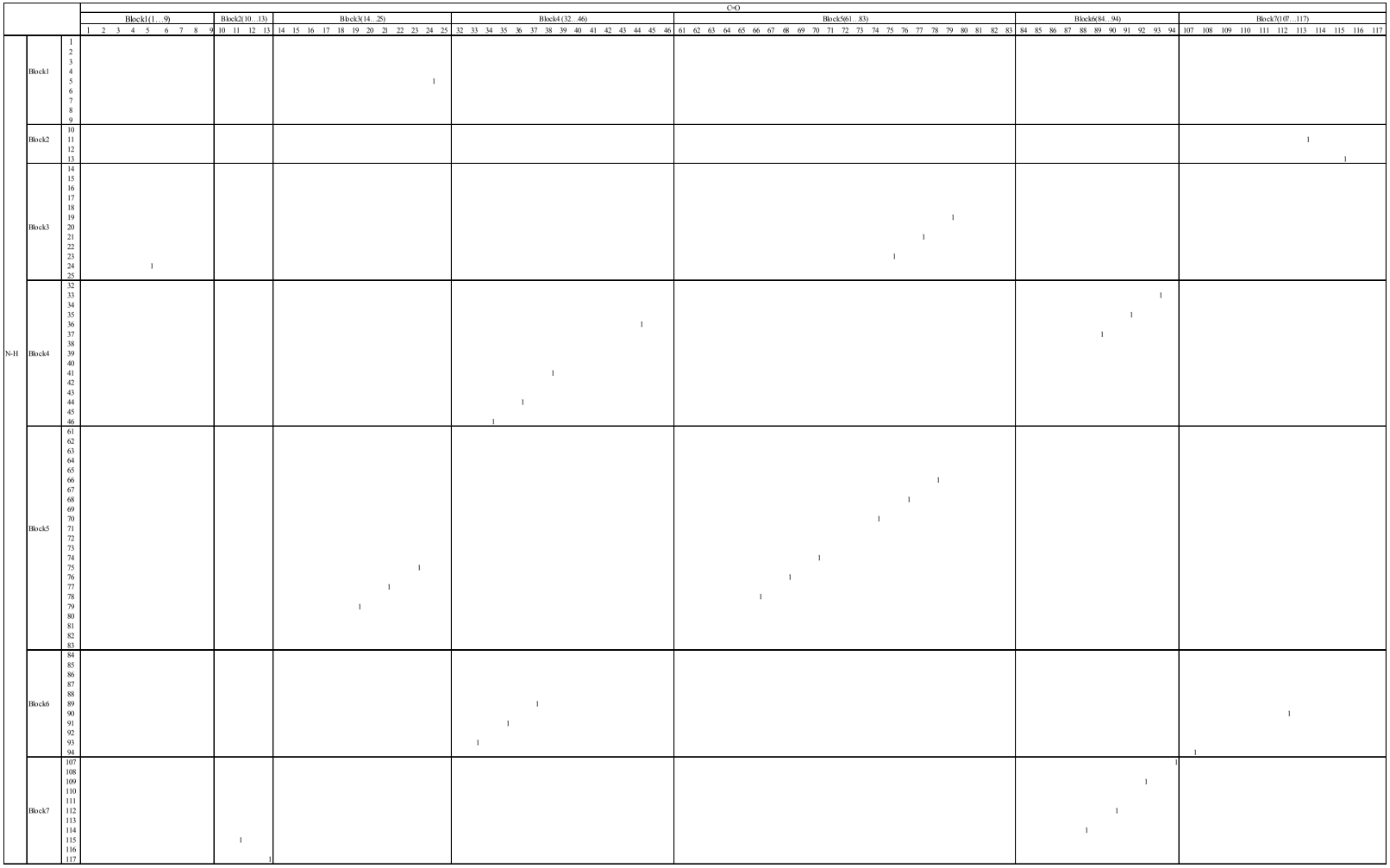


Table 2.4: Summary of blocks and hydrogen bonds conserved in TCR β domains.



2.4 Identify and classify immunoglobulin variable domains

To identify and classify immunoglobulin variable domains, we should compare the hydrogen bond pattern of the examined chain to the templates based on the conserved main chain hydrogen bonds. Each template has been partitioned into several blocks, a number of non-empty Pattern[i][j] in the template are obtained [Tables 2.1, 2.2, 2.3 and 2.4]. For example, there are 12 patterns in VL domain template [Table 2.1] including two intra-block hydrogen bond patterns (Pattern[4][4] and Pattern[5][5]) and five pairs inter-block hydrogen bond patterns (Pattern[1][3] and Pattern[3][1], Pattern[2][7] and Pattern[7][2], Pattern[3][5] and Pattern[5][3], Pattern[4][6] and Pattern[6][4], Pattern[6][7] and Pattern[7][6]).

Next, we record the hydrogen bonds of the template using the method shown in Figure 2.5. Here, sum_block refers to the total number of blocks in the template; Blockstart[i] and Blocklength[i] are the starting point and the length of the Block[i], respectively. The two dimensional array Tempposition records the positions of hydrogen bonds in the template. Table 2.5 shows the format of the recorded hydrogen bonds in the VL template using the method shown in Figure 2.5.

```
if ((out1=fopen("Hbondtemplate.txt","wb"))==NULL
{
        printf("Hbondtemplate.txt cannot be opened!\n");
}
else
{
        for (i=0;i<sum_block;i++)
        {
                for (j=0;j<sum_block;j++)
                {
                        for (xpos=Blockstart[i];xpos<Blockstart[i]+Blocklength[i];xpos++)
                        {
                                for (ypos=Blockstart[j];ypos<Blockstart[j]+Blocklength[j];ypos++)
                                {
                                        if (Hbondtemplate[xpos][ypos]==1)
                                        {
                                                Tempposition[count][0]=xpos;
                                                Tempposition[count][1]=ypos;
                                                fprintf(out1,"%d  %d\n",xpos,ypos);
                                                count=count+1;
                                        }
                                }
                        }
                }
        }
        closeresult=fclose(out1);
        if (closeresult!=0)
        {
                printf("Hbondtemplate.txt cannot be closed!\n");
        }
}
```

Record the template Hbond position for each pair of blocks, and record the positions of hydrogen bonds into file "Hbondtemplate.txt"

Figure 2.5: Record Hydrogen bonds of template.

Table 2.5: Recorded hydrogen bonds in VL template (file "Hbondtemplate.txt").

	C=O	N-H
Pattern[1][3]	5	24
Pattern[2][7]	9	103
	11	105
	13	107
Pattern[3][1]	24	5
Pattern[3][5]	19	75
	21	73
	23	71
Pattern[4][4]	33	50
	35	47
	35	48
	37	45
	39	42
	45	37
	47	55
	48	35
	49	53
	53	49
Pattern[4][6]	34	89
	36	87
	38	85
Pattern[5][3]	71	23
	73	21
	75	19
Pattern[5][5]	61	76
	63	74
	65	72
	67	70
	70	67
	72	65
	74	63
Pattern[6][4]	85	38
	87	36
	89	34
Pattern[6][7]	84	104
	86	102
	88	99
	90	97
Pattern[7][2]	103	11
	105	13
Pattern[7][6]	97	90
	102	86

In order to compare the whole hydrogen bond pattern between the examined chain and the template, we can compare each non-empty Pattern[i][j] step by step instead of the whole pattern at one time. Finally, we count the total number of the found Pattern[i][j] of the template in hydrogen bond pattern of examined chain to determine whether the examined chain matches the template domain.

When we analysed the hydrogen bond pattern Pattern[i][j] of each template, we found that, usually, intra-block hydrogen bond patterns have more hydrogen bonds than inter-block hydrogen bond patterns. For example, there are 10 and 7 hydrogen bonds in two intra-block hydrogen bond patterns (Pattern[4][4] and Pattern[5][5]) of the VL template, much more than the hydrogen bonds in inter-block hydrogen bond patterns; these have at most 4 hydrogen bonds. Therefore, it is more likely that we will find a unique match when searching for an intra-block pattern. Moreover, for the intra-block hydrogen bond patterns formed between the main chain C=O groups and N-H groups of the same block, since we only need to search the hydrogen bond patterns in the main diagonal line from the N-terminal to the C-terminal, it is faster to search for them than inter-block hydrogen bond patterns between two different blocks. Therefore, we search for the intra-block hydrogen bond patterns at first. Searching areas for other blocks are adjusted after finding this kind of patterns. Then, we search for inter-block hydrogen bond patterns.

2.4.1 Searching for intra-block hydrogen bond patterns

Since hydrogen bonds usually occur between two antiparallel strands in these templates, most of the hydrogen bond patterns are in a line from the lower left to the upper right, perpendicular to the main diagonal, as the one shown in Figure 2.6. Sometimes there are more than two antiparallel strands involved to form one hydrogen bond pattern, for example, there are three antiparallel strands involved in Pattern[4][4] in the VL template, which gives hydrogen bonds in two lines. In some cases, hydrogen bonds form between two parallel strands, which also gives hydrogen bonds in one line from the upper left to the lower right, for example, Pattern[2][7] and Pattern[7][2] in VL, TCR α and β templates, Pattern[2][8] and Pattern[8][2] in VH template. All these hydrogen bond patterns are shown in Tables 2.1, 2.2, 2.3 and 2.4.

Suppose we want to search for an intra-block hydrogen bond Pattern[i][i] shown in Figure 2.6. Suppose in total there are n hydrogen bonds in all of the Pattern[c][r] when c<i or c=i and r<i. The numbers in parenthesis shown in Figure 2.6 are the sequence numbers we record with the corresponding hydrogen bonds. According to these

sequence numbers, we can easily find the corresponding hydrogen bond positions from file “Hbondtemplate.txt”.

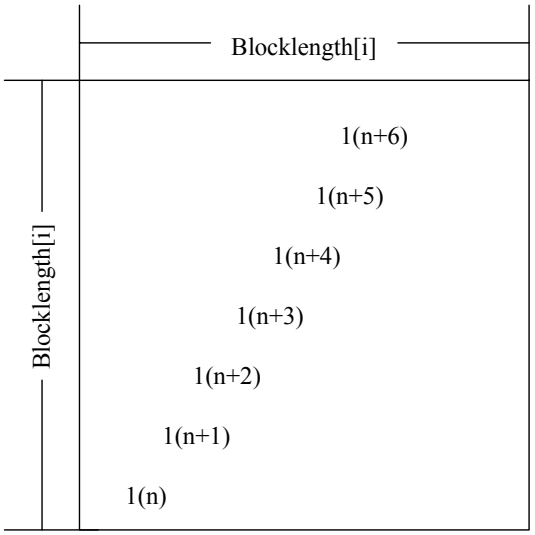


Figure 2.6: One example of intra-block hydrogen bond pattern.

Suppose the length of the examined chain is L. A hydrogen bond pattern in a square with the side of L is obtained after calculating the energy between main chain C=O groups and N-H groups of the examined chain. In this case, we divide the hydrogen bond pattern of the examined chain into small squares with the side length same as the length of Block[i]. Since this kind of hydrogen bond pattern lies in one block of the template, we only need to check the squares on the main diagonal line of the hydrogen bond pattern of the examined chain.

We call the process of searching the observed hydrogen bond pattern of the examined chain from the N-terminal to the C-terminal of the searching area for a specific template pattern a ***round***. We searched the whole hydrogen bond pattern of the examined chain for Pattern[i][i] in the first round. If no match is found, we ignore one more residue of the examined chain from the N-terminal for each round, search the hydrogen bond pattern of the examined chain with one residue less for each round for Pattern[i][i] until it is found. If it is still not found until the total number of missed residue is up to the length of Block[i], we relax the hydrogen bond pattern Pattern[i][i] by allowing one hydrogen bond to be missed step by step, and repeat the searching process. If it is found, we

continue searching for other patterns. Otherwise, we give up and draw the conclusion that the examined chain does not match the template.

For example, if we search the hydrogen bond pattern of the examined chain for Pattern[4][4] of the VL template, we partition the pattern of the examined chain into small squares with the side of 23 residues long, the same as the length of the Block[4] of the VL template, from residue 0 of the N-terminal to the C-terminal, then we compare Pattern[4][4] to those patterns on the diagonal line of the pattern of the examined chain from N-terminal. Since we cannot find the corresponding pattern in that of the examined chain in this round, we ignore the first residue with position number 0 from the N-terminal and then partition and search the rest of the pattern of the examined chain. If no match found in this round, we continue ignore one more reside from the N-terminal and partition and search the rest of the pattern of the examined chain step by step until one match is found. If there is still no match until the total number of ignored residue from the examined chain is up to the 23 residues, we search for Pattern[4][4] again by permitting one hydrogen bond to be missed from the beginning until we find the match or the total number of missed hydrogen bonds is up to 3.

Figure 2.7 shows the procedure for searching for Pattern[i][i] in one round, distance refers to the number of residues permitted to be missed in a specific round and its range can be from 0 to Blocklength[i]-1. Here, we use block_distance to limit the maximum distance between the starting points of two blocks and it can be 50 residues for two consecutive blocks because these cannot be too far apart, and a reliable match can be found using this limit. Hbond[0] and Hbond[2] records the starting points of two consecutive blocks or two blocks involved in two consecutive intra-block hydrogen bond patterns, while Hbond[1] and Hbond[3] records the ending points of these two blocks. For example, when we search for Pattern[4][4] of the VL template, Hbond[0] and Hbond[2] refer to the starting points of Block[4] and Block[3], Hbond[1] and Hbond[3] refer to the ending points of Block[4] and Block[3].

```
c=(float)((Hbondend-Hbondstart-distance+1)/Blocklength);
k=(int)floor(c);
for (i=0;i<k;i++)
{
        if (pattern==1&&Hbond[0]-Hbond[2]<block_distance)
        {
                break;
        }
        count1=0;
        start=Blocklength*i+distance+Hbondstart;
        end=start+Blocklength;

        for (x=start;x<end;x++)
        {
                for (y=start;y<end;y++)
                {
                        if(x<chain_size&&y<chain_size&&Hbondno[x][y]==1)
                        {
                                count1=count1+1;
                        }
                }
        }

        if (count1>=SumHbond-Hbond_less)
        {
                count2=0;
                for (q=0;q<SumHbond;q++)
                {
                        xpos=start+Tempposition[sum+q][0]-Blockstart;
                        ypos=start+Tempposition[sum+q][1]-Blockstart;
                        if (xpos<chain_size&&ypos<chain_size&&Hbondno[xpos][ypos]==1)
                        {
                                count2=count2+1;
                        }
                }
                if (count2>=SumHbond-Hbond_less)
                {
                        Hbondstart=start;
                        if (Hbondstart-Hbond[2]<block_distance)
                        {
                                pattern=1;
                                for (l=0;l<SumHbond;l++)
                                {
                                        xpos=start+Tempposition[sum+l][0]-Blockstart;
                                        ypos=start+Tempposition[sum+l][1]-Blockstart;
                                        if (xpos<chain_size&&ypos<chain_size&&position[xpos]==position[xpos-1])
                                        {
                                                printf("(%d  %d)  (%d%c  %d)  %d\n",Tempposition[sum+l][0],Tempposition[sum+l][1],
                                                        position[xpos],sign[xpos][0],position[ypos],Hbondno[xpos][ypos]);
                                        }
                                        else if (xpos<chain_size&&ypos<chain_size&&position[ypos]==position[ypos-1])
                                        {
                                                printf("(%d  %d)  (%d  %d%c)  %d\n",Tempposition[sum+l][0],Tempposition[sum+l][1],
                                                        position[xpos],position[ypos],sign[ypos][0],Hbondno[xpos][ypos]);
                                        }
                                        else if (xpos<chain_size&&ypos<chain_size&&position[xpos]==position[xpos-1]&&position[ypos]==position[ypos-1])
                                        {
                                                printf("(%d  %d)  (%d%c  %d%c)  %d\n",Tempposition[sum+l][0],Tempposition[sum+l][1],
                                                        position[xpos],sign[xpos][0],position[ypos],sign[ypos][0],Hbondno[xpos][ypos]);
                                        }
                                        else if (xpos<chain_size&&ypos<chain_size&&position[xpos]!=position[xpos-1]&&position[ypos]!=position[ypos-1])
                                        {
                                                printf("(%d  %d)  (%d  %d)  %d\n",Tempposition[sum+l][0],Tempposition[sum+l][1],
                                                        position[xpos],position[ypos],Hbondno[xpos][ypos]);
                                        }
                                        else printf("(%d  %d)\n",Tempposition[sum+l][0],Tempposition[sum+l][1]);
                                }
                                Hbond[0]=start;
                                Hbond[1]=end-1;
                        }
                }
        }
}
```

Calculate the number of blocks divided in the examined chain at each round

Record the total number of hydrogen bonds of within each individual block of the examined chain

Calculate the number of corresponding hydrogen bonds in squares of the examined domain with enough hydrogen bonds inside

Print hydrogen bond positions in the matched square

Figure 2.7: Procedure for searching for intra-block hydrogen bond pattern in one round.

After Pattern[i][i] is found, the searching areas for other blocks are adjusted. For those blocks before Block[i], the searching area should end before the starting point of Block[i]; for those blocks after Block[i], the searching area should start after the ending point of Block[i]. After reducing the searching areas, we continue searching for other patterns.

2.4.2 Searching for inter-block hydrogen bond patterns

After all the non-empty Pattern[i][i] of the template have been found, we begin to search for Pattern[i][j] between different blocks. For each pair of Pattern[i][j] and Pattern[j][i], there are hydrogen bonds in Pattern[j][i] only when hydrogen bonds exist in Pattern[i][j]. During this stage, we search for Pattern[i][j] and Pattern[j][i] at the same time to increase the reliability of the results. In this case, we sort each pair of Pattern[i][j] and Pattern[j][i] based on the total number of hydrogen bonds in these patterns, in decreasing order, to decide the order in which to search for these patterns. This makes it more likely that we will find unique matches when searching for inter-block patterns.

Since we have minimized the searching area for each block, in order to search for Pattern[i][j], we only need to partition the area with the width the same as the searching area of Block[i] and the height the same as the searching area of Block[j] into smaller rectangles with the size same as Pattern[i][j]. Figure 2.8 shows the method used to search for Pattern[i][j] and Pattern[j][i] at the same time. Here, column_distance and row_distance are the numbers of residues ignored in searching area of Block[column] and Block[row], respectively. Hbondstart[column] and Hbondstart[row] refer to the starting point of the searching area of Block[column] and Block[row], respectively.

```
m=(float)((Hbondend[column]-Hbondstart[column]-column_distance+1)/Blocklength[column]);
k=(int)floor(m)+1;
n=(float)((Hbondend[row]-Hbondstart[row]-row_distance+1)/Blocklength[row]);
r=(int)floor(n)+1;
for (i=0;i<k;i++)
{
        column_start=Blocklength[column]*i+column_distance+Hbondstart[column];
        column_end=column_start+Blocklength[column];
        for (j=0;j<r;j++)
        {
                row_start=Blocklength[row]*j+row_distance+Hbondstart[row];
                row_end=row_start+Blocklength[row];
                count=0;

                for (x=column_start;x<column_end;x++)
                {
                        for (y=row_start;y<row_end;y++)
                        {
                                if(x<chain_size&&y<chain_size&&Hbondno[x][y]==1)
                                {
                                        count=count+1;
                                }
                        }
                }

                if (count>=SumHbond[column][row]-Hbond_less1)
                {
                        count1=0;
                        for (q=0;q<SumHbond[column][row];q++)
                        {
                                xpos=column_start+Tempposition[sum1+q][0]-Blockstart[column];
                                ypos=row_start+Tempposition[sum1+q][1]-Blockstart[row];
                                if (xpos<chain_size&&ypos<chain_size&&Hbondno[xpos][ypos]==1)
                                {
                                        count1=count1+1;
                                }
                        }

                        if (count1>=SumHbond[column][row]-Hbond_less1)
                        {
                                if (SumHbond[row][column]!=0)
                                {
                                        count2=0;
                                        for (l=0;l<SumHbond[row][column];l++)
                                        {
                                                xpos=row_start+Tempposition[sum2+l][0]-Blockstart[row];
                                                ypos=column_start+Tempposition[sum2+l][1]-Blockstart[column];
                                                if (xpos<chain_size&&ypos<chain_size&&Hbondno[xpos][ypos]==1)
                                                {
                                                        count2=count2+1;

                                                }
                                        }

                                        if (count2>=SumHbond[row][column]-Hbond_less2)
                                        {
                                                /*Here print Pattern[column][row] and Pattern[row][column]*/
                                                . . . . . .
                                        }
                                }
                        }
                }
        }
}
```

Calculate the number of blocks to search in X and Y directions

Record the searching area for each possible Block[column]

Record the searching area for each possible Block[row]

Count the total number of hydrogen bonds in each searched rectangle

Count the total number of hydrogen bonds at corresponding positions in Pattern[column][row]

Count the total number of hydrogen bonds at corresponding positions in Pattern[row][column]

Figure 2.8: Procedure for searching for a pair of inter-block hydrogen bond patterns in one round.

When we search for intra-block hydrogen bond patterns or a pair of inter-block hydrogen bond patterns, we search for the same patterns from that of the examined chain at first. If no match is found, we then relax the hydrogen bond pattern to find a similar one by permitting one hydrogen bond missing step by step until we find it. Finally, we count the total number of intra-block and inter-block hydrogen bond patterns found in the pattern of the examined chain. If we find all of them, the examined chain matches the template. For example, if we found 12 patterns of the VL template from the patterns of the examined chain, we say that this examined chain matches VL template.

Chapter 3

Results and Discussion

3.1 Template calibration

The first set of tests performed in this project aimed to check the VL, VH, TCR α , β and the immunoglobulin variable domain templates to see whether immunoglobulin variable domains can be identified reliably using these templates. Figure 3.1 shows the relationship of the VL, VH, TCR α , TCR β and immunoglobulin variable domains. From this figure we can see that the subsets of VL, VH, TCR α , TCR β variable domains belong to the family of immunoglobulin variable domains. Therefore, the known VL, VH, TCR α and TCR β variable domains should be compared to not only their own templates, but also the immunoglobulin variable domain template to test the reliability of these templates.

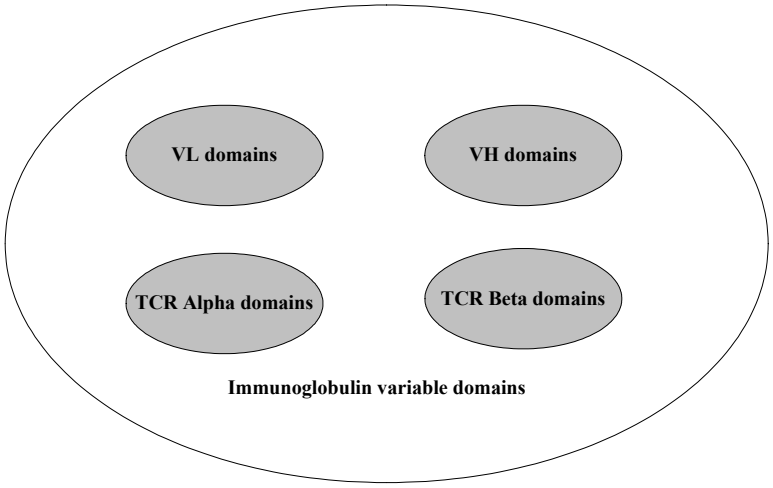


Figure 3.1 The relationship of VL, VH, TCR α , TCR β and immunoglobulin variable domains.

For example, we compared known VH domains to VH domain template and the immunoglobulin variable domain template to test the reliability of VH and immunoglobulin variable domain templates. Similarly, we compared the known VL

domains, known TCR α domains and known TCR β domains to the templates they are supposed to match and the immunoglobulin variable domain template to test their reliability.

The results here show that the templates of VL, VH and TCR α domains work well. It means that known VL domains can be classified correctly using the VL template. Similarly, VH and TCR α domains can also be classified correctly using the VH and TCR α templates, respectively. However, the TCR β domain and immunoglobulin variable domain templates did not work well at first. Therefore, these two templates were improved [Section 3.1.1 and Section 3.1.2] before being used to classify the corresponding domains.

3.1.1 New TCR β domain template

The new template of the conserved main chain hydrogen bond pattern in TCR β domain is shown in Figure 3.2. This improved template was obtained based on the results of our initial tests by comparing known TCR β domains to the TCR β domain template shown in Figure 1.4. During this step, we found that there is usually false match with Pattern[4][6] and Pattern[6][4], because in most cases there is one pair of hydrogen bonds between residues 31 and 95, which makes the found Block[4] two residues before and the found Block[6] two residues after the corresponding blocks. After adding this pair of hydrogen bonds between residues 31 and 95, the match is reliable.

Compared to the old one, seven more hydrogen bonds were added in the new template. In this figure, the seven dashed lines represent the seven hydrogen bonds added in the new template. These added main chain hydrogen bonds are also listed in Table 3.1. Here the position codes follow the Chothia Numbering Scheme. The lengths of blocks were also adjusted to include these added hydrogen bonds in order to improve the reliability of this template.

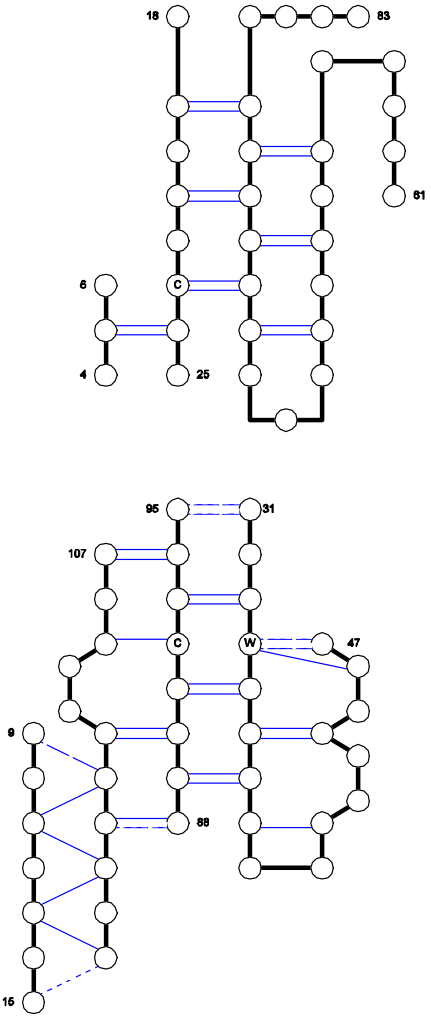


Figure 3.2: Improved plane of the $\beta - sheet$ framework conserved in TCR β domain.

Table 3.1: Added hydrogen bonds in the new TCR β domain template.

C=O	N-H
31	95
95	31
34	47
47	34
114	88
9	113
117	15

The main reason to add more hydrogen bonds is that more hydrogen bonds in Pattern[i][j] will increase the probability to get only one match for this pattern. For example, if only one part of the hydrogen bond pattern between the main chain C=O group and N-H group of residues of the examined chain is same as the hydrogen pattern between Block[4] and Block[6] of the TCR β domain template shown in Figure 3.2, which means there are four pairs of main chain hydrogen bonds with the corresponding positions same as those between Block[4] and Block[6] of the template. If we search the hydrogen bond pattern of the examined chain only for the lower three pair hydrogen bonds, we can get two matches either the upper three pairs or the lower three pairs. However, if we search for these four pair hydrogen bonds, we can get only one match. Thus, the reliability of this template was improved to be used for classifying TCR β variable domains by adding more conserved main chain hydrogen bonds.

3.1.2 New immunoglobulin variable domain template

Figure 3.3 shows the improved $\beta - sheet$ framework conserved in VL, VH, TCR α and TCR β variable domains. In this figure, four numbers are shown at the ends of each framework region. The upper pair, M_1/M_2, are the VL and VH sequence position codes proposed in [Chothia and Lesk, 1987]; the lower pair, N_1/N_2, are the TCR α and β domain sequence position codes proposed in [Chothia et al., 1988].

The new immunoglobulin variable domain template was developed in two steps. First, we compared the four templates including the VL, VH, TCR α and β templates. When we superposed them, we found a common core pattern based on conserved main chain hydrogen bonds. Based on our initial tests by comparing known VL, VH, TCR α and β variable domains to the old template proposed in [Chothia et al., 1988], we then modified the pattern first by removing one main chain hydrogen bond and adding two more. The reason to remove the hydrogen bond represented by dotted line in Figure 3.3 is because most of TCR α domains do not have this hydrogen bond and the results of searching for other kinds of variable domains are not affected after removing this one. In Figure 3.3, two hydrogen bonds represented by dashed lines are added in order to make automatic identification of variable domain structures more reliable. Table 3.2

summarises the difference between the new template and the old one. The position codes in Table 3.2 are TCR α position codes using the Chothia Numbering Scheme. The corresponding position codes in other variable domains can be found easily from Figure 3.3.

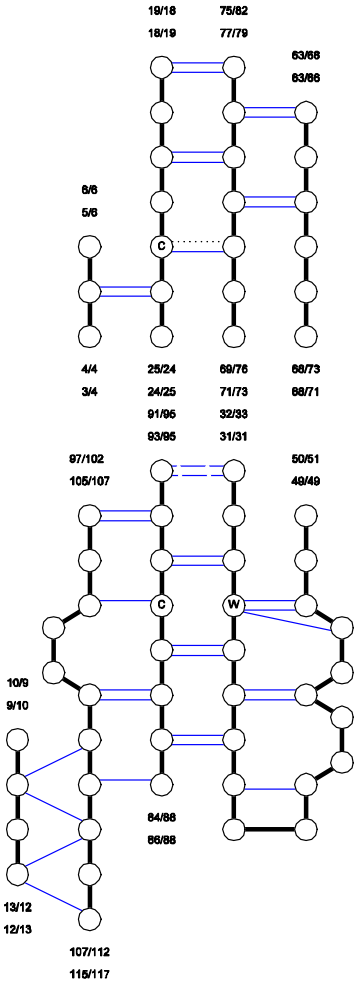


Figure 3.3: Improved plane of the $\beta-sheet$ framework conserved in VL, VH, TCR α and TCR β domains.

Table 3.2: Different hydrogen bonds between the new immunoglobulin variable domain template and the old one.

C=O	N-H	New template	Old template
73	22	-	+
31	93	+	-
93	31	+	-

"+": Has this hydrogen bond; "-": Does not have this hydrogen bond.

Although VL, VH, TCR α and β variable domains share the common core pattern shown in Figure 3.3, they have different position codes at structurally equivalent positions even using the same numbering scheme (e.g. the Chothia Numbering Scheme). In this project, a new numbering scheme is proposed to help the process of finding structurally equivalent positions automatically. Table 3.3 lists the new codes at the corresponding positions using the Chothia Numbering Scheme in VL, VH, TCR α and β variable domains. Using the new code proposed here, block i begins with position 100*i from its N-terminal. The first half of those residues between two consecutive blocks are numbered consecutively after the former block, while the second half are numbered consecutively before the later block. Since VL and VH domains have one more block than TCR α and β variable domains, the corresponding positions are ignored in TCR α and β variable domains.

Table 3.3: Proposed New Codes vs. the codes using Chothia Numbering Scheme.

New Code	VL	VH	TCR α	TCR β	New Code	VL	VH	TCR α	TCR β
100	4	4	3	4	501	54	59		
101	5	5	4	5	502	55	60		
102	6	6	5	6	600	63	68	63	66
200	10	9	9	10	601	64	69	64	67
201	11	10	10	11	602	65	70	65	68
202	12	11	11	12	603	66	71	66	69
203	13	12	12	13	604	67	72	67	70
300	19	18	18	19	605	68	73	68	71
301	20	19	19	20	700	69	76	71	73
302	21	20	20	21	701	70	77	72	74
303	22	21	21	22	702	71	78	73	75
304	23	22	22	23	703	72	79	74	76
305	24	23	23	24	704	73	80	75	77
306	25	24	24	25	705	74	81	76	78
400	33	34	32	32	706	75	82	77	79
401	34	35	33	33	800	84	88	86	88
402	35	36	34	34	801	85	89	87	89
403	36	37	35	35	802	86	90	88	90
404	37	38	36	36	803	87	91	89	91
405	38	39	37	37	804	88	92	90	92
406	39	40	38	38	805	89	93	91	93
407	40	41	39	39	806	90	94	92	94
408	41	42	40	40	900	97	102	105	107
409	42	43	41	41	901	98	103	106	108
410	43	44	42	42	902	99	104	107	109
411	44	45	43	43	903	100	105	108	110
412	45	46	44	44	904	101	106	109	111
413	46	47	45	45	905	102	107	110	112
414	47	48	46	46	906	103	108	111	113
415	48	49	47	47	907	104	109	112	114
416	49	50	48	48	908	105	110	113	115
417	50	51	49	49	909	106	111	114	116
500	53	58			910	107	112	115	117

3.2 Template verification

After the templates had been improved to increase their reliability in identifying and classifying immunoglobulin variable domains, tests were carried out to verify these templates. In this case, for each known domain, we compared it not only to the template it is supposed to match; we also compared it to other templates to verify the reliability of those templates. For example, we compared known VL domains not only to VL template and the improved immunoglobulin variable domain template; we also compared them to other templates including the VH, TCR α and β variable domain templates. Similar tests were carried out for other kinds of known variable domains.

The program was tested with the following 65 structures from the Protein Data Bank.

- Antibody structures with VL and VH domains:
  - Intact antibody: 1IGT, 1IGY;
  - Fab: 12E8, 15C8, 1ACY, 1AD0, 1AD9, 1A0Q, 1ADQ, 1AE6, 1AFV, 1AHW, 1AI1, 1A14, 1A3L, 1A3R, 1A4J, 1A5F, 1A6T;
  - Fv: 1A6V, 1A6W, 1A7N, 1A7O, 1A7P, 1A7Q, 1A7R, 1A6U;
  - scFv: 1LMK(diabody), 1NQB, 2AP2, 1AP2, 1QOK, 1F3R, 1H8N, 1I3G, 1H8O, 1H8S;
  - anti-idiotope:1CIC;
- Bence-Jones proteins with only VL domains: 1BWW, 1REI, 1AR2, 2RHE, 1BJM, 3BJL, 4BJL;
- Camel antibodies with only VH domains: 1JTP, 1JTO, 1MEL, 1F2X, 1G6V, 1B2Q, 1JTT;
- T cell receptors with α and β domains: 1BD2, 1QRN, 1FYT, 1QSE, 1QSF, 1AO7, 1TCR, 1KB5, 1NFD, 2CKB, 1FO0, 1D9K, 1G6R.

Table 3.4: Summary of results of tests.

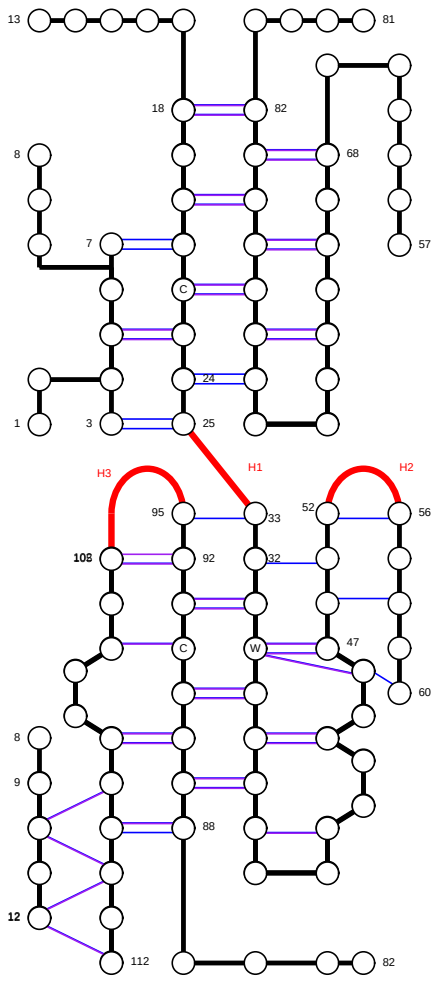
		Identified as					Total
		VL	VH	TCR α	TCR β	Variable domain	
Known to be	VL	66	0	0	0	64	66
	VH	0	69	54	0	66	69
	TCR α	0	17	16	0	16	17
	TCR β	0	0	0	16	14	17
Total						160	169

Table 3.4 summarises the results of these tests. In total, 169 known immunoglobulin variable domains have been checked and 160 of them matched the improved template of immunoglobulin variable domain, which means that the improved one works well for identifying immunoglobulin variable domains. Among these, we checked 66 VL domains and 17 TCR β variable domains. From this table, we can see that all of the known VL variable domains matched the VL template, and most of the known TCR β variable domains matched the improved TCR β variable domain template. Moreover, no false positive is found for these known VL and TCR β variable domains. In other words, all known VL domains did not match the VH, TCR α and β variable domain templates and all known TCR β variable domains did not match the VL, VH and TCR α templates. Therefore, these two kinds of templates work well to classify VL and TCR β variable domains.

We also checked 69 VH domains and 17 TCR α variable domains. Although all of the VH domains matched the VH domain template and most of the TCR α variable domains matched the TCR α variable domain template, most of the VH domains could also match the TCR α variable domain template and all of the TCR α variable domains matched the VH domain template. In this case, there is misclassification between these two kinds of variable domains. The main reason is the similar structure of these two kinds of domains. The superposed figure of VH and TCR α variable domain templates is shown in Figure 3.4. In this figure, blue lines represent hydrogen bonds in the VH template, while purple lines represent hydrogen bonds in the TCR α variable domain template. From this figure, we can see that the only difference between them is that there are 12 hydrogen bonds existing in the VH domain template but not in the TCR α

variable domain template and one in the TCR α variable domain template but not in the VH domain template. Table 3.5 lists the hydrogen bonds that are different between the VH variable domain template and the TCR α variable domain template. All the position codes in Table 3.5 are VH position codes using the Chothia Numbering Scheme.

However, if we compare the structures of VH and TCR α variable domain templates to the VL and TCR β domain templates (Figure 1.3 and Figure 1.4), different structures were found. Therefore, misclassification was found between VH and TCR β variable domains, but no misclassification between them and other variable domains.



Blue: Hydrogen bonds in the VH template;

Purple: Hydrogen bonds in the TCR α variable domain template.

Figure 3.4: Superposed template of VH and TCR α variable domain templates.

Table 3.5: Different hydrogen bonds between VH and TCR α templates.

C=O	N-H	VH template	TCR α template
3	25	+	-
25	3	+	-
7	21	+	-
21	7	+	-
24	76	+	-
76	24	+	-
109	88	+	-
33	95	+	-
34	51	+	-
56	52	+	-
50	58	+	-
48	60	+	-
94	102	-	+

"+": Has this hydrogen bond; "-": Does not have this hydrogen bond.

Chapter 4

Conclusions

In this project, we have improved the TCR β and immunoglobulin variable domain templates based on conserved main chain hydrogen bond pattern to increase the reliability of these templates in identifying and classifying these variable domains. The C program developed here can successfully identify immunoglobulin variable domains and classify VL and TCR β variable domains, although there is some misclassification between VH and TCR α variable domains because of the similarity of their structures.

Although it is still difficult to discriminate VH and TCR α variable domains only considering the conserved main chain hydrogen bond patterns, the program developed in this project will help the process of finding residues at structurally equivalent positions from different variable domains automatically. Therefore, it will make the process of extending an antibody structure database [Kemp et al., 1994] much more easy and convenient than before.

Generally, immunoglobulin variable domains can be identified and classified reliably based on the conserved main chain hydrogen bonds although false positives exist among VH and TCR α variable domains. Since the structure of CDR regions has a high degree specificity, false positives may be avoided by combining conserved main chain hydrogen bond pattern with the structure information of the different CDR regions. In this way, immunoglobulin variable domains can be identified better in the future and the existing antibody database can be updated automatically to include more immunoglobulin variable domains.

References

Bernstein, F., Koetzle, T., Williams, G., Mayer, E., Bruce, M., Rodgers, J., Kennard, O., Shimanouchi, T., and Tasummi, M. (1977). The Protein Data Bank: a Computer-Based Archival File for Macromolecular Structures. *J. Mol. Biol*., 112:535-542.

Chothia, C., Boswell, D.R., and Lesk, A.M. (1988). The outline structure of the T-cell alpha-beta receptor. *EMBO J*., 7:3745-3755.

Chothia, C. and Lesk, A.M. (1987). Canonical Structures for the Hypervariable Regions of Immunoglobulins. *J. Mol. Biol*., 196:901-917.

Kabsch, W. and Sander, C. (1983). Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features. *Biopolymers*, 22:1577-2637.

Kemp, G.J.L., Jiao, Z., Gray, P.M.D., and Fothergill, J.E. (1994). Combining Computation with Database Access in Biomolecular Computing. In Litwin, W. and Risch, T., editors, *Applications of Databases: Proceedings of the First International Conference*, pages 317-335. Springer-Verlag.

Murzin, A., Brenner, S., Hubbard, T., and Chothia, C. (1995). SCOP: A Structural Classification of Proteins Database for the Investigation of Sequences and Structures. *J. Mol. Biol*., 247:536-540.

Appendix A

Maintenance Manual

A.1 Project files

Figure A.1 lists the files and directories in the root directory. All C source codes are stored in directory "src" shown is Figure A.2, compiled file "search" in directory "bin" shown in Figure A.3, the perl file and all the text files produced by C programs here are stored in directory "Data" shown in Figure A.4. There are also two shell files runperl.sh and install.sh in the root directory to help to run these perl and C programs. The directory of tcl stores the tcl/tk source code, all the txt files needed in this program and the produced file "temp.ps", shown in Figure A.5.

```
>ls -l
total 18
drwxr-xr-x   2 md1huihu cthstud     1024 Jan 12 12:32 Data
-rw-r--r--   1 md1huihu cthstud     2212 Dec 31 01:14 Makefile
drwxr-xr-x   2 md1huihu cthstud      512 Jan 12 12:32 bin
-rwxr-xr-x   1 md1huihu cthstud       31 Dec 31 01:15 install.sh
-rwxr-xr-x   1 md1huihu cthstud       27 Dec 31 01:15 runperl.sh
drwxr-xr-x   2 md1huihu cthstud     1024 Jan 12 12:36 src
drwxr-xr-x   2 md1huihu cthstud     1024 Jan 12 12:32 tcl
```

Figure A.1: Contents of root directory.

```
> ls -l
total 276
-rw-r--r--   1 md1huihu cthstud     5821 Jan  5 10:02 HbondtempTAlpha.c
-rw-r--r--   1 md1huihu cthstud     5969 Jan  5 10:04 HbondtempTBeta.c
-rw-r--r--   1 md1huihu cthstud     5841 Jan  5 10:06 HbondtempV.c
-rw-r--r--   1 md1huihu cthstud     6173 Jan  5 10:10 HbondtempVH.c
-rw-r--r--   1 md1huihu cthstud     5983 Jan  5 10:12 HbondtempVL.c
-rw-r--r--   1 md1huihu cthstud     2261 Jan  5 10:14 Hcoordinate.c
-rw-r--r--   1 md1huihu cthstud     2692 Jan  5 10:15 ReadHbond.c
-rw-r--r--   1 md1huihu cthstud     4680 Jan  5 10:16 ReadPatterntemp.c
-rw-r--r--   1 md1huihu cthstud     5935 Jan  5 10:20 getHbond.c
-rw-r--r--   1 md1huihu cthstud      590 Jan  5 10:18 getHbondScope.c
-rw-r--r--   1 md1huihu cthstud     1233 Jan  5 10:19 getSum.c
-rw-r--r--   1 md1huihu cthstud    13270 Jan  5 09:59 main.c
-rw-r--r--   1 md1huihu cthstud     1916 Dec 31 01:12 patternsearch.h
-rw-r--r--   1 md1huihu cthstud    48057 Jan  5 10:33 printHbond.c
-rw-r--r--   1 md1huihu cthstud     8929 Jan  5 10:35 searchDiagonalPattern.c
-rw-r--r--   1 md1huihu cthstud    13559 Jan  5 10:40 searchPattern.c
-rw-r--r--   1 md1huihu cthstud      720 Jan  5 10:40 size.c
-rw-r--r--   1 md1huihu cthstud      445 Jan  5 10:40 sortHbond.c
```

Figure A.2: Contents of directory "src".

```
>ls -l
total 304
-rwxr-xr-x   1 md1huihu cthstud   145324 Jan 12 12:38 search
```

Figure A.3: Contents of directory “bin”.

```
>ls -l
total 2230
-rw-r--r--   1 md1huihu cthstud       24 Jan 12 12:38 Blocklength.txt
-rw-r--r--   1 md1huihu cthstud       26 Jan 12 12:38 Blockstart.txt
-rw-r--r--   1 md1huihu cthstud     6325 Dec 31 01:30 CAcoordinate.txt
-rw-r--r--   1 md1huihu cthstud     6327 Dec 31 01:30 Ccoordinate.txt
-rw-r--r--   1 md1huihu cthstud      748 Dec 31 01:30 Chainposition.txt
-rw-r--r--   1 md1huihu cthstud      856 Dec 31 01:30 Chainresidue.txt
-rw-r--r--   1 md1huihu cthstud     3452 Jan 12 12:38 CorrespondingChain.txt
-rw-r--r--   1 md1huihu cthstud     1416 Jan 12 12:38 CorrespondingHbond.txt
-rw-r--r--   1 md1huihu cthstud     1144 Jan 12 12:38 Hbond.txt
-rw-r--r--   1 md1huihu cthstud        4 Jan 12 12:38 HbondSize.txt
-rw-r--r--   1 md1huihu cthstud        3 Jan 12 12:38 HbondTempSize.txt
-rw-r--r--   1 md1huihu cthstud    91806 Jan 12 12:38 Hbondno.txt
-rw-r--r--   1 md1huihu cthstud   259560 Jan 12 12:38 Hbondnotemp.txt
-rw-r--r--   1 md1huihu cthstud      299 Jan 12 12:38 Hbondtemplate.txt
-rw-r--r--   1 md1huihu cthstud     6326 Dec 31 01:30 Ncoordinate.txt
-rw-r--r--   1 md1huihu cthstud     6326 Dec 31 01:30 Ocoordinate.txt
-rw-r--r--   1 md1huihu cthstud        4 Dec 31 01:30 Size.txt
-rw-r--r--   1 md1huihu cthstud        2 Jan 12 12:38 SumBlock.txt
-rw-r--r--   1 md1huihu cthstud      106 Jan 12 12:38 SumHbond.txt
-rw-r--r--   1 md1huihu cthstud        2 Jan 12 12:38 Sumpattern.txt
-rw-r--r--   1 md1huihu cthstud    69336 Dec 31 01:30 coordinates.txt
-rw-r--r--   1 md1huihu cthstud   646053 Dec 31 01:30 data.txt
-rw-r--r--   1 md1huihu cthstud     4984 Jan  5 10:42 getcoordinate.pl
-rw-r--r--   1 md1huihu cthstud       26 Jan 12 12:38 start.txt
```

Figure A.4: Contents of directory “Data”.

```
> ls -l
total 258
-rw-r--r--   1 md1huihu cthstud      405 Dec  3 22:14 TBCodesnew.txt
-rw-r--r--   1 md1huihu cthstud      336 Oct 28 15:35 VCodesnew.txt
-rw-r--r--   1 md1huihu cthstud      432 Nov 19 09:39 frameworkGridXY_TA.txt
-rw-r--r--   1 md1huihu cthstud      426 Nov 19 09:42 frameworkGridXY_TB.txt
-rw-r--r--   1 md1huihu cthstud      387 Dec  3 22:07 frameworkGridXY_TBnew.txt
-rw-r--r--   1 md1huihu cthstud      320 Nov 20 11:46 frameworkGridXY_V.txt
-rw-r--r--   1 md1huihu cthstud      388 Nov 19 09:15 frameworkGridXY_VH.txt
-rw-r--r--   1 md1huihu cthstud      363 Nov 19 09:10 frameworkGridXY_VL.txt
-rw-r--r--   1 md1huihu cthstud      329 Nov 19 09:33 frameworkGridXY_Vnew.txt
-rw-r--r--   1 md1huihu cthstud      438 Oct 25 15:10 frameworkPositions_TA.txt
-rw-r--r--   1 md1huihu cthstud      429 Oct 25 17:02 frameworkPositions_TB.txt
-rw-r--r--   1 md1huihu cthstud      395 Dec  3 22:19 frameworkPositions_TBnew.txt
-rw-r--r--   1 md1huihu cthstud      317 Oct 26 04:30 frameworkPositions_V.txt
-rw-r--r--   1 md1huihu cthstud      389 Oct 20 08:55 frameworkPositions_VH.txt
-rw-r--r--   1 md1huihu cthstud      367 Oct 20 16:09 frameworkPositions_VL.txt
-rw-r--r--   1 md1huihu cthstud      326 Oct 28 15:16 frameworkPositions_Vnew.txt
-rw-r--r--   1 md1huihu cthstud      439 Oct 25 16:36 hBonds_TA.txt
-rw-r--r--   1 md1huihu cthstud      415 Oct 25 17:29 hBonds_TB.txt
-rw-r--r--   1 md1huihu cthstud      502 Dec  3 22:20 hBonds_TBnew.txt
-rw-r--r--   1 md1huihu cthstud      407 Oct 26 05:03 hBonds_V.txt
-rw-r--r--   1 md1huihu cthstud      568 Oct 20 08:54 hBonds_VH.txt
-rw-r--r--   1 md1huihu cthstud      509 Oct 21 02:11 hBonds_VL.txt
-rw-r--r--   1 md1huihu cthstud      419 Oct 29 05:57 hBonds_Vnew.txt
-rw-r--r--   1 md1huihu cthstud    51269 Dec  8 11:26 temp.ps
-rwxr-xr-x   1 md1huihu cthstud    53264 Jan 11 19:06 v_seq.tcl
-rw-r--r--   1 md1huihu cthstud      699 Oct 20 08:15 vhCodes.txt
-rw-r--r--   1 md1huihu cthstud      627 Oct 21 03:01 vlCodes.txt
```

Figure A.5: Contents of directory “tcl”.

A.2 Description of files

getcoordintate.pl

Here, perl version 5.6 is used. This perl file is in the directory of Data to read the coordinates of main chain atom positions of C, O, N and C_α from corresponding PDB file. The PDB file is recorded into "Data/data.txt". All these atom coordinates are recorded into "Data/coordinates". C, O, N and C_α positions are recorded into "Data/Ccoordinate.txt", "Data/Ocoordinate.txt", "Data/Ncoordinate.txt" and "Data/CAcoordinate.txt", respectively. The position numbers and each residue are recorded into files "Data/Chainposition" and "Data/Chainresidue", respectively. The calculated length of the examined chain is recorded into file "Data/Size.txt".

patternsearch.h

This header file records all the functions included in this C program and defines all the constants used.

Size.c

Running this c file returns the length of the examined chain.

Hcoordinate.c

As no position of hydrogen bonded to main chain N is available from PDB file, this file describes the method used to calculate the position of hydrogen bonded to main chain atom N.

getHbond.c

This file calculates all the positions of hydrogen bonded to main chain N at first, then assign all the possible hydrogen bonds between main chain C=O and N-H groups and record these positions with assigned hydrogen bond into file "Data/Hbond.txt" and the total number of the hydrogen bonds into file "Data/HbondSize.txt".

HbondtempTAlpha.c, HbondtempTBeta.c, HbondtempVL.c, HbondtempVH.c, HbondtempV.c

Running these files records the hydrogen bond pattern of the corresponding template into file "Data/Hbondnotemp.txt", total number of hydrogen bonds between each pair of blocks of the template into "Data/SumHbond.txt", total number of patterns with hydrogen bonds inside into "Data/SumPattern.txt", total number of blocks of the template into "Data/SumBlock.txt", the starting point and the length of each block of the template into "Data/Blockstart.txt" and "Data/Blocklength.txt", respectively.

ReadPatterntemp.c

After running this file, all the hydrogen bond positions of the corresponding template are recorded into file "Data/Hbondtemplate.txt", and the total number of the hydrogen bonds in the template is recorded into "Data/HbondTempSize.txt".

ReadHbond.c

This file records the hydrogen bond pattern of the examined chain into file "Data/Hbondno.txt".

getHbondScope.c

This file initializes the search scope for each block in the examined chain.

getSum.c

Running this file gives us the total number of hydrogen bonds of the template before Pattern[i][j]. These include all the Patterns between main chain C=O group of Block[k] (when k<i) and main chain N-H group of either block, and the Patterns between main chain C=O group of Block[i] and main chain N-H group of Block[r] (when r<j).

SearchDiagonalPattern.c

Those hydrogen bonds within one block can be searched after running this file. At the same time, the search scope of other blocks is adjusted.

sortHbond.c

As for each Pattern[i][j] (when i!=j), hydrogen bonds exist in Pattern[j][i] only when there are some hydrogen bonds in Pattern[i][j], in order to increase the probability of getting one match for each Pattern[i][j], the total number of hydrogen bonds in one

pair of Patterns of the corresponding template are calculated and sorted using this program before searching these patterns.

SearchPattern.c

Those hydrogen bond patterns between two different blocks are searched by running this file after those within one block have been found.

printHbond.c

By running this file, the hydrogen bonds of the template and the corresponding hydrogen bonds in the examined chain are recorded into file "Data/CorrespondingHbond.txt", and the examined chain with different position indexes is recorded into file "Data/CorrespondingChain.txt".

main.c

This is the main C file and all other files are called by running this file.

Makefile

The file here aims to make the general configuration file. The executable file obtained here is stored in the directory of bin. Here the compiler GCC version 2.95 is used.

v_seq.tcl

This program aims to draw the schematic representations of different kinds of $\beta - sheet$ frameworks including VL, VH, TCR α, TCR β and immunoglobulin variable domain templates. Each time one figure is drawn and recorded into file "temp.ps".

A.3 Pseudo code for the critical algorithm (main.c)

```
count_pattern=0;
for (i=0;i<sum_block;i++)
{
        if (sumHbond[i][i]!=0){
                for (Hbond_less=0;Hbond_less<=n;Hbond_less++){
                        if (Pattern[i][i] is found){
                                break;
                        }
                        else {
                                for (j=0;j<Blocklength[i];j++){
                                        do search Pattern[i][i];
                                        if (Pattern[i][i] is found){
                                                adjust the search scope for other Block[k] when k!=i;
                                                count_pattern=count_pattern+1;
                                                break;
                                        }
                                }
                        }
                }
        }
}

do sort pairs of Pattern[i][j] and Pattern[j][i] when i!=j;
if (all Pattern[i][i] are found){
        for (c=0;c<count_pair;c++){
                for (Hbond_less1=0;Hbond_less1<=n;Hbond_less1++){
                        if (Pattern[i][j] and  Pattern[j][i] have been found){
                                break;}
                        else {
                                for (Hbond_less2=0;Hbond_less2<=m;Hbond_less2++){
                                        for (i=0;i<Blocklength[k];i++){
                                                if (Pattern[i][j] and  Pattern[j][i] have been found){
                                                        break;}
                                                else {
                                                        for (j=0;j<Blocklength[r];j++){
                                                                do search Pattern[k][r]
                                                                 and Pattern[r][k];
                                                                if (Pattern[i][j] and  Pattern[j][i]
                                                                 have been found){
                                                                        adjust the search scope
                                                                        of other blocks;
                                                                        count_pattern=count_pattern+2;
                                                                        break;
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
}
if (all the Patterns are found){
The examined chain matches the template;}
else {The examined chain does not match the template;}
```

Appendix B

User Manual

B.1 Using the programs

Before running this program, it is better for the user to have the knowledge of the structure of immunoglobulin variable domains such as the structure of natural and artificial VL, VH, T cell receptor α and β variable domains. It is also required to understand the format of PDB files in order to run this program. This program can be used under Unix/Linux. The executable file "search" is in the directory "bin". Before running "search", "getcoordinate.pl" should be run using the command "runperl.sh" at first to get all the coordinates needed here.

```
>runperl.sh
Please input PDB ID:12e8
Please input chain name, if chain name is NULL, please press 'Enter'! L
Examined chain is L!
```

Figure B.1 Typescript of the getcoordinate.pl program using command runperl.sh.

After running "getcoordinate.pl", the coordinates of main chain C, O, N and C_α are recorded into "Data/Ccoordinate.txt", "Data/Ocoordinate.txt", "Data/Ncoordinate.txt" and "Data/CAcoordinate.txt", respectively. The calculated length of the examined chain is recorded into file "Data/Size.txt". Then compile the C files using command "install.sh" and record the executable file "search" in the directory of bin if something changed in these files (Figure B.2).

Finally compare the hydrogen bond pattern of the examined chain to that of the template using command "bin/search". In this case, you need to choose the corresponding template by input 1, 2, 3, 4 or 5 at first, and then input the starting point to search. Figure B.3 shows the typescript of the running program search. It lists the hydrogen bonds in the template and the corresponding hydrogen bonds of the examined chain. "0" or "1" in the last column shows whether a hydrogen bond is assigned at the corresponding position in the examined chain or not.

```
> install.sh
gcc -c src/main.c
gcc -c src/size.c
gcc -c src/Hcoordinate.c
gcc -c src/getHbond.c
gcc -c src/ReadHbond.c
gcc -c src/HbondtempVH.c
gcc -c src/HbondtempVL.c
gcc -c src/HbondtempTAlpha.c
gcc -c src/HbondtempTBeta.c
gcc -c src/HbondtempV.c
gcc -c src/ReadPatterntemp.c
gcc -c src/getSum.c
gcc -c src/sortHbond.c
gcc -c src/getHbondScope.c
gcc -c src/searchDiagonalPattern.c
gcc -c src/searchPattern.c
gcc -c src/printHbond.c
gcc  -o    search  -lm  main.o  size.o  Hcoordinate.o  getHbond.o  ReadHbond.o
HbondtempVH.o  HbondtempVL.o  HbondtempTAlpha.o  HbondtempTBeta.o  HbondtempV.o
ReadPatterntemp.o getSum.o sortHbond.o getHbondScope.o searchDiagonalPattern.o
searchPattern.o printHbond.o
rm search *.o
```

Figure B.2 Typescript of running install.sh.

```
> bin/search

This program aims to check whether the examined chain
belongs to VL or VH domain, T cell receptor alpha or beta domain;

1: compared to VL template!
2: compared to VH template!
3: compared to T cell receptor Alpha domain template!
4: compared to T cell receptor Beta domain template!
5: compared to Variable domain template!

Pleae input your choice now: 1
choice:  1
Now compare the examined chain to template: VL

Please input starting point now!

NOTE: Please input starting point as 0 if examined chain is not scFv!
If examined chain is scFv, please check N terminal at first!
The starting point to search the domain after the N terminal domain
is the next residue after the ending point of the N terminal domain!

start: 0
Starting point is 0!

template  examined chain
Pattern[4][4]
(33  50)  (33  50)  0
(35  47)  (35  47)  1
(35  48)  (35  48)  1
(37  45)  (37  45)  1
(39  42)  (39  42)  1
(45  37)  (45  37)  1
(47  55)  (47  55)  1
(48  35)  (48  35)  1
```

Figure B.3 Typescript of running the search program using command bin/search.

```
(49  53)  (49  53)  1
(53  49)  (53  49)  1
Pattern[5][5]
(61  76)  (61  76)  1
(63  74)  (63  74)  1
(65  72)  (65  72)  1
(67  70)  (67  70)  1
(70  67)  (70  67)  1
(72  65)  (72  65)  1
(74  63)  (74  63)  1
Pattern[3][5]
(19  75)  (19  75)  1
(21  73)  (21  73)  1
(23  71)  (23  71)  1
Pattern[5][3]
(71  23)  (71  23)  1
(73  21)  (73  21)  1
(75  19)  (75  19)  1
Pattern[4][6]
(34  89)  (34  89)  1
(36  87)  (36  87)  1
(38  85)  (38  85)  1
Pattern[6][4]
(85  38)  (85  38)  1
(87  36)  (87  36)  1
(89  34)  (89  34)  1
Pattern[6][7]
(84  104)  (84  104)  1
(86  102)  (86  102)  1
(88  99)  (88  99)  1
(90  97)  (90  97)  1
Pattern[7][6]
(97  90)  (97  90)  1
(102  86)  (102  86)  1
Pattern[2][7]
(9  103)  (9  103)  1
(11  105)  (11  105)  1
(13  107)  (13  107)  1
Pattern[7][2]
(103  11)  (103  11)  1
(105  13)  (105  13)  1
Pattern[1][3]
(5  24)  (5  24)  1
Pattern[3][1]
(24  5)  (24  5)  1
count is 12, sum is 12
This is VL domain!
This VL domain terminates at 106th residue!
```

Figure B.3 Typescript of runnning the search program using command bin/search (continued).

All these variable domain templates can be drawn one by one using command "tcl/v_seq.tcl". Figure B.4 shows the new screen shown after using this command. By pressing button "Template", a list of choices is shown (Figure B.5).

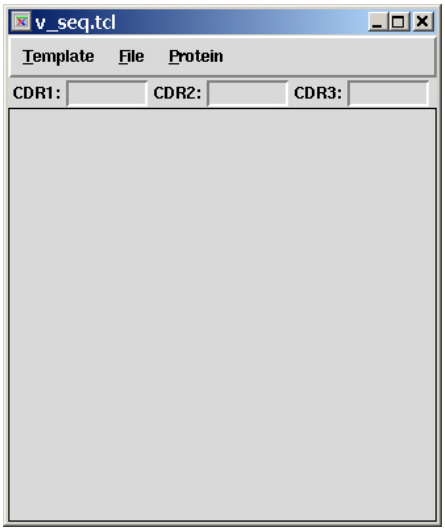


Figure B.4: New screen shown after using command "tcl/v_seq.tcl".

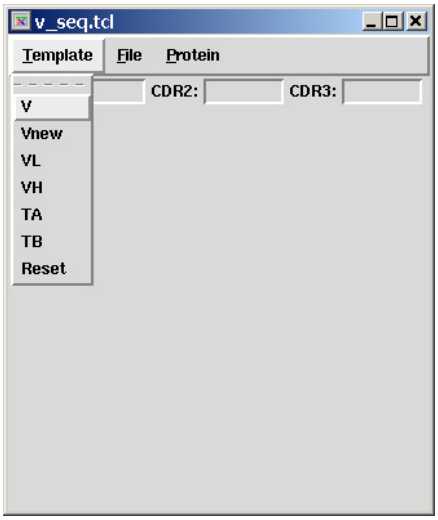


Figure B.5: List of choices shown in the new screen.

In order to draw one template, we only need to press the corresponding choice from this list. Figure B.6 shows the schematic representation of immunoglobulin variable domain template by pressing button "V". By pressing button "File" and choose "PostScript", this figure is saved to file "temp.ps". In order to draw another picture, "Reset" should be pressed before making another choice under "Template".

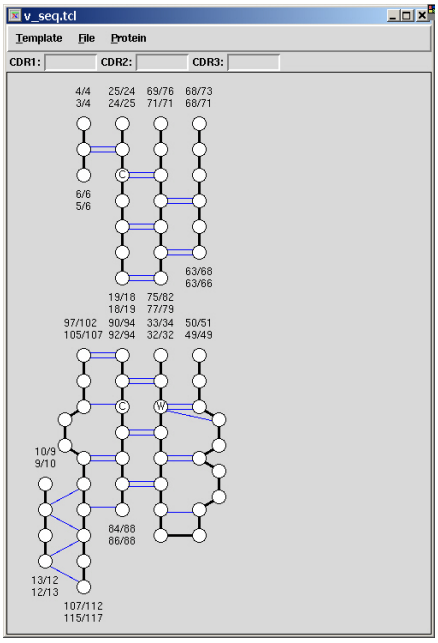


Figure B.6: One example of schematic representation of the template.

B.2 Format of data files

All these data files produced by running perl and C files are in the directory of Data.

Blocklength.txt

This one row numbers lists the length of each block consecutively in the template.

Blockstart.txt

This one row numbers lists the position of the starting residue of each block consecutively in the template.

Ccoordinate.txt, Ocoordinate.txt, Ncoordinate.txt, CAcoordinate.txt

These files records the positions of corresponding atoms with the X coordinates in the first column, the Y coordinates in the second column and the Z coordinates in the third column.

Chainposition.txt

This file records the position index from the corresponding PDB file in one column.

Chainresidue.txt

This file records the name of each residue from the corresponding PDB file in one column.

coordinates.txt

This file records all the positions of main chain C, N, O and C_α. Each line begins with "ATOM", the second column is the series number, the third column is the name of the atom, the fourth column is the name of the residue it belongs to, the fifth column is the name of the chain, the sixth is the residue number, followed by three dimensional coordinates in the seventh, eighth and ninth columns.

CorrespondingChain.txt

The upper part of the file records the starting point of each block. The lower part records the residue positions using different numbering schemes including the one used in the corresponding PDB file, Chothia Numbering Scheme and the New code.

CorrespondingHbond.txt

This file records the positions of hydrogen bonds in the template in the first column, the corresponding positions in the examined chain in the second column, followed by the third column with lists whether the hydrogen bond is assigned or not in the corresponding positions by using "0" or "1".

Data.txt

This file only copies the corresponding PDB file.

Hbond.txt

In this file, the first column records the position of the residues which provide main chain C=O groups to form hydrogen bonds; while the second column records the position of the residues which provide main chain N-H groups to form hydrogen bonds with the corresponding C=O groups recorded in the first column.

Hbondnotemp.txt

Here “0” and “1” are used to represent the hydrogen bond pattern of the template, where “0” represents no hydrogen bond is assigned in the template, while “1” represents hydrogen bond is assigned.

Hbondno.txt

Here “0” and “1” are used to represent the hydrogen bond pattern of the examined chain.

HbondSize.txt

The number here represents the total number of hydrogen bonds assigned in the examined chain.

Hbondtemplate.txt

This file records the positions of each pair of residues where hydrogen bond is assigned in the template the same way as in the file “Hbond.txt”. The only difference is this file records the positions one pattern after another.

HbondTempSize.txt

The one number here means the total number of hydrogen bonds in the template.

Size.txt

The one number here represents the length of the examined chain.

start.txt

This row of numbers lists the corresponding starting positions of each block in the examined chain.

SumBlock.txt

This number lists the total number of blocks divided in the template.

SumHbond.txt

The matrix here lists the total number of hydrogen bonds between each pair of blocks of the template.

SumPattern.txt

The number here lists the total number of patterns with hydrogen bonds inside in the template.

All the txt files needed to run tcl file "v_seq.tcl" are in the directory of "tcl".

frameworkGridXY_TA.txt, frameworkGridXY_TB.txt, frameworkGridXY_TBnew.txt, frameworkGridXY_V.txt, frameworkGridXY_VH.txt, frameworkGridXY_VL.txt, frameworkGridXY_Vnew.txt

These files record two dimensional coordinates of each residue in the conserved framework of the corresponding templates consecutively from the N-terminal to the C-terminal. The numbers in the first column are the X coordinates of the corresponding residues, while those in the second column are the Y coordinates of the corresponding residues.

frameworkPositions_TA.txt, frameworkPositions_TB.txt, frameworkPositions_TBnew.txt, frameworkPositions_V.txt, frameworkPositions_VH.txt, frameworkPositions_VL.txt, frameworkPositions_Vnew.txt

The codes in these files are the position codes of the conserved frameworks using the Chothia Numbering Scheme except for those codes in files "frameworkPositions_V.txt" and "frameworkPositions_Vnew.txt", where position codes of strand i begins with 100*i.

hBonds_TA.txt, hBonds_TB.txt, hBonds_TBnew.txt, hBonds_V.txt, hBonds_VH.txt, hBonds_VL.txt, hBonds_Vnew.txt

Each pair of codes in one parenthesis in these files refer to one hydrogen bond formed between the main chain N-H group of the first residue and the main chain C=O group of the second one. The positions codes here are also codes using the Chothia Numbering Scheme except for those codes in files "hBonds_V.txt" and "hBonds_Vnew.txt", where position codes of strand i begins with 100*i.

TBCodesnew.txt, VCodesnew.txt, VhCodes.txt, VlCodes.txt

These files contain both the position codes within the conserved framework and part or all of the position codes outside of the conserved framework.