UNIVERDIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS
MESTRADO EM INFORMÁTICA

PREDIÇÃO DE ESTRUTURA DE PROTEÍNA POR HOMOLOGIA ASSISTIDA POR ONTOLOGIA

KELLEN FABIANE DA SILVA PINAGÉ

MANAUS
2007

UNIVERDIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS
MESTRADO EM INFORMÁTICA

KELLEN FABIANE DA SILVA PINAGÉ

PREDIÇÃO DE ESTRUTURA DE PROTEÍNA POR HOMOLOGIA ASSISTIDA POR ONTOLOGIA

Dissertação apresentada ao Mestrado em Informática da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Mestre em Informática, área de concentração Inteligência Artificial

Orientadora: Prof^a Dr^a Virgínia Verônica Bezerra Brilhante

MANAUS
2007

Agradecimentos

A meus pais Frederico Pinagé e Neide Pinagé por sempre me incentivarem nos meus estudos e me apoiarem nas decisões de minha vida;

Aos meus irmãos Gizele Pinagé e Frederico Pinagé pela amizade e companheirismo durante todos esses anos;

Aos meus amigos que sempre estiveram ao meu lado nos momentos de dificuldade e nos momentos de conquista em minha vida;

A minha orientadora Virgínia Brilhante pelo apoio, confiança, paciência e coragem por embarcar comigo neste trabalho interdisciplinar de muitos desafios;

A UFAM por proporcionar esse oportunidade de crescimento em pesquisa a nós amazonenses;

Ao graduando da UFAM Júlio Silva pela ajuda na etapa de desenvolvimento;

Ao pesquisador Amandeep S. Sidhu, da Universidade de Tecnologia de Sydney, Austrália, pelas informações sobre a *Protein Ontology*;

Ao pesquisador Björn Wallner pelo fornecimento da amostra de alinhamentos usada na etapa de testes;

Ao meu chefe e amigo Ernande Melo por ser compreensivo e entender das minhas necessidades quanto a realização do Mestrado;

A Fucapi, por me permitir participar deste Mestrado, me fornecendo parte do meu expediente para realizar meus estudos e crescer com mais esta realização em minha vida.

Resumo

Predição de estrutura de proteína é um processo na Biologia Molecular pelo qual a estrutura 3D de uma proteína é determinada com base em estruturas já conhecidas de outras proteínas. É um processo importante porque a estrutura de uma proteína é um fator determinante para sua função na célula. Conhecendo a estrutura de uma proteína, os cientistas podem descobrir que tipo de atividade a proteína realiza na célula e criar drogas para combater doenças. O processo de predição é baseado na similaridade entre as seqüências de aminoácidos que formam proteínas: a estrutura de uma proteína alvo é predita reusando conhecimento de proteínas cujas estruturas já foram determinadas e suas seqüências de aminoácidos são similares à seqüência alvo. Portanto, reuso de conhecimento ocorre no processo de predição de estrutura de proteína, mas sem a utilização de uma ontologia de domínio. Neste trabalho, nós aplicamos uma técnica de reuso de conhecimento baseado em ontologia na predição de estrutura de proteína com o objetivo de melhorar o processo de predição em sua eficiência e qualidade das estruturas obtidas. Um experimento foi realizado no qual a técnica foi aplicada para predizer a estrutura de 286 seqüências alvo. Houve melhorias e perdas de qualidade das estruturas preditas, ao passo que um ganho de performance (tempo de execução) foi observado em 38% das seqüências alvo.

Abstract

Protein structure prediction is a process in Molecular Biology by way of which the 3D structure of a protein is determined based on known structures of other proteins. This is an important process because the structure of a protein is a determinant factor for its function in the cell. Knowing a protein's structure allows scientists to describe the kind of activity that the protein performs in the cell and to develop drugs to treat diseases. The prediction process is based on similarity between the amino acid sequences that form proteins: the structure of a target protein is predicted by re-using knowledge on proteins whose structures are already determined and whose amino acid sequences are similar to the former's. Therefore, knowledge re-use occurs in the process of predicting protein structure, but without employing a domain ontology. In this work, we apply a technique of ontology-driven knowledge re-use in protein structure prediction aiming at improving the prediction process in its efficiency and in the quality of the obtained structures. An experiment has been carried out in which the technique was applied to predict the structure of 286 target sequences. There has been improvement as well as loss of quality of predicted structures, whereas a run time performance gain in 38% of the target structures was observed.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

As proteínas são moléculas formadas pela ligação de aminoácidos e têm funções específicas num organismo. Em seu meio natural essas cadeias se dobram de diversas formas apresentando estruturas tridimensionais. Estas estruturas tridimensionais estão relacionadas com a função das moléculas na célula, pois é na conformação tridimensional que a proteína é capaz de desempenhar sua função biológica. Portanto, a determinação da estrutura tridimensional de uma proteína é fundamental para conhecer a função que a mesma desempenha na célula.

Para determinar a estrutura tridimensional de uma proteína existe um método chamado Predição de Estrutura Terciária ou Tridimensional. Esta predição é feita a partir da conformação primária da proteína, ou seja, a seqüência de aminoácidos. A predição é feita dessa forma porque os cientistas descobriram que conhecendo a seqüência de aminoácidos que compõem uma proteína é possível prever sua conformação tridimensional [10].

Entre os métodos existentes para a predição, existem os realizados por computador, como o método de Modelagem Comparativa de Estrutura de Proteína (ou Predição por Homologia). Neste método podemos verificar a aplicação de reuso de conhecimento, onde bases de informações sobre estruturas de proteínas já conhecidas são utilizadas na predição de estruturas de proteínas desconhecidas. Com isso podemos perceber que a Predição por Homologia pode ser abordada a partir da perspectiva da Engenharia de Conhecimento.

Em [3] e [4] é proposta uma técnica de reuso de conhecimento baseado em ontologia, onde modelos existentes são reusados por um mecanismo de modelagem

na geração de novos modelos utilizando como suporte dados descritos por uma ontologia do domínio da Ecologia, a Ecolingua [5]. Genericamente, dados são descritos através da ontologia e um processo de solução de problemas baseado em conhecimento utiliza soluções existentes, associadas aos conceitos na ontologia, para gerar novas soluções. Neste trabalho é mostrado que a utilização da ontologia para o reuso do conhecimento possibilita um ganho na eficiência da modelagem.

Temos como objetivo neste trabalho desenvolver a aplicação desta técnica no domínio da Biologia Molecular, mais especificamente no processo de Predição de Estrutura de Proteína pelo método de Predição por Homologia, que já aplica o reuso de conhecimento mas sem aplicação de uma ontologia [11]. Outro objetivo deste trabalho é tornar explícito o conhecimento sobre o processo de predição de estrutura de proteína por homologia.

Além de um melhor desempenho no tempo de predição, verificamos também ganho de qualidade em algumas estruturas geradas pelo processo de predição, com o reuso de conhecimento baseado em ontologia. Porém, algumas estruturas tiveram perda de qualidade e, desta forma, é necessária uma análise detalhada sobre a qualidade das estruturas geradas.

Esta dissertação está organizada da seguinte forma: no Capítulo 2 é mostrada a predição de estrutura de proteína como um problema de engenharia de conhecimento e são colocados alguns conceitos da biologia molecular necessários ao entendimento do trabalho, além de informações sobre a ferramenta de predição adotada neste trabalho. No Capítulo 3 é descrita a ontologia selecionada para aplicar reuso de conhecimento baseado em ontologia na predição de estrutura de proteína. No Capítulo 4 é descrita a técnica de reuso de conhecimento baseado em ontologia no domínio da Ecologia e é mostrada a adaptação da técnica para aplicar o reuso de conhecimento baseado em ontologia na predição de estrutura de proteína. No Capítulo 5 é feita a avaliação dos resultados obtidos e no Capítulo 6 temos a discussão, conclusões e trabalhos futuros. A revisão bibliográfica e trabalhos relacionados serão tratados ao longo do texto.

Capítulo 2

Predição de estrutura de proteína como um problema de engenharia de conhecimento

Engenharia de conhecimento é a área responsável pela aquisição do conhecimento tácito do especialista (coleta, seleção, decomposição, composição e modelagem) e sua integração com o conhecimento existente em bases de dados relacionadas ao escopo deste especialista. Seu objetivo final é a criação de sistemas inteligentes que auxiliem e/ou substituam especialistas humanos em suas tarefas, sistemas estes capazes de executar uma difícil tarefa adequadamente.

A engenharia do conhecimento descreve o processo global de desenvolvimento destes sistemas especialistas, baseados em conhecimento. Uma das atividades deste processo e a aquisição do conhecimento e sua meta é obter conhecimento detalhado utilizado pelo especialista para solucionar problemas, e então transformar e transferir essa informação para um sistema de computador.

A predição de estrutura de proteína é um problema de engenharia de conhecimento, e sistemas baseados em conhecimento são construídos para realizar a complexa tarefa de predizer a estrutura tridimensional de uma proteína a partir de sua estrutura primária, que corresponde à seqüência de aminoácidos que compõem a proteína. Para realizar essa predição o processo utiliza o conhecimento prévio de estruturas de proteínas já conhecidas, que constituem uma base de dados para auxiliar na descoberta de estruturas de proteínas desconhecidas, e o conhecimento de

especialistas sobre a formação das estruturas de proteínas. Todo esse conhecimento sobre estruturas de proteínas adquirido por especialistas em laboratórios: de como é formada a estrutura, que variáveis interferem em sua forma final, são reunidos e sistemas são criados para realizar o processo de predição.

Como podemos ver, a predição de estrutura de proteína é realizada baseada no conhecimento da estrutura primária da proteína, conhecimento das estruturas de outras proteínas similares a ela e o conhecimento reunido de especialistas. Desta forma, no problema de predição de estrutura de proteína, assim como em outros problemas de engenharia de conhecimento, o conhecimento é reunido, transformado e transferido para sistemas baseados em conhecimento.

Com a criação dos sistemas especialistas o conhecimento tácito dos especialistas sobre as estruturas de proteínas é tornado explícito e o processo de predição da proteína é formalizado baseado nesse conhecimento. Os sistemas de computador existentes hoje para realizar o processo de predição de estrutura de proteína formalizaram esse processo e tornaram explícito o conhecimento tácito dos especialistas em relação as estruturas de proteínas. Neste trabalho a reengenharia de um sistemas de predição de estrutura de proteína, com o objetivo de melhorar seu desempenho, torna o conhecimento sobre o processo de predição explítico à comunidade.

Entre as técnicas de IA usadas em sistemas de Engenharia de Conhecimento optamos por utilizar ontologias. Essa escolha se dá devido [3] ter mostrado que a utilização da ontologia para o reuso de conhecimento possibilita um ganho na eficiência de modelagem, sendo essa demonstração feita no processo de geração de estruturas de modelos ecológicos. Como no problema de predição de estrutura de proteína são gerados modelos, porém de estruturas de proteínas, nosso principal objetivo é desenvolver a aplicação da técnica usada em [3] no domínio da Biologia Molecular, no processo de predição de proteína, para obter esse ganho de eficiência.

Nas Seções seguintes deste Capítulo veremos informações relevantes para o entendimento do processo de predição de estrutura de proteína.

2.1 Proteínas e suas estruturas

As proteínas são macromoléculas formadas pela ligação de várias unidades semelhantes, chamadas aminoácidos e têm funções específicas dentro de um organismo. Elas podem ter caráter estrutural, usadas para confeccionar paredes celulares, cabelo, unhas e diversos tecidos. Podem também estar ligadas a determinadas atividades dentro do organismo, como é o caso dos anticorpos, de hormônios e das enzimas [6, 10].

Em seu meio natural, essas cadeias de aminoácidos se dobram de diversas formas apresentando estruturas tridimensionais. Tais estruturas estão intimamente relacionadas à função das moléculas e, portanto, sua determinação é parte fundamental do estudo das proteínas. Por isso a importância do processo de predição de estrutura de proteína no domínio da biologia molecular.

Cada tipo de proteína difere em sua seqüência e número de aminoácidos. Existem 20 aminoácidos que são comumente encontrados nas proteínas e cada um deles possui uma estrutura similar um ao outro, porém única. A Figura 2.1 mostra a estrutura básica de um aminoácido, onde temos um grupo amino, um grupo carboxila e um grupo R (ou cadeia lateral) que diferencia cada aminoácido. O carbono onde o grupo R está ligado é chamada carbono alpha (Cα).

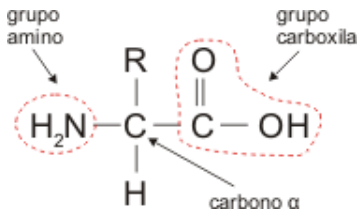

Figura 2.1: Aminoácido

Os 20 aminoácidos existentes estão listados na Tabela 2.1.

Cada aminoácido pertencente a uma proteína é conectado a um outro através de uma ligação entre o grupo carboxila de um aminoácido e o grupo amino do aminoácido seguinte, como mostram as Figuras 2.2 e 2.3. Na Figura 2.3 podemos ver também que o peptídeo formado possui, em um resíduo da cadeia, um grupo amino livre, sendo denominado resíduo N-terminal e, em outro resíduo da cadeia, um grupo carboxila livre, sendo denominado resíduo C-terminal.

Nome	Abreviatura	Letra(identificadora)
Alanina	Ala	A
Cisteína	Cys	C
Ácido aspártico	Asp	D
Ácido glutâmico	Glu	E
Fenilalanina	Phe	F
Glicina	Gly	G
Histidina	His	H
Isoleucina	Ile	I
Lisina	Lys	K
Leucina	Leu	L
Metionina	Met	M
Asparagina	Asn	N
Prolina	Pro	P
Glutamina	Gln	Q
Arginina	Arg	R
Serina	Ser	S
Treonina	Thr	T
Valina	Val	V
Triptofano	Trp	W
Tirosina	Tyr	Y

Tabela 2.1: Os 20 aminoácidos encontrados na natureza, com a sigla e letra identificadora

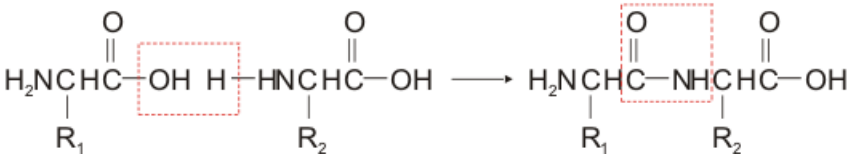

Figura 2.2: Cadeia polipeptídica - ligação entre grupo carboxila e grupo amino

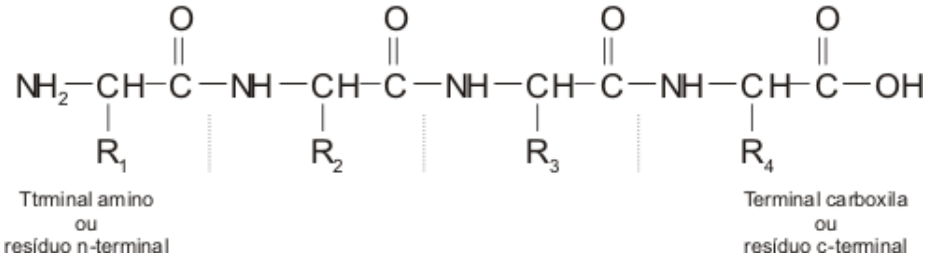

Figura 2.3: Cadeia polipeptídica - grupo amino livre e grupo carboxila livre

A proteína consiste então de um *backbone* peptídico com cadeias laterais anexadas, como mostra a Figura 2.4.

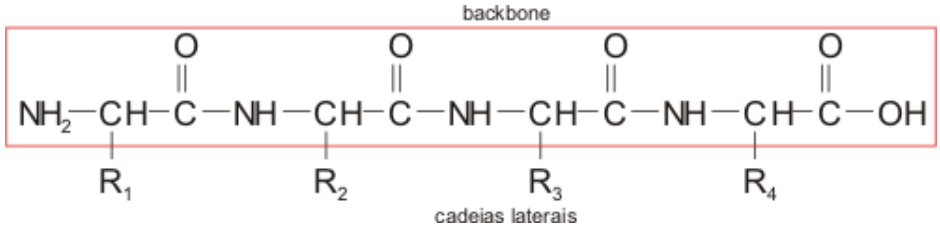

Figura 2.4: Backbone e cadeia lateral

Essas ligações entre os aminoácidos, chamadas de ligações peptídicas formam a estrutura primária da proteína, que corresponde a sua seqüência de aminoácidos. Então, a estrutura primária nada mais é do que a seqüência de aminoácidos constituintes da proteína. Os aminoácidos quando ligados entre si, formando a estrutura primária da proteína, são também chamados de resíduos.

A estrutura secundária de uma proteína descreve a forma tridimensional de segmentos locais de uma proteína. Entretanto, não descreve posições atômicas no espaço tridimensional. Essa descrição de posições atômicas no espaço tridimensional é considerada a estrutura terciária da proteína. A estrutura secundária consiste de interações locais entre resíduos mediados por pontes de hidrogênio. Pode-se dizer que a estrutura secundária resulta da interação da estrutura primária com ela mesma, através da formação de pontes de hidrogênio. As estruturas secundárias mais comuns

são *alpha (α) helix* e *beta (β) sheet*.

Uma estrutura secundária *alpha helix* está organizada no formato de um espiral, similar ao formato de um saca-rolha, como mostra a Figura 2.5. Uma *alpha helix* é formada por pontes de hidrogênio entre o oxigênio carboxila de um aminoácido e o nitrogênio do *backbone* de um segundo aminoácido localizado quatro posições adiante.

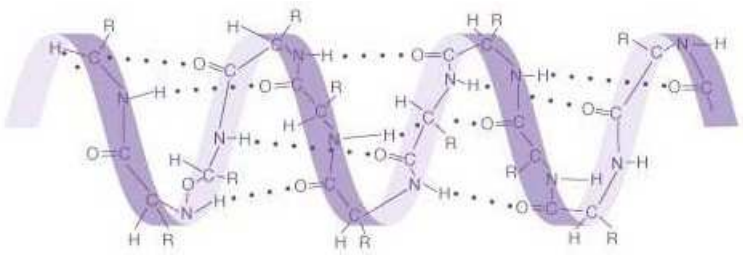

Figura 2.5: Forma estrutural de uma *Alpha Helix*

Uma estrutura secundária *beta sheet* é organizada em zig-zag. No mínimo dois *strands* são necessários para definir um *beta sheet*. Cada *strand* é uma extensão contínua de aminoácidos que se ligam por pontes de hidrogênio formando o *beta sheet*. Isso pode ser visto na Figura 2.6.

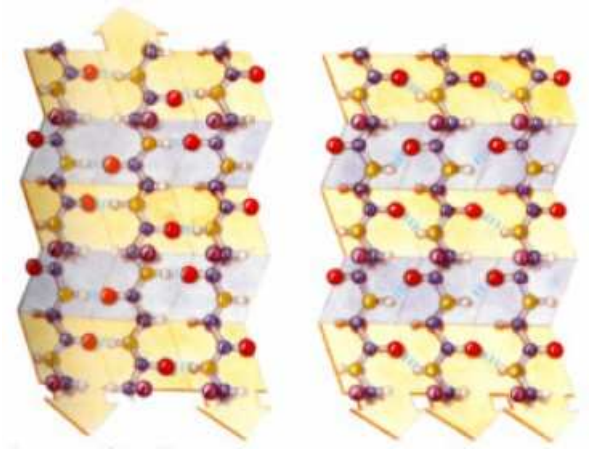

Figura 2.6: Forma estrutural de uma *Beta Sheet*. Estrutura anti-paralela e estrutura paralela, nesta ordem

Uma *beta sheet* é formada por pontes de hidrogênio entre o oxigênio carboxila de uma aminoácido em um *strand* e o nitrogênio do *backbone* de um segundo aminoácido em outro *strand*. As *beta sheets* podem ser paralelas ou anti-paralelas. Se o resíduo N-terminal de cada *strand* "apontar" na mesma direção a estrutura

é considerada paralela. Caso contrário, se o resíduo N-terminal de cada strand "apontar" para direções opostas a estrutura é considerada anti-paralela.

Existe um código chamado DSSP que é usado freqüentemente para descrever as estruturas secundárias de uma proteína. Nesse código as estrutura secundárias são descritas com uma única letra, onde a letra H simboliza uma *alpha helix* e a letra E simboliza uma *beta sheet* em uma conformação paralela ou anti-paralela.

A estrutura terciária de uma proteína é sua forma total. Refere-se à conformação espacial da proteína como um todo e não de determinados segmentos particulares da cadeia protéica, como a estrutura secundária. A estrutura terciária determina as posições atômicas no espaço tridimensional de toda a proteína. Como a estrutura terciária corresponde a forma total, ela engloba as ligações peptídicas, formadoras da estrutura primária, e as ligações de pontes de hidrogênio, formadoras da estrutura secundária. Os conceitos ontológicos utilizados neste trabalho são referentes a estrutura secundária, que faz parte da estrutura final, terciária, da proteína.

Certas proteínas são formadas por mais de uma cadeia peptídica. Neste caso, a especificação de como estas sub-unidades protéicas são ligadas entre si para a formação da proteína é a sua estrutura quaternária. A Figura 2.7 resume todas as estruturas de uma proteína.

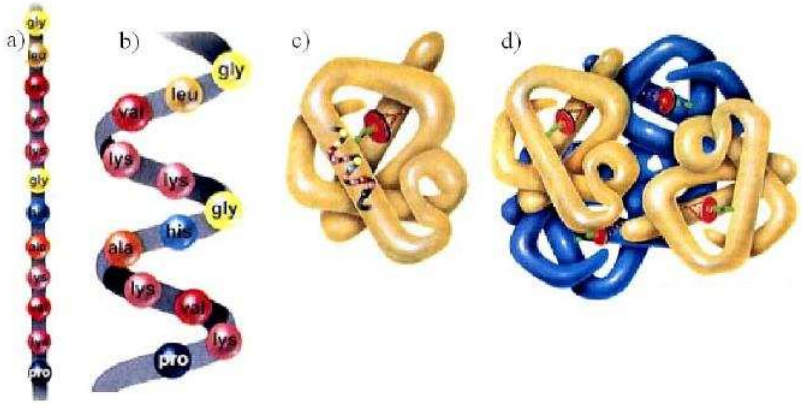

Figura 2.7: (a) Estrutura primária, (b) estrutura secundária, (c) estrutura terciária, (d) estrutura quaternária.

Os biólogos afirmam que conhecendo a seqüência de aminoácidos que compõem a proteína é possível prever sua conformação tridimensional, pois experiências em

laboratório mostram que ao "esticar" *in vitro* uma proteína e depois "soltá-la", em pouco tempo essa proteína assume novamente sua conformação tridimensional, sem interferência de qualquer outra molécula [1]. Essa conformação tridimensional de uma proteína pode ser obtida por métodos experimentais, métodos de predição por homologia e pelo método *ab-initio*.

Os métodos experimentais de cristalografia de raio-X e espectroscopia por RNM (ressonância nuclear magnética) possuem um custo elevado e constituem um processo muito trabalhoso. Além disso, muitas estruturas são difíceis ou impossíveis de serem determinadas por raio-X ou por RNM.

Os métodos de predição por homologia são realizados utilizando estruturas de proteínas já conhecidas. Por isso o motivo de selecionarmos este tipo de método para o trabalho em questão, pois é um processo que implementa a idéia de reuso de conhecimento, onde estruturas de proteínas já conhecidas são utilizadas na predição de estruturas de outras proteínas.

2.2 Método de predição por homologia

O objetivo do método de predição (modelagem) por homologia é construir um modelo tridimensional de uma proteína de estrutura desconhecida, denominada seqüência alvo, baseado na similaridade de seqüência com proteínas de estruturas já conhecidas, denominadas moldes.

Este tipo de predição é possível porque uma pequena mudança na seqüência resulta em uma também pequena mudança na estrutura tridimensional [9] e proteínas relacionadas de forma evolutiva (proteínas homólogas), pertencentes a uma mesma família de proteína, compartilham estruturas similares. A construção de um modelo bem sucedido requer ao menos uma estrutura tridimensional molde encontrada de forma experimental que tenha uma significativa similaridade com a seqüência alvo [2, 8].

O método de predição por homologia envolve basicamente quatro passos: 1. identificação e seleção de moldes; 2. alinhamento da seqüência alvo com a(s) seqüência(s) molde(s); 3. construção de um modelo da proteína alvo, baseado na informação do alinhamento; 4. avaliação do modelo. Todos estes passos podem ser

repetidos até que um modelo satisfatório seja obtido [9].

Entre os métodos que podem ser usados no passo 3 para construção de um modelo tridimensional da proteína alvo temos modelagem por montagem de corpos-rígidos, modelagem por casamento de segmento e modelagem por satisfação de restrições. O método de modelagem por montagem de corpos-rígidos é um dos mais utilizados [9], onde um modelo é montado a partir de um pequeno número de corpos rígidos obtidos de estruturas de proteínas alinhadas.

Na Seção seguinte veremos uma ferramenta de predição por homologia, denominada Nest, a qual foi selecionada, dentre as ferramentas existentes, para aplicar reuso de conhecimento baseado em ontologia.

2.2.1 Nest - ferramenta de predição por homologia

A Nest, parte central do pacote JACKAL, é uma ferramenta de predição por homologia de estruturas tridimensionais de proteínas, que implementa a modelagem por montagem de corpos-rígidos e é considera uma das melhores ferramentas de modelagem por homologia [18]. O JACKAL é uma pacote de modelagem de estrutura de proteína, que fornece ferramentas eficientes para construção, refinamento, manipulação e reconstrução de estruturas de proteína.

A ferramenta Nest foi a ferramenta de predição selecionada para aplicar reuso de conhecimento baseado em ontologia por ser considerada uma das melhores ferramentas de modelagem por homologia e por ser, dentre as melhores indicadas em [18], a única ferramenta de código aberto, nos permitindo ter acesso ao seu código fonte para modificá-lo de forma a aplicar o reuso de conhecimento baseado em ontologia.

A utilização da ferramenta Nest é feita através de linha de comando e foi desenvolvida em linguagem C++, funcionando apenas em ambiente Linux. Como o resultado final de seu processamento gera um arquivo no formato PDB (*Protein Data Bank*), contendo o modelo final, é possível visualizar graficamente a estrutura resultante em qualquer ferramenta gráfica que lê arquivo PDB, como a RasMol [24] e a DeepView (Swiss-PdbViewer) [21].

De acordo com os quatro passos básicos da predição por homologia, citados na Seção 2.2, a ferramenta Nest implementa os passos 3 e 4. Os passos 1 e 2 devem ser realizados por uma ferramenta de alinhamento e a Nest recebe então como entrada

um arquivo contendo o alinhamento resultante. Este arquivo de entrada deve conter o alinhamento entre a seqüência alvo e as estruturas moldes. Além do arquivo com o alinhamento, a Nest utiliza também arquivos PDB contendo as informações de estrutura dos moldes selecionados no processo de alinhamento.

O modelo constituído pela ferramenta Nest é baseado em um método de evolução artificial, onde dado um alinhamento entre a seqüência alvo e os moldes, este alinhamento pode ser visto como uma lista de operações de troca, inserção e deleção de resíduos. A construção do modelo de estrutura da seqüência alvo, baseado nos moldes, é então um processo de executar esta lista de operações de troca, inserção e deleção de resíduos [19], como um processo de evolução artificial desses resíduos para se chegar na estrutura da proteína alvo que seja a mais próxima de sua estrutura nativa.

Cada operação realizada pela Nest no processo de construção do modelo é finalizada com uma minimização de energia do modelo gerado. O motivo para isto é que a estrutura de uma proteína em seu estado natural possui a menor energia e é nesse estado que ela realiza suas funções dentro da célula.

Como no alinhamento a seqüência alvo pode ter várias regiões correspondendo a regiões de moldes diferentes, a ferramenta Nest, neste caso, constrói estruturas compostas, onde o modelo é construído da fusão desses diferentes moldes, gerando um modelo composto.

No caso da seqüência alvo ter mais de um molde homólogo a ela, a ferramenta Nest constrói modelos baseados em cada um dos moldes homólogos, construindo desta forma múltiplos modelos. Esses modelos são então sobrepostos e as regiões de alta variabilidade são identificadas. Para essas regiões, a ferramenta Nest tenta todas as possibilidades de conformação dos moldes e procura identificar a conformação com a menor energia.

Na Seção seguinte são mostrados os possíveis formatos do arquivo de entrada contendo o alinhamento, utilizado pela ferramenta Nest no processo de predição.

2.2.1.1 Formatos dos arquivos de alinhamento para Nest

Nest adota o formato de alinhamento pir. Cada arquivo pode conter vários blocos de alinhamento, sendo que cada bloco de alinhamento é uma região marcada

pelos símbolos ♯start e ♯end [19].

Formato para um único bloco de alinhamento

O bloco de alinhamento mais simples é composto de dois alinhamentos pir, correspondendo à seqüência molde e à seqüência alvo, respectivamente. O exemplo a seguir mostra o formato com um bloco.

```
♯start !exemplo 1
>P1;1hbaA
structureN:1hbaA:0:A:141:A
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFD--L----SH-GS
AQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAA
HLPAEFTPAVHASLDKFLASVSTVLT-SK-YR*
>P1;SEQ
sequence:myg:1:  :150:  :
GLSDGEWQLVLNVWGKVEADVAGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKASE
DLKKHGNTVLTALGGILKKKGHHEAELTPLAQSHATKHKIPVKYLEFISEAIIQVLQSK
HPGDFGADAGAMSKALELFRNDMAAKYKLG*
♯end
```

No arquivo de alinhamento, linhas com comentário começam com !. Se o arquivo contém mais de um bloco de alinhamento, os símbolos ♯start e ♯end devem existir para separar os diferentes blocos de alinhamento, caso contrário os símbolos podem ser omitidos.

A linha iniciada com >P1; é chamada de linha pir inicial, que é usada para indicar um novo alinhamento pir. Em cada bloco de alinhamento deve existir pelo menos duas linhas iniciando com >P1;. A linha seguinte ao >P1; é chamada de linha de marca pir, que é usada para indicar informações da seqüência alvo ou do molde sobre o alinhamento pir.

Em cada bloco de alinhamento deve existir um e somente um alinhamento pir para estrutura e seqüência, nesta ordem. Os alinhamentos pir de estrutura e seqüência são marcados pelas palavras structure e sequence iniciando a linha de

marca. Como pode ser visto no exemplo acima.

Marcas de estrutura e seqüência

As linhas de marca de estrutura e seqüência têm o seguinte formato:

<marca>:<nome>:<num. res. inicial>:<id cadeia>:<num. res. final>:<id cadeia>

Onde:

- <marca> pode ser `structure`, `structureX`, `sequence`, `sequence-X`, sendo X qualquer caracter de a-z, A-Z ou 0-9. A marca `structure` indica que o alinhamento pir é da estrutura molde e a marca `sequence` indica que o alinhamento pir é da seqüência alvo.

- <nome> deve ser o nome do arquivo PDB da estrutura molde, no caso de linha iniciada com a <marca> `structure`. No caso da linha iniciada com a <marca> `sequence`, <nome> é um nome que deve ser dado para identificar a seqüência, podendo ser omitido. Porém, se for omitido o modelo construído não será gravado no disco rígido. O arquivo do modelo final construído pela ferramenta Nest será gravado no disco rígido com o nome <nome>_final.pdb.

- <id cadeia> indica qual cadeia deverá ser usada no alinhamento pir. Se <id cadeia> não for especificada, o padrão é usar a primeira cadeia. Se for especificada, a <id cadeia> que fica depois de <num. res. inicial> e a que fica depois de <num. res. final> devem ser iguais.

- <num. res. inicial> e <num. res. final> identificam o número do resíduo inicial e o número do resíduo final na cadeia. Não são obrigatórios.

Todos os demais caracteres nas linhas que seguem a linha de marca e acima da próxima linha `>P1;` são os resíduos que formam a seqüência. Somente as letras padrão, indicadas na Tabela 2.1, que identificam os aminoácidos são aceitos, além do hífen "`-`". Outros caracteres que não sejam esses, são tratados como hífens.

Formato para múltiplos blocos de alinhamento

Para arquivo com múltiplos blocos de alinhamento, cada um dos alinhamentos é delimitado pelos símbolos ♯start e ♯end, como no exemplo abaixo.

```
#start
>P1;primeiro bloco de alinhamento
structure:test1:*:A:*:A
ERYENLFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTL
iDAGADALELGVPFSDPLADG
>P1; sequence:query1::::::::
-----------MFKDGSLIPYLTAGDPDKQSTLNFLLAL
-DEYAGAIELGIPFSDPIADG
#end
#start
>P1;segundo bloco de alinhamento
structure:test2:*:A:*:A
MERYENLFAQLNDRR--EGAFVPFVTLGDPGIEQSLKII
>P1; sequence:query2::::::::
--------------MFKDGSLIPYLTAGDPDKQSTLNFL
#end
```

Neste caso a Nest trata este arquivo de alinhamento como dois arquivos de alinhamento independentes, cada um contendo um bloco de alinhamento.

Formato para moldes compostos

Em alguns casos o melhor modelo pode ser construído pela fusão de duas regiões de moldes diferentes. Por exemplo, uma seqüência alvo A tem um molde B que é homólogo a A. Entretanto, pode existir uma região x na seqüência A que tem maior identidade de seqüência com uma região em um molde C. Neste caso, o modelo para a seqüência alvo A pode ser melhor construído usando o molde B e, para a região x, o molde C. O exemplo abaixo mostra o formato para moldes compostos.

```
#start
>P1; molde1
structure:tmp1:*:A:*:A:
```

```
LFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLADG
>P1;T0122
sequence-a:xa::::::::
MFKDG-----SLIP-YLTAGDPDKQSTLNFLLALDE-YAGAIELGIPFSDPIADG
>P1;alinhamento pir composto
composite:xco:3:20:
aaaabbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#end
#start
>P1; molde2
structure:tmp2:*:A:*:A
eRYENLFAQLNDRREGAFVPFVTLGDPGIEQSLKIIDTLIDAGADALELGVPFSDPLAD
>P1;
sequence-b:xb::::::::
-----MFK------DGSLIPYLTAGDPDKQSTLNFLLALDEYAGAIELGIPFSDPIADG
#end
```

Neste exemplo existem dois blocos de alinhamento. Cada alinhamento tem um alinhamento pir de estrutura e seqüência. Porém, no primeiro bloco de alinhamento, há um terceiro alinhamento pir marcado pela palavra `composite` que é chamado alinhamento pir composto. Um alinhamento pir composto não contém uma seqüência de resíduos, mas somente a seqüência de código x da <marca> `sequence-x`. Por esse motivo, no caso de molde composto, a <marca> usada no alinhamento pir de seqüência deve ser especificada na forma `sequence-x`.

Com esse tipo de entrada, Nest funcionará da seguinte forma: primeiro irá construir dois modelos para a seqüência alvo no primeiro e segundo bloco de alinhamento, criando como arquivos de saída <nome 1>_final.pdb e <nome 2>_final.pdb, para o primeiro e segundo bloco respectivamente. Em seguida, construirá a estrutura composta, dando como saída o arquivo <nome comp.>_final.pdb contendo a estrutura composta. Neste exemplo o nome do arquivo composto seria xco_final.pdb.

Estes são alguns dos formatos do arquivo de entrada, contendo o alinhamento, para o processo de predição pela ferramenta Nest. Na Seção seguinte veremos in-

formações sobre o outro tipo de arquivo utilizado pela Nest no processo de predição, arquivo do PDB.

2.2.2 *Protein Data Bank*

A ferramenta Nest, descrita na Seção anterior, utiliza, como entrada para seu processo, arquivos do PDB contendo dados estruturais de proteínas existentes. Esses arquivos são oriundos de uma base de dados denominada *Protein Data Bank* (PDB) e contém informações como coordenadas dos átomos na estrutura, citações bibliográficas, informações sobre a estrutura secundária e primária da proteína, bem como dados dos experimentos utilizados na obtenção da estrutura da proteína.

O *Protein Data Bank* (PDB) é um repositório de estruturas tridimensionais de macro-moléculas biológicas determinadas experimentalmente. Abriga os arquivos de coordenadas atômicas que identificam a localização de átomos em milhares de proteínas e ácidos nucléicos, informações estas obtidas usando técnicas de cristalografia de raio-X e espectroscopia de RNM.

Programas de imagem podem interpretar os arquivos abrigados no PDB, permitindo desta forma que cientistas visualizem a forma de uma proteína e possam avaliar, por exemplo, como a proteína pode interagir com drogas.

O PDB foi iniciado em 1971. Nesta época somente cerca de uma dúzia de estruturas de proteína tinham sido descobertas e somente alguns cientistas necessitavam ter acesso aos dados. Em 1998 o PDB continha aproximadamente 8.000 entradas e esse número vem crescendo desde então. Hoje, a procura por dados estruturais cresceu em vários campos da biologia e o PDB passou a ser a base principal para publicação e recuperação desses dados de estrutura [12].

No PDB, os biólogos podem depositar dados estruturais de macro-moléculas, fazer *downloads* dos arquivos contendo dados estruturais e checar a situação de dados estruturais enviados por cientistas para serem publicados no PDB, pois antes de serem publicados os dados passam por análise e formatação. O acesso aos dados no PDB é gratuito e pode ser acessado na Web pelo site www.pdb.org.

O PDB tem a tarefa de anotar (formatar), validar e publicar dezenas de arquivos de estruturas toda semana. Dependendo do tamanho da proteína, o processamento de dados pode levar de algumas horas a vários dias. Esse processamento

de dados envolve a anotação dos dados enviados ao PDB no formato padrão do PDB e a validação desses dados, para verificar se estão corretos, através de comparações com valores padrões. Essas atividades de anotação e validação são realizadas pela equipe que gerencia o PDB com o auxílio de ferramentas computacionais.

A submissão de dados estruturais de proteínas para ser publicado no PDB, passa por vários passos até ser realmente publicado. Inicialmente o pesquisador faz um *upload* dos dados estruturais, resultante de seus experimentos, para um servidor do grupo que gerencia o PDB. O pesquisador recebe então um PDBid, um código único de quatro caracteres identificando a submissão, com o qual pode verificar posteriormente o andamento de sua submissão. Em seguida, um membro da equipe que gerencia o PDB realiza a anotação dos dados no formato padrão e também roda a ferramenta de validação para verificar a qualidade destes dados. O arquivo anotado é então retornado ao pesquisador para revisão. Uma vez que o pesquisador e a equipe do PDB aprovam o arquivo, ele está pronto para ser publicado via uma atualização semanal enviada para o SDSC (San Diego Supercomputer Center).

Os dados estruturais armazenados no PDB são então utilizados pela Nest através de arquivos .pdb contendo dados estruturais de proteínas, que são recuperados pelo processo de *download* no site do PDB. Devem ser recuperados os arquivos .pdb referentes a cada estrutura homóloga que aparece no arquivo de alinhamento utilizado pela Nest. Estes arquivos recuperados devem então ser armazenados no computador onde a Nest é executada. Cada estrutura é identificada com o PDBid, citado anteriormente, e os arquivos contendo os dados estruturais das proteínas são disponibilizados no site do PDB com o nome <PDBid>.pdb. No Capítulo 4 são mostrados alguns dos dados armazenados nos arquivos .pdb.

Capítulo 3

Uma ontologia para anotação de dados de proteínas

No domínio da Biologia Molecular, mais especificamente no domínio proteômico onde são estudadas as proteínas, suas estruturas e funções, verificamos a *Protein Ontology* (PO) [23, 14, 15] como uma ontologia para anotação de dados de proteínas, podendo ser utilizada no processo de reuso de conhecimento baseado em ontologia, na predição de estrutura de proteína. Trata-se de uma ontologia para descrever dados de proteínas, desde seqüências de aminoácidos, dados de ligações entre seus átomos, até dados de funções da proteína. Durante a pesquisa, esta foi a única ontologia encontrada no domínio da Biologia que descrevia o domínio das proteínas. A PO mostrou ser adequada para uso no projeto por ser uma ontologia com conceitos que fazem parte do processo de predição de estrutura de proteína, como conceitos referentes a estruturas de proteínas e suas ligações químicas. A análise desta ontologia mostrou que o uso de seus conceitos poderiam trazer ganho de eficiência (tempo de execução) no processo de predição, pois verificou-se a existência de conceitos na PO referentes a estrutura de proteína que podem ser usados diretamente no processo de predição, eliminando cálculos realizados pela Nest para obter dados de estrutura. Isso será visto nos capítulos seguintes. Por esses motivos a PO foi escolhida para aplicação no processo de predição.

A PO é um projeto da Universidade de Tecnologia de Sydney, Austrália, e surgiu da necessidade de integrar formatos de dados de proteínas e fornecer um vocabulário unificado e estruturado para representar conceitos de síntese de proteínas,

dando suporte à integração de fontes heterogêneas de dados biológicos e proteínas. O uso da PO possibilita converter dados coletados por geneticistas e biólogos moleculares em informações que cientistas, físicos e outros profissionais e pesquisadores podem usar para entender melhor as relações dentro de moléculas de proteínas, interações entre duas moléculas de proteína e interações entre proteínas e outras macromoléculas a nível de célula [14, 15].

O Protégé [22] é utilizado como ferramenta de construção da PO, é um editor de ontologias e framework de base de conhecimento. A linguagem de representação é a *Web Ontology Language* (OWL) [23, 25]. A OWL é uma linguagem para definição e instanciação de ontologias Web [17].

As instâncias de dados de proteínas registradas com a PO têm como fonte de dados o PDB, SCOP (*Structural Classification of Proteins*), OMIM (*Online Mendelian Inheritance in Man*) e várias publicações/documentos de cientistas publicadas para reunir dados de proteína [23]. A criação da PO teve como referência estas fontes de dados e ela especifica conhecimento dessas várias fontes como detalhes dos registros de proteínas no PDB, representação estrutural 3D de proteínas, dobramentos e domínios estruturais conservados em proteínas, domínios funcionais e famílias, além de diversas restrições como defeitos genéticos e propriedades químicas da célula que afetam a estrutura molecular final da proteína [13].

A Seção seguinte descreve a estrutura da PO com descrição de suas classes e propriedades das classes.

3.1 Estrutura da *Protein Ontology*

A PO consiste de conceitos e relacionamentos entre eles, sendo os conceitos descritores de dados proteômicos. Tem uma classificação hierárquica de conceitos representados como classes, do geral para o específico, além de lista de atributos relacionados a cada classe e um conjunto de relacionamento entre classes. A Figura 3.1 mostra a hierarquia de classes da PO, onde no quadro vermelho tem-se as subclasses de *StructuralDomains* que descrevem estruturas secundárias de proteínas.

A classe principal da PO é a *ProteinOntology* e possui as seguintes propriedades:

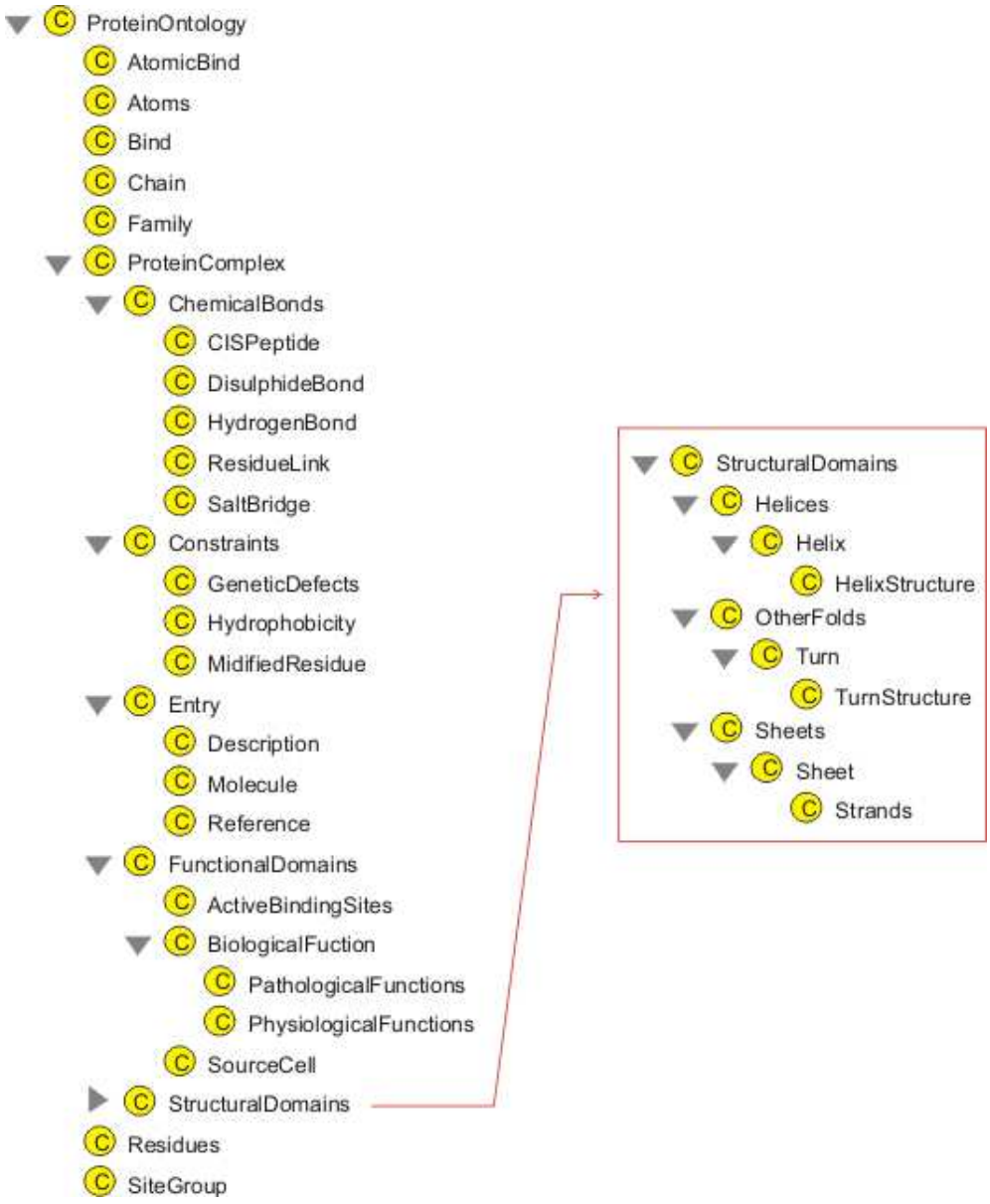

Figura 3.1: Hierarquia de classes da *Protein Ontology*

- *ProteinOntologyID*: identificador da proteína na ontologia.
- *ProteinOntologyDescription*: uma descrição da proteína.

Detalhes e propriedades de resíduos em uma seqüência de proteína são descritos como instâncias do conceito *Residues*. Este conceito possui as seguintes propriedades:

- *Residue*: a identificação do aminoácido, formada por três letras, ex. CYS.
- *ResidueName*: nome do resíduo, ex. Cistina.

- *ResidueProperty*: informações do resíduo como sua fórmula química; seu tamanho e letra identificadora, ex. C.

Cadeia de resíduos são descritas como instâncias do conceito *Chains*. Uma proteína pode ter mais de uma cadeia e cada uma delas é descrita como instância do conceito *Chains*. Este conceito possui as seguintes propriedades:

- *Chain*: a identificação da cadeia, formada por uma letra, ex. A, B. Cada cadeia numa proteína é identificada por uma letra do alfabeto.
- *ChainName*: nome da cadeia, ex. Cadeia A.
- *ChainProperty*: propriedades da cadeia.

Todos os dados de estrutura tridimensional dos átomos das proteínas são descritos como instâncias do conceito *Atoms*. Um átomo é uma unidade de um resíduo. Um resíduo é formado por vários átomos ligados quimicamente. Nas bases de dados de proteínas, os registros de átomos descrevem as coordenadas atômicas x, y, z e outras informações sobre cada átomo no resíduo. O conceito *Atoms* possui as seguintes propriedades:

- *Atom*: letra que identifica o átomo. ex. H (hidrogênio), O (oxigênio).
- *AtomID*: um número sequencial que identifica o átomo anotado pela ontologia.
- *ATOMResSeqNum*: número sequencial que identifica o resíduo na seqüência de resíduos da proteína, do qual o átomo faz parte.
- *Charge*: carga elétrica no átomo.
- *Element*: símbolo de elemento do átomo.
- *Occupancy*: ocupação ortogonal do átomo na estrutura de cristal da molécula.
- *SegmentIdentifier*: segmento específico da molécula onde o átomo está presente.
- *TemperatureFactor*: fator de temperatura do átomo.
- *X*: coordenada X do átomo.

- *Y*: coordenada Y do átomo.
- *Z*: coordenada Z do átomo.
- *_ATOM_Chain*: referência para a instância do conceito *Chains* que descreve a cadeia da qual o resíduo onde o átomo se encontra faz parte.
- *_ATOM_Residue*: referência para a instância do conceito *Residues* que descreve o resíduo onde o átomo se encontra.

As famílias e super famílias de proteínas são descritas como instâncias do conceito *Family*. Uma família de proteínas é um grupo de proteínas relacionadas evolutivamente e a super família é um grupo de famílias de proteínas. As informações de família e super família com suas classificações estão registradas na base de dados SCOP. O conceito *Family* possui as seguintes propriedades:

- *ProteinFamily*: família da proteína.
- *ProteinSuperFamily*: super família da proteína.

Ligações químicas do tipo ponte de hidrogênio, ligações de resíduos e pontes salinas são descritas como instâncias do conceito *AtomicBind*. As bases de dados de proteínas descrevem cada uma dessas ligações químicas identificando os átomos envolvidos nas ligações. Porém, cada instância do conceito *AtomicBind* descreve as informações de cada átomo da ligação individualmente. Os conceitos *HydrogenBond*, *ResidueLink* e *SaltBridge* é que descrevem os dados da ligação em si, reunindo os átomos envolvidos na ligação através de referências a instâncias do conceito *AtomicBind* para cada átomo da ligação. Este conceito possui as seguintes atividades:

- *AtomicBindResSeqNum*: número sequencial do resíduo na seqüência de resíduos da proteína.
- *AtomicBindSymmetry*: operador de simetria.
- *_AtomicBind_ATOM*: referência para a instância do conceito *Atoms* que descreve o átomo em questão que faz parte da ligação química.
- *_AtomicBind_Chain*: referência para a instância do conceito *Chains* que descreve a cadeia onde o átomo da ligação química está presente.

- *_AtomicBind_Residue*: referência para a instância do conceito *Residues* que descreve o resíduo onde o átomo da ligação química se encontra.

O conceito *HydrogenBond* que descreve as ligações químicas do tipo ponte de hidrogênio possui as seguintes propriedades:

- *_HydrogenBond_AtomicBind1*: referência para a instância do conceito *AtomicBind* que descreve o primeiro átomo na ligação.
- *_HydrogenBond_AtomicBind2*: referência para a instância do conceito *AtomicBind* que descreve o segundo átomo na ligação.
- *_HydrogenBond_AtomicBindH*: referência para a instância do conceito *AtomicBind* que descreve o átomo de hidrogênio na ligação.

O conceito *ResidueLink* que descreve as ligações de resíduos possui as seguintes propriedades:

- *_ResidueLink_AtomicBind1*: referência para a instância do conceito *AtomicBind* que descreve o primeiro átomo na ligação.
- *_ResidueLink_AtomicBind2*: referência para a instância do conceito *AtomicBind* que descreve o segundo átomo na ligação.

O conceito *SaltBridge* que descreve as ligações químicas do tipo potes salinas possui as seguintes propriedades:

- *_SaltBridge_AtomicBind1*: referência para a instância do conceito *AtomicBind* que descreve o primeiro átomo na ligação.
- *_SaltBridge_AtomicBind2*: referência para a instância do conceito *AtomicBind* que descreve o segundo átomo na ligação.

Ligações químicas do tipo ligações dissulfídicas e CIS peptídeo são descritas como instâncias do conceito *Bind*. As bases de dados de proteínas descrevem cada uma dessas ligações químicas identificando os resíduos envolvidos nas ligações. Porém, cada instância do conceito *Bind* descreve as informações de cada resíduo da ligação individualmente. Os conceitos *DisulphideBond*, *CISPeptide* é que descrevem

os dados da ligação em si, reunindo os resíduos envolvidos na ligação através de referências a instâncias do conceito *Bind* para cada resíduo da ligação. Este conceito possui as seguintes propriedades:

- *BindResSeqNum*: número sequencial do resíduo na seqüência de resíduos da proteína.
- *BindSymmetry*: operador de simetria.
- *_Bind_Chain*: referência para a instância do conceito *Chains* que descreve a cadeia onde o resíduo da ligação química está presente.
- *_Bind_Residue*: referência para a instância do conceito *Residues* que descreve o resíduo em questão que faz parte da ligação química.

O conceito *DisulphideBond* que descreve as ligações dissulfídicas possui as seguintes propriedades:

- *_DisulphideBond_Bind1*: referência para a instância do conceito *Bind* que descreve o primeiro resíduo na ligação.
- *_DisulphideBond_Bind2*: referência para a instância do conceito *Bind* que descreve o segundo resíduo na ligação.

O conceito *CISPeptide* que descreve as ligações químicas do tipo CIS peptídeo possui as seguintes propriedades:

- *AngleMeasure*: medida do ângulo em graus formado pela ligação.
- *Model*: identifica o modelo específico.
- *_CISPeptide_Bind1*: referência para a instância do conceito *Bind* que descreve o primeiro resíduo na ligação.
- *_CISPeptide_Bind2*: referência para a instância do conceito *Bind* que descreve o segundo resíduo na ligação.

O complexo protéico e as moléculas contidas nele são descritos como instâncias do conceito *Entry* e seus sub-conceitos *Description*, *Molecule* e *Reference*.

Seqüências de proteínas e dados de estrutura são descritos como instâncias do conceito *Structure* e seus sub-conceitos *ATOMSequence* e *UnitCell*. *ATOMSequence* representa as seqüências de proteína e estruturas. *UnitCell* representa os dados experimentais de cristalografia usado para obter a estrutura da proteína.

As estruturas secundárias *helix* e *sheet* das proteínas são descritas como instâncias do conceito *StructureDomains*. Sendo que a estrutura secundária *helix* é descrita como instância do sub-conceito *Helix* e a estrutura secundária *sheet* como instância do sub-conceito *Sheet*. O conceito *StructureDomains* possui as seguintes propriedades:

- *_StrDomain_Family*: referência para a instância do conceito *Family* que descreve a família da proteína onde se encontra a estrutura secundária.
- *_StrDomain_SuperFamily*: referência para a instância do conceito *Family* que descreve a super família da proteína onde se encontra a estrutura secundária.

O conceito *Helix* descreve cada estrutura secundária *helix* existente na proteína. Possui o sub-conceito *HelixStructure* que descreve a composição da estrutura. O conceito *Helix* possui as seguintes propriedades:

- *HelixID*: identificador da *helix*, formado por caracter alfanumérico.
- *HelixClass*: classe da *helix*.
- *HelixLength*: tamanho da *helix*.
- *HelixNumber*: número serial que identifica a *helix*.

O conceito *HelixStructure* descreve a composição de uma *helix*, descrevendo qual o resíduo inicial da *helix*, qual o resíduo final e a cadeia a qual pertence. Este conceito possui as seguintes propriedades:

- *HelixInitialResidueSeqNum*: número sequencial do resíduo inicial da helix na seqüência de resíduos da proteína.

- *HelixEndResidueSeqNum*: número sequencial do resíduo final da helix na seqüência de resíduos da proteína.
- *_Helix_Chain*: referência para instância do conceito *Chains* que descreve a cadeia onde a *helix* está presente.
- *_Helix_InitialResidue*: referência para a instância do conceito *Residues* que descreve o resíduo inicial da *helix*.
- *_Helix_EndResidue*: referência para a instância do conceito *Residues* que descreve o resíduo final da *helix*.

O conceito *Sheet* descreve cada estrutura secundária *sheet* existente na proteína. Possui o sub-conceito *Strands* que define a composição da estrutura. O conceito *Sheet* possui as seguintes propriedades:

- *SheetID*: identificador da *sheet*.
- *NumberStrands*: número de *strands* na *sheet*.

O conceito *Strands* descreve a composição de cada *strand* existente na *sheet*. Este conceito possui as seguintes propriedades:

- *StrandNumber*: número da *strand*.
- *StrandInitialResidueSeqNum*: número sequencial do resíduo inicial da *strand* na seqüência de resíduos da proteína.
- *StrandEndResidueSeqNum*: número sequencial do resíduo final da *strand* na seqüência de resíduos da proteína.
- *StrandCurrentResidueSeqNum*: número sequencial do resíduo na *strand* corrente.
- *StrandCurrentResidueSeqNum*: número sequencial do resíduo na *strand* anterior.
- *StrandSense*: sentido da *strand* em relação a *strand* anterior na sheet. 0 se for o primeiro, 1 se for paralelo, -1 se for anti-paralelo.

- *_Strand_Chain*: referência para a instância do conceito *Chains* que descreve a cadeia onde a *strand* está presente.

- *_Strand_InitialResidue*: referência para a instância do conceito *Residues* que descreve o resíduo inicial da *strand*.

- *_Strand_EndResidue*: referência para a instância do conceito *Residues* que descreve o resíduo final da *strand*.

- *_Strand_CurrentATOM*: referência para a instância do conceito *Atoms* que descreve o átomo na *strand* corrente.

- *_Strand_CurrentResidue*: referência para a instância do conceito *Residues* que descreve o resíduo na *strand* corrente.

- *_Strand_PreviousATOM*: referência para a instância do conceito *Atoms* que descreve o átomo na *strand* anterior.

- *_Strand_PreviousResidue*: referência para a instância do conceito *Residues* que descreve o resíduo na *strand* anterior.

A classificação de domínio funcional é descrita como instância do conceito *FunctionalDomains*. Este conceito descreve a fonte celular e a fonte do organismo da proteína através de seu sub-conceito *SourceCell*, descreve a função biológica da proteína através do sub-conceito *BiologicalFunction* e, descreve também, sítios ativos de ligações da proteína através do sub-conceito *ActiveBindingSites*. Esses sítios ativos de ligações são descritos como uma coleção de vários grupos de sítios, definidos através do conceito genérico *SiteGroups*.

As restrições que afetam a conformação estrutural final de uma proteína são definidas através do conceito *Constraints*. As restrições descritas na PO são: defeitos genéticos presentes nos genes que estão presentes nas moléculas, propriedades hidrofóbicas de proteínas e modificações na seqüência de resíduos da proteína devido a mudanças em ambientes químicos e mutações. Essas restrições são descritas respectivamente pelos conceitos *GeneticDefects*, *Hydrophobicity* e *ModifiedResidue*.

No próximo Capítulo será mostrada a aplicação desta ontologia no processo de predição de estrutura de proteína.

Capítulo 4

Reuso de conhecimento baseado em ontologia na predição de estrutura de proteína

Ontologias possibilitam compartilhamento e reuso de conhecimento em dois aspectos: a) reuso das próprias ontologias na construção de novas ontologias ou como parte de aplicações baseadas em conhecimento; e b) reuso de conhecimento que é fundamentado em ontologias ou delas derivado.

Em [4] é proposto que em resolução de problemas restringida por ontologia, uma vez que a ontologia e o mecanismo geral de resolução do problema estão estabelecidos, instâncias existentes de soluções podem ser reusadas para produzir soluções para novas formulações do problema. Nesse cenário de reuso de conhecimento, tem-se a descrição de problema através de uma ontologia, que fornece um vocabulário formal, e o reuso de soluções existentes por algum mecanismo de resolução de problema que utiliza restrições ontológicas, como mostra a Figura 4.1.

Figura 4.1: Cenário de reuso de conhecimento. Adaptada de [4].

Esta técnica foi demonstrada no domínio de modelagem ecológica, onde des-

crições de problemas consistem de dados ecológicos anotados com a ontologia Ecolingua [5] e as soluções são estruturas de modelos ecológicos de simulação automaticamente construídos a partir dos dados anotados. As estruturas de modelos previamente estabelecidos são reusadas para construir novos modelos dadas anotações de novos conjuntos de dados [3, 4].

Em termos gerais, isto é uma técnica de engenharia do conhecimento no qual: i) dados que alimentam uma estrutura de modelo são anotados com uma ontologia do domínio considerado; ii) estruturas são abstraídas de modelos de referência existentes e suas características casadas, ou associadas, com os conceitos ontológicos que descrevem os dados do domínio; iii) Desta forma, as estruturas de referência são reusadas pra produzir novas estruturas de modelo compatíveis com os novos dados.

A Figura 4.2 mostra um esquema deste processo de reuso de modelos de referência por um mecanismo de modelagem na geração de um novo modelo. Dados anotados informam o mecanismo de modo que o modelo gerado seja consistente com propriedades semânticas dos dados capturadas pela ontologia.

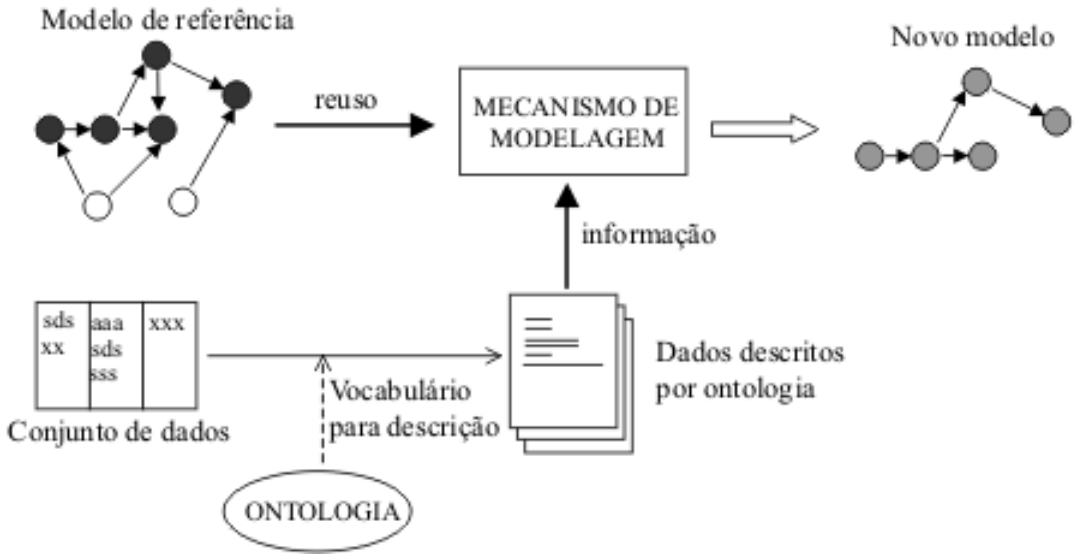

Figura 4.2: Reuso de modelo estrutural. Adaptada de [4].

As estruturas de referência são usadas como um esqueleto, sendo casadas com as anotações de dados fornecidas. Para realizar este casamento, a estrutura de referência é reduzida aos conceitos ontológicos que descrevem os dados do domínio. O casamento é realizado entre a estrutura de referência e a descrição do novo problema através da ontologia para verificar relações existentes entre esta estrutura e

os dados anotados. A partir destas relações, um novo modelo compatível com os novos dados é criado.

A predição de estrutura de proteína pelo método de predição por homologia já emprega o reuso de conhecimento, pois realiza a predição da estrutura de uma proteína desconhecida baseada em informações de estruturas de proteínas já conhecidas que possuem alguma relação com a proteína de estrutura desconhecida.

Como vimos anteriormente, o compartilhamento e reuso de conhecimento fundamentado em ontologia é possível. Em [3] isto é feito no domínio da Ecologia no processo de construção de estruturas de modelos de simulação. Aqui replicamos a abordagem no domínio da Biologia Molecular, no processo de predição de estrutura de proteína utilizando a *Protein Ontology* (PO) descrita na Seção 3.

Analisando a Figura 4.2, onde temos o modelo de reuso de conhecimento baseado em ontologia, vemos os dados oriundos de uma base de dados sendo anotados por uma ontologia. Em nosso cenário, seguindo este modelo, temos então dados oriundos de bases de dados, sendo anotados pela PO. Como vimos na Seção 3, a PO teve sua construção baseado em certas bases existentes de dados sobre proteínas, sendo sua principal referência a base PDB (*Protein Data Bank*). Baseados nesta informações, e após verificar que os dados necessários para o processo de predição poderiam ser recuperados desta base, neste trabalho tivemos a necessidade de construir uma ferramenta que realizasse a anotação automatizada de dados do PDB com a PO. Essa ferramenta é descrita na Seção a seguir.

4.1 Ferramenta de anotação

Para anotar os dados de proteína com a PO, foi criada a ferramenta anotaPO, construída em linguagem Perl, por ser uma linguagem apropriada para se trabalhar com arquivos de texto, formato este usado pelo PDB para disponibilizar os dados de proteínas armazenados nele.

Como a PO é escrita em OWL, a ferramenta anotaPO gera como saída dados anotados em OWL. Como o projeto da PO, pela Universidade de Tecnologia de Sydney, realiza um trabalho de gerar arquivos contendo os dados de uma proteína anotados pela PO em OWL, criamos a ferramenta anotaPO de forma a gerar esses

arquivos também em formato OWL.

A anotaPO realiza a tarefa de recuperar cada dado da base PDB e anotar o mesmo com o conceito adequado na PO. A anotaPO identifica o conceito adequado para anotar o dado através da sigla que identifica o dado na base PDB. A seguir temos descrições das anotações de alguns dados de proteínas realizados pela anotaPO onde são citadas as siglas que identificam cada dado na base PDB.

Dados de átomos

Os dados sobre os átomos constituintes de uma proteína são identificados na base de dados pela sigla ATOM e contém dados de coordenadas atômicas x, y, z em Angstroms de cada átomo, o resíduo ao qual pertence, a cadeia da qual faz parte, a fator de temperatura e a carga no átomo. Esses dados são anotados pela ferramenta com o conceito *Atoms* da PO. Abaixo um exemplo de dados de átomos anotados com a PO.

```
<Atoms rdf:ID="AtomoInstance_18_22">
<X rdf:datatype="http://www.w3.org/2001/XMLSchema#float">35.481</X>
<Y rdf:datatype="http://www.w3.org/2001/XMLSchema#float">24.871</Y>
<Z rdf:datatype="http://www.w3.org/2001/XMLSchema#float">25.772</Z>
<ATOMResSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
4</ATOMResSeqNum>
<TempratureFactor rdf:datatype="http://www.w3.org/2001/XMLSchema#float">
18.76</TempratureFactor>
<Element rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
C</Element>
<Atom rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CA</Atom>
<Occupancy rdf:datatype="http://www.w3.org/2001/XMLSchema#float">
1.00</Occupancy>
<SegmentIdentifier rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
</SegmentIdentifier>
< _ATOM_Chain rdf:resource="#ChainInstance_18_A"/>
<AtomID rdf:datatype="http://www.w3.org/2001/XMLSchema#int">22</AtomID>
```

```
< _ATOM_Residue rdf:resource="♯Residue_Inst_20"/>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PO_18</ProteinOntologyID>
</Atoms>
```

No exemplo acima o que aparece em verde corresponde a identificação dos atributos da classe *Atoms*, onde aparece o nome do atributo e o tipo de dado com o qual o atributo está definido na ontologia. Em vermelho aparecem os dados. Por exemplo, 35.481 é o valor da coordenada atômica x do átomo anotado e encontra-se entre as *tags* <X rdf:datatype="http://www.w3.org/2001/XMLSchema♯float"> e </X> que correspondem ao atributo X da classe átomo, onde esse atributo representa a coordenada atômica x de um átomo. A mesma estrutura é seguida para os demais exemplos de anotação com a PO que são mostrados no decorrer deste Capítulo.

Dados de resíduos

Os resíduos na base de dados PDB são identificados com um código de três letras, ex: GLY. Os dados referentes a cada resíduo como nome, peso molecular e letra que identifica o resíduo, ex. G, podem ser encontrados no PDB. Esses dados são anotados pela ferramenta com o conceito *Residues* da PO. Abaixo um exemplo de resíduo anotado pela PO.

```
<Residues rdf:ID="Residue_Inst_3">
<ResidueProperty rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
FORMULA: C5 H10 N2 O3</ResidueProperty>
<ResidueProperty rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
MOLECULAR WEIGHT: 146.15</ResidueProperty>
<Residue rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
GLN</Residue>
<ResidueName rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
GLUTAMINE</ResidueName>
<ResidueProperty rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
```

```
1-LETTER CODE: Q</ResidueProperty>
</Residues>
```

Dados de cadeia

As proteínas são formadas por cadeias e o PDB descreve as informações das cadeias que constituem a proteína nos registros identificados pela sigla COMPND. No PDB as cadeias são identificadas por uma letra do alfabeto ou um número de 0 a 9, porém é mais frequente a utilização de letras. Além disso, COMPND contém informações como nome da cadeia e demais propriedades que forem importantes registrar sobre a cadeia. Essas dados são anotados com o conceito *Chains* da PO. Abaixo um exemplo.

```
<Chains rdf:ID="ChainInstance_18_B">
<Chain rdf:datatype="http://www.w3.org/2001/XMLSchema#string">B</Chain>
<ChainName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
CHAIN B</ChainName>
<ChainProperty rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
BETA CHAIN</ChainProperty>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_18</ProteinOntologyID>
</Chains>
```

Identificação da proteína

Dados descrevendo as proteínas são encontrados no registro identificado pela sigla TITLE no PDB. Este registro contém informações da proteína e do experimento no qual esta foi analisada. É na verdade um resumo identificando o registro da proteína. Esses dados são anotados com o conceito *ProteinOntology* da PO. Abaixo um exemplo.

```
<ProteinOntology rdf:ID="ProtOntoInstance_18">
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_18</ProteinOntologyID>
```

```
<ProteinOntologyDescription rdf:datatype="http://www.w3.org/2001/XMLSchema
♯string">
CRYSTAL STRUCTURE OF THE COMPLEX OF BACTERIAL
TRYPTOPHAN SYNTHASE WITH THE TRANSITION STATE ANALOGUE
INHIBITOR 4-(2- HYDROXYPHENYLTHIO)-1-BUTENYLPHOSPHONIC ACID
</ProteinOntologyDescription>
```

Dados de ligações químicas

As proteínas são formadas por cadeias de aminoácidos constituídos de um conjunto de átomos. Esses átomos se ligam através de ligações químicas. As principais ligações químicas que podem ser encontradas em uma proteína são Ponte de Hidrogênio, Pontes Salinas, Ligação de Resíduos, Ligação CIS Peptídio e Ponte Dissulfídica.

As Pontes de Hidrogênio de uma proteína estão especificadas, cada uma, nos registros identificados pela sigla HYDBND no PDB. Nestes registros encontramos informações como: identificação dos dois átomos que participam da ligação, juntamente com a identificação dos resíduos ao qual esses átomos pertencem, as cadeias das quais esses resíduos fazem parte e o operador de simetria dos átomos. Além disso, encontramos as informações do átomo de hidrogênio que participa da ligação, juntamente, também, com a identificação do resíduo ao qual pertence e a cadeia da qual o resíduo faz parte. Esses dados são anotados com os conceitos *HydrogenBond* e *AtomicBind* da PO. Onde o conceito *AtomicBind* anota os dados da ligação em si: os átomos participantes, com seus respectivos resíduos e cadeias das quais os resíduos fazem parte, sendo que cada átomo da ligação é um elemento da classe *AtomicBind*. Já o conceito *HydrogenBond* anota o par de *AtomicBind* que identifica a ligação. Abaixo temos um exemplo de uma Ponte de Hidrogênio anotada pela PO.

```
<AtomicBind rdf:ID"AtomicBindInstance_44_1">
< _AtomicBind_Chain rdf:resource="♯ChainInstance_44_A"/>
< _AtomicBind_Residue rdf:resource="♯Residue_Inst_15"/>
<AtomicBindResSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema♯int">
```

```
64</AtomicBindResSeqNum>
<AtomicBindSymmetry rdf:datatype="http://www.w3.org/2001/XMLSchema
#string"> </AtomicBindSymmetry>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_44</ProteinOntologyID>
</AtomicBind>

<HydrogenBond rdf:ID="HydBondInstance_44_1">
< _HydrogenBond_AtomicBind1 rdf:resource="#AtomicBindInstance_44_1"/>
< _HydrogenBond_AtomicBindH rdf:resource="#AtomicBindInstance_44_2"/>
< _HydrogenBond_AtomicBind2 rdf:resource="#AtomicBindInstance_44_3"/>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_44</ProteinOntologyID>
</HydrogenBond>
```

As pontes salinas existentes em um proteína são descritas, cada uma, nos registros identificados pela sigla SLTBRG. Nestes registros encontramos informações como: identificação dos dois átomos que participam da ligação, juntamente com a identificação dos resíduos aos quais esses átomos pertencem, as cadeias das quais esses resíduos fazem parte e o operador de simetria dos átomos. Esses dados são anotados com os conceitos *SaltBridge* e *AtomicBind* da PO. O conceito *AtomicBind* anota os dados da ligação em si: os átomos participantes, com seus respectivos resíduos e as cadeias das quais os resíduos fazem parte, sendo que cada átomo da ligação é um elemento da classe *AtomicBind*. Já o conceito *SaltBridge* anota o par de *AtomicBind* que identifica a ligação. Abaixo temos um exemplo de ponte salina anotada pela PO. Neste exemplo só aparece a anotação com o conceito *SaltBridge*, pois o *AtomicBind* é igual ao exemplo mostrado anteriormente na anotação de Ponte de Hidrogênio.

```
<SaltBridge rdf:ID="SaltBridgeInstance_47_1">
< _SaltBridge_AtomicBind1 rdf:resource="#AtomicBindInstance_47_1"/>
< _SaltBridge_AtomicBind2 rdf:resource="#AtomicBindInstance_47_2"/>
```

```
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_47</ProteinOntologyID>
</SaltBridge>
```

As ligações de resíduos de uma proteína são definidas, cada uma, nos registros identificados pela sigla LINK. Nestes registros são encontradas informações como: identificação dos dois átomos que participam da ligação, a identificação dos resíduos aos quais os átomos pertencem e as cadeias nas quais esses resíduos estão presentes, além do operador de simetria dos átomos. Esses dados são anotados com os conceitos *ResidueLink* e *AtomicBind* da PO. O conceito *AtomicBind* anota os dados da ligação em si: os átomos participantes, com seus respectivos resíduos e cadeias das quais os resíduos fazem parte, sendo que cada átomo da ligação é um elemento da classe *AtomicBind*. Já o conceito *ResidueLink* anota o par de *AtomicBind* que identifica a ligação. Abaixo temos um exemplo de Ligação de Resíduo anotada pela PO. Neste exemplo só aparece a anotação com o conceito *ResidueLink*, pois o *AtomicBind* é igual ao exemplo mostrado anteriormente na anotação de ponte de hidrogênio.

```
<ResidueLink rdf:ID="ResidueLinkInstance_44_1">
< _ResidueLink_AtomicBind1 rdf:resource="#AtomicBindInstance_44_4"/>
< _ResidueLink_AtomicBind2 rdf:resource="#AtomicBindInstance_44_5"/>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_44</ProteinOntologyID>
</ResidueLink>
```

As ligações CIS peptídicas de uma proteína estão especificadas, cada uma, nos registros identificados pela sigla CISPEP no PDB. Nestes registros encontramos informações como: identificação dos dois resíduos que participam da ligação, identificação das cadeias de cada resíduo que participa da ligação, o modelo específico da ligação CIS e a medida do ângulo em graus. Esses dados são anotados com os conceitos *CisPeptide* e *Bind* da PO. O conceito *Bind* anota os dados da ligação em si: os resíduos participantes, com suas respectivas cadeias, sendo que cada resíduo da ligação é um elemento da classe *Bind*. Já o conceito *CisPeptide* anota o par de

Bind que identifica a ligação, além do modelo e a medida do ângulo. Abaixo temos um exemplo de uma ligação CIS Peptídica anotada pela PO.

```
<Bind rdf:ID="BindInstance_13_1">
< _Bind_Chain rdf:resource="♯ChainInstance_13_A"/>
< _Bind_Residue rdf:resource="♯Residue_Inst_4"/>
<BindResSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema♯int">
372</BindResSeqNum>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PO_13</ProteinOntologyID>
</Bind>

<CISPeptide rdf:ID="CISPepInstance_13_1">
< _CISPeptide_Bind1 rdf:resource="♯BindInstance_13_1"/>
< _CISPeptide_Bind2 rdf:resource="♯BindInstance_13_2"/>
<AngleMeasure rdf:datatype="http://www.w3.org/2001/XMLSchema♯float">
20.75</AngleMeasure>
<Model rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">0</Model>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PO_13</ProteinOntologyID>
</CISPeptide>
```

As pontes dissulfídicas existentes em uma proteína são descritas, cada uma, nos registros identificados pela sigla SSBOND no PDB. Nestes registros encontramos informações como: identificação dos dois resíduos que participam da ligação, identificação das cadeias de cada resíduo que participa da ligação e os operadores de simetria de cada resíduo. Esses dados são anotados com os conceitos *Disulphide-Bond* e *Bind* da PO. O conceito *Bind* anota os dados da ligação em si: os resíduos participantes, cadeias e operador de simetria, sendo que cada resíduo da ligação é um elemento da classe *Bind*. Já o conceito *DisulphideBond* anota o par de *Bind* que identifica a ligação. Abaixo temos um exemplo de uma ligação do tipo ponte dissulfídica anotada pela PO. Neste exemplo só aparece a anotação com o conceito

DisulphideBond, pois o *Bind* é igual ao exemplo mostrado anteriormente na anotação de CIS peptídeo.

```
<DisulphideBond rdf:ID="DisulpBondInstance_1151_1">
< _DisulphideBond_Bind1 rdf:resource="♯BindInstance_1151_1"/>
< _DisulphideBond_Bind2 rdf:resource="♯BindInstance_1151_2"/>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PO_1151</ProteinOntologyID>
</DisulphideBond>
```

Dados de estrutura secundária

As proteínas possuem níveis de estrutura até chegar à sua estrutura completa, entre esses níveis temos a estrutura secundária da proteína, como mostra a Seção 2.1. Essa estrutura secundária é formada basicamente por pontes de hidrogênio entre os átomos dos resíduos. As principais estruturas secundárias encontradas em uma proteína são *Alpha-Helix*, *Beta-Sheet*. No PDB são registradas as estruturas secundárias da proteína, sendo que em uma proteína pode existir qualquer uma dessas estruturas combinadas. Por exemplo, uma proteína pode ter *Alpha-Helix* e *Beta-Sheets*, só *Alpha-Helix* ou só *Beta-Sheet*.

As estruturas *Alpha-Helix* de uma poteína cadastrada no PDB são descritas pelos registros identificados pela sigla HELIX. Nestes registros podemos encontrar informação da estrutura secundária como: o identificador da *helix*, nome do resíduo inicial da *helix*, a cadeia a qual o resíduo inicial pertence, identificador do resíduo inicial na seqüência de resíduos da proteína, nome do resíduo final, a cadeia do resíduo final, identificador do resíduo final na seqüência de resíduos, a classe da *helix* e o seu tamanho. Esses dados são anotados com o conceito *HelixStructure* da PO. Abaixo um exemplo de uma *helix* anotada.

```
<HelixStructure rdf:ID="HelixStructureInst_13_1">
< _Helix_Chain rdf:resource="♯ChainInstance_13_A"/>
< _Helix_InitialResidue rdf:resource="♯Residue_Inst_3"/>
< _Helix_EndResidue rdf:resource="♯Residue_Inst_5"/>
```

```
<HelixEndResidueSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema
#int">8</HelixEndResidueSeqNum>
<HelixInitialResidueSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema
#int">2</HelixInitialResidueSeqNum>
<HelixLength rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
7</HelixLength>
<HelixClass rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
Right-handed alpha (default)</HelixClass>
<HelixNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
1</HelixNumber>
<HelixID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
1</HelixID>
<HelixChain rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
A</HelixChain>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_13</ProteinOntologyID>
</HelixStructure>
```

As estruturas *Beta-Sheets* são descritas no PDB pelos registros identificados pela sigla SHEET. Nestes registros encontramos informação das estruturas secundárias *sheets* como: número de *strands* no *sheet*, nome do resíduo inicial, a cadeia do resíduo inicial, identificador do resíduo inicial na seqüência de resíduos da proteína, nome do resíduo final, a cadeia do resíduo final, identificador do resíduo final, sentido do *strand* no *sheet*, além de informações sobre átomo, resíduo e cadeia do *strand* corrente e anterior. Esses dados são anotados pelo conceito *Strands* da PO. Abaixo um exemplo.

```
<Strands rdf:ID="StrandsInstance_13_2">
< _Strand_Chain rdf:resource="#ChainInstance_13_A"/>
< _Strand_CurrentATOM rdf:resource="#AtomoInstance_13_26380"/>
< _Strand_CurrentResidue rdf:resource="#Residue_Inst_13"/>
<StrandCurrentResidueSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema
```

```
#int">22</StrandCurrentResidueSeqNum>
< _Strand_EndResidue rdf:resource="#Residue_Inst_5"/>
< _Strand_InitialResidues rdf:resource="#Residue_Inst_13"/>
< _Strand_PreviousATOM rdf:resource="#AtomoInstance_13_28690"/>
< _Strand_PreviousResidue rdf:resource="#Residue_Inst_1"/>
<StrandPreviousResidueSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema
#int">327</StrandPreviousResidueSeqNum>
<StrandEndResidueSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema
#int">26</StrandEndResidueSeqNum>
<StrandInitialResidueSeqNum rdf:datatype="http://www.w3.org/2001/XMLSchema
#int">22</StrandInitialResidueSeqNum>
<StrandNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
2</StrandNumber>
<StrandsChain rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
A</StrandsChain>
<StrandSense rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
1</StrandSense>
<NumberStrands rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
2</NumberStrands>
<SheetID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
A</SheetID>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_13</ProteinOntologyID>
</Strands>
```


A ferramenta de anotação automática criada, anotaPO, faz a anotação dos dados com a ontolgia, conforme mostrado nesta Seção. Outros dados anotados com a anotaPO podem ser vistos no Apêndice A.

4.2 Aplicação da ontologia na predição por homologia

A predição por homologia, como foi visto na Seção 2.2, possui quatro passos, onde os passos 3 e 4 correspondem, respectivamente, aos passos de construção e avaliação do modelo. A proposta deste trabalho é a aplicação da ontologia PO, mostrada na Seção 3, utilizando os dados anotados com esta ontologia nos passos 3 e 4 do processo de predição por homologia. Estes passos realizam a construção propriamente dita, onde são tratados os dados de ligações químicas e estruturas secundárias que irão interferir diretamente na estrutura final da proteína alvo. Os dados de ligações químicas e estruturas secundárias podem ser anotados com a PO e o proposto é utilizar esses dados anotados com a PO na construção da estrutura da proteína, reusando, desta forma, o conhecimento baseado na ontologia.

A ferramenta Nest, descrita na Seção 2.2.1, realiza os passos 3 e 4 do processo de predição por homologia e os passos de alinhamento e seleção do(s) molde(s) são realizados por ferramentas de alinhamento. A ferramenta Nest já recebe como arquivo de entrada o alinhamento da seqüência alvo com a(s) seqüência(s) molde(s).

Nas Seções seguintes é mostrada a estrutura da Nest e a reengenharia da ferramenta Nest para que esta utilize os dados anotados como a PO, visto na Seção anterior, realizando a predição de estrutura de proteína por homologia assistida por ontologia.

4.2.1 Classes da Nest

As principais classes da Nest que fazem parte do processo de predição na ferramenta Nest são StrFmt, Pdb, Chn, Res, Atm, Mutate, PddFix. Um diagrama contendo as principais classes da ferramenta Nest pode ser visto na Figura 4.3. Vale ressaltar aqui a dificuldade de compreender o processo de predição de estrutura de proteína da ferramenta Nest, pois esta não possui documentação detalhando o processo e nem o código-fonte possui documentação explicando o significado das classes, atributos e nem o funcionamento de cada método.

A classe StrFmt corresponde à formatação da estrutura. Funciona como uma classe principal que controla todo o processo de predição, contendo métodos

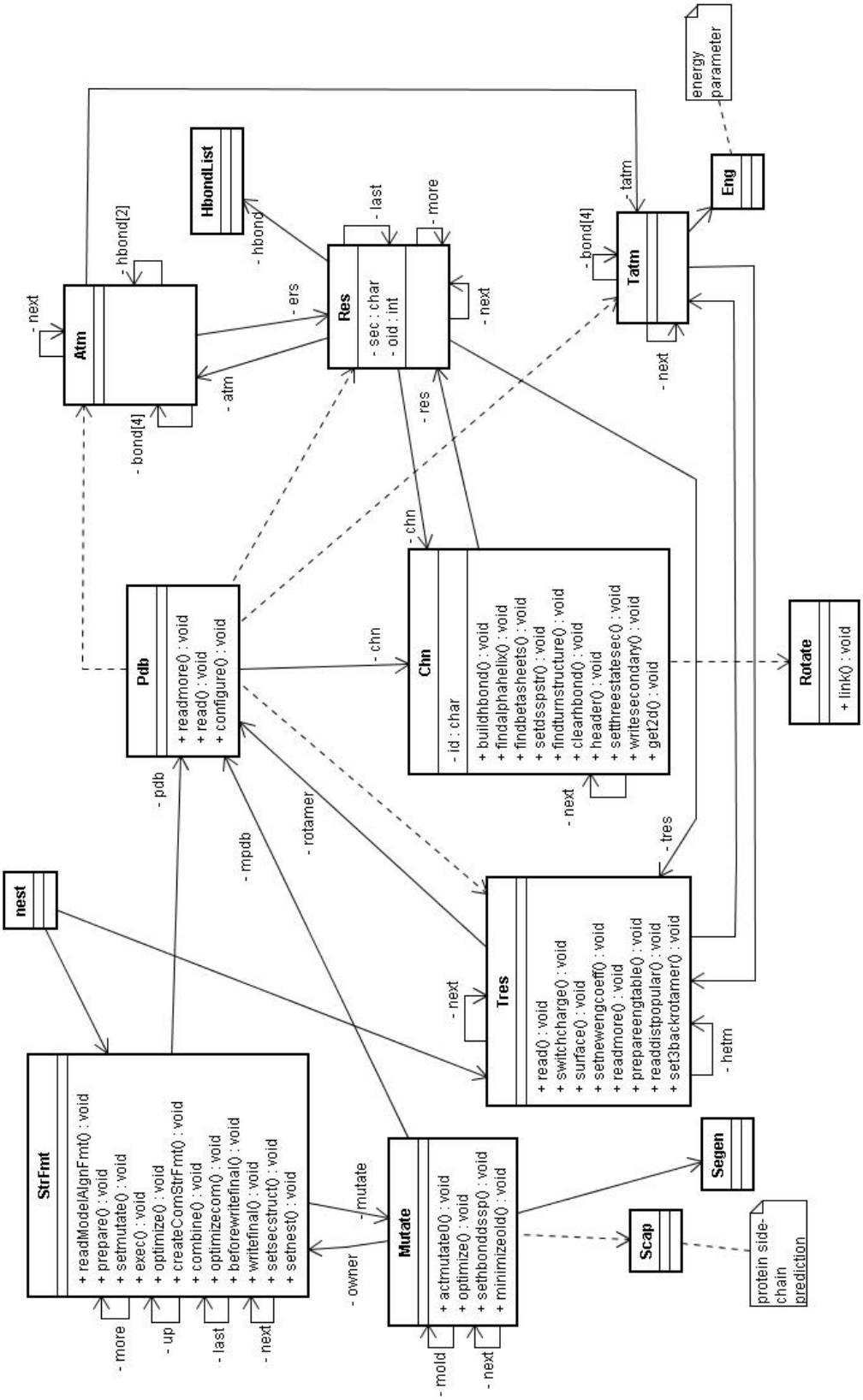

Figura 4.3: Digrama de Classes contendo classes principais da ferramenta Nest

centrais do processo como: readModelAlgnFmt, prepare, exec, optimize, combine, getstructure e setsecstructure. Posteriormente veremos onde esses métodos entram no processo e o que fazem.

A classe Pdb representa um arquivo PDB. No caso da seqüência molde, representa o arquivo PDB dessa seqüência, contendo todas as informações da estrutura molde. No caso da seqüência alvo, representa o arquivo PDB de saída contendo as informações da estrutura alvo no final do processo de predição. Durante o processo essa classe Pdb representando a seqüência alvo, sofre modificações em seus dados, pois cada operação de mutação e otimização acarreta mudança nos dados da estrutura alvo, até chegar a estrutura final.

As classes Chn, Res e Atm são classes que representam partes principais da proteína: suas cadeias, resíduos e átomos, respectivamente. A classe Chn possui um atributo, de nome res, que corresponde à lista de resíduos a ela pertencentes. Esse atributo constitui uma lista encadeada do tipo Res.

O mesmo acontece com a classe Res. Esta possui o atributo atm que corresponde ao conjunto de átomos que compõem o resíduo. Este atributo constitui uma lista encadeada do tipo Atm. Vale ressaltar a existência também de um atributo que corresponde à lista de cadeias existentes em uma proteína, sendo que esse atributo é da classe Pdb.

A classe Mutate representa todas as mutações que ocorrem durante o processo para obter a estrutura alvo final. Corresponde àquela lista de operações como troca, inserção e deleção de resíduos, realizadas no processo, como descrito na Seção 2.2.1 sobre a Nest. É nesta classe que acontece a maioria das operações para construção da estrutura alvo final, como as mutações, as minimizações de energia e otimização de estrutura.

A classe PdbFix representa o conserto da estrutura quando a seqüência molde possui átomos e/ou resíduos perdidos. Ela é responsável por fixar esses problemas, reinserindo o átomo ou resíduo perdido nos dados da proteína armazenados nas classes da Nest. Utiliza as bibliotecas de *rotamer* para recuperar o átomo ou resíduo perdido. Rotamer é o nome dado a uma coleção de ângulos que define uma conformação particular de um resíduo freqüentemente vista em proteínas. As bibliotecas de rotamer são geradas através de análises estatísticas sobre estruturas moleculares

conhecidas [7].

Na próxima Seção é descrito o processo de predição na Nest, mostrando como as classe e seus métodos, citados nesta Seção, participam do processo.

4.2.2 Processo de predição na Nest

O processo de predição na Nest é executado a partir da chamada ao executável nest e inicia com a leitura das informações do arquivo de entrada contendo o alinhamento entre a seqüência alvo e a seqüência da estrutura molde. Essa atividade é realizada pelo método **readModelAlgnFmt**, da classe StrFmt. Esse método lê cada bloco de alinhamento, caso haja mais de um, e armazena nos atributos da classe as informações das seqüências. Um diagrama contendo parte do processo de predição na Nest é mostrado na Figura 4.4.

Após a leitura das seqüências, é chamado o método **prepare**. Este método é responsável por realizar as atividades de preparação dos dados a serem utilizados no processo de predição. Algumas das atividades realizadas por este método são:

- checagem das informações de seqüência, para saber se estão corretas, como o número inicial e final dos resíduos e a identificação da cadeia.
- retirada dos *gaps* nos atributos que guardam a seqüência de resíduos.
- recuperação das estruturas nos arquivos PDB, realizando as devidas operações de organização e checagem dessas informações.
- estabelecimento da estrutura secundária.
- preparação de objetos da classe Mutate, definindo os atributos necessários. Essa atividade é realizada através da chamada ao método **setnest** da classe StrFmt. Este método realiza um cópia das informações de seqüência alvo, seqüência molde e estrutura molde para os atributos equivalentes na classe Mutate, pois é nesta classe que serão feitas as modificações na estrutura molde para se chegar na estrutura alvo, com base nas informações das duas seqüências.

A atividade de recuperação da estruturas nos arquivos PDB, citada anteriormente, é realizada pelo método **getstructure** da classe StrFmt. Esse método realiza

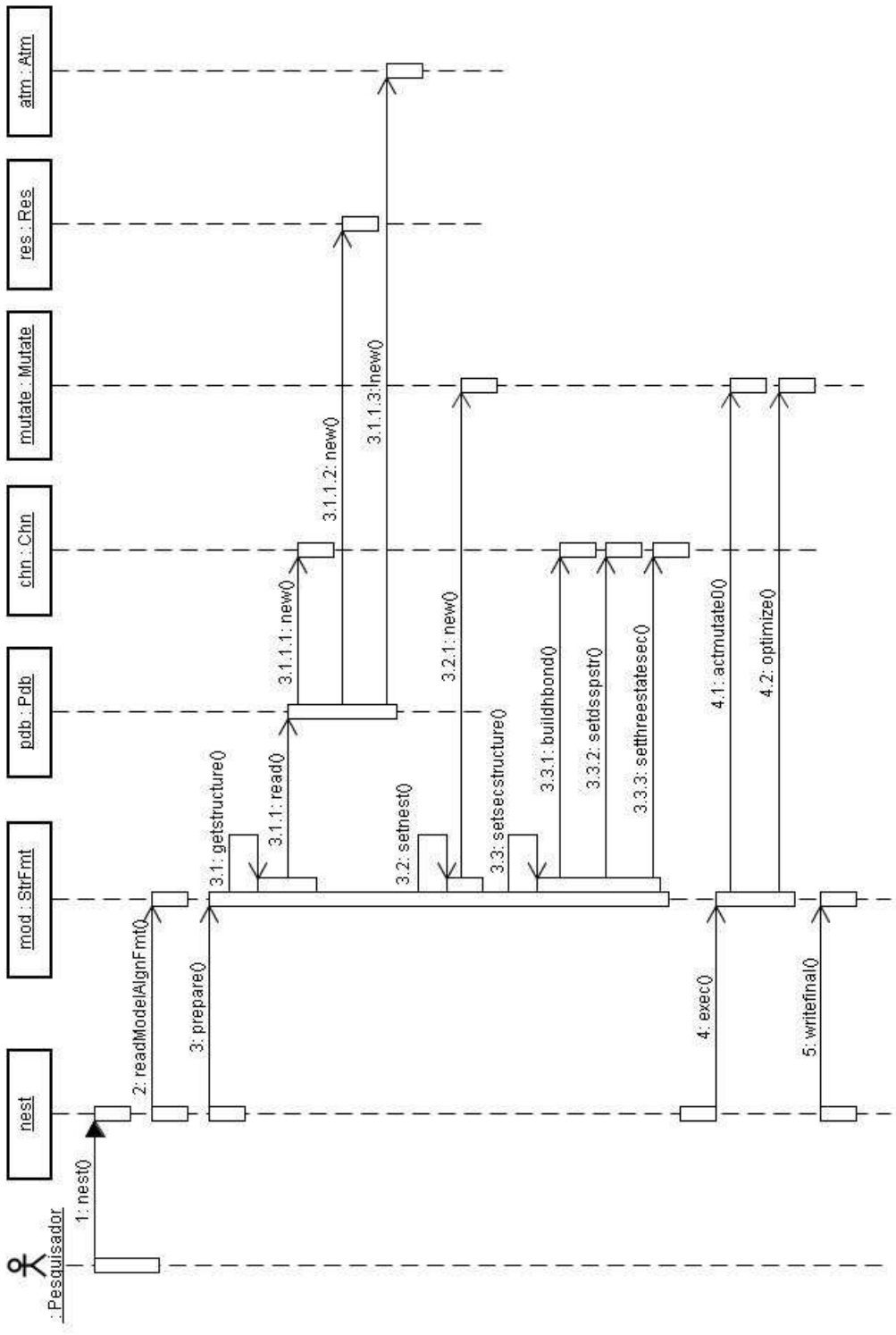

Figura 4.4: Digrama de seqüência contendo parte do processo de predição na Nest

a leitura dos arquivos PDB através do método **read** da classe Pdb. É no método **read** que os objetos das classes Chn, Res, Atm são criados e seus atributos preenchidos, armazenando todas as informações da estrutura molde. Além disso, o método **getstructure** verifica se existe átomo perdido na estrutura molde recuperada, caso

haja, chama métodos da classe PdbFix para resolver esse problema.

A atividade de estabelecimento da estrutura secundária é realizada pelo método **setsecstructure** da classe StrFmt. Esse método realiza as seguintes atividades:

- Calcula as pontes de hidrogênio existentes na estrutura molde, através da chamada ao método **buildhbond** da classe Chn.
- Calcula as *Alpha-helix* e *Beta-Sheet* existentes na estrutura molde, através da chamada ao método **setdsspstr** da classe Chn.
- Organiza a estrutura secundária no objeto da classe Res, dando ao atributo **sec** desta classe o valor '-' para cada resíduo que não tem o valor 'h' ou 'e' neste atributo. Este atributo, **sec**, guarda a informação de estrutura secundária da proteína, ficando com o valor 'e' se o resíduo em questão está presente em um estrutura *beta-sheet*, o valor 'h' se o resíduo está presente em uma estrutura *alpha-helix* e o valor '-' caso não pertença a nenhum desses dois tipos de estrutura secundária. Essa atividade é realizada através da chamada ao método **setthreestatesec** da classe Chn.

Após essa fase de preparação, começa então a construção da estrutura alvo. Para realizar essa construção é chamado inicialmente o método **exec** da classe StrFmt. A primeira atividade deste método é a chamada ao método **actmutate0** da classe Mutate, onde se concentra a construção da estrutura alvo. Após a chamada ao **actmutate0** é chamado o método **optimize** da classe Mutate, para realizar uma otimização da estrutura, levando em consideração a minimização de energia.

O método **actmutate0** realiza várias operações na estrutura molde para se chegar a estrutura alvo. Uma das atividades iniciais desse método é realizar a troca (mutação) dos resíduos da seqüência molde pelos resíduos da seqüência alvo que estão alinhados e são diferentes. Consideremos, por exemplo, um arquivo de entrada para a ferramenta Nest com o alinhamento a seguir. Todos os resíduos que aparecem em vermelho sofrerão troca, ou seja, o resíduo da seqüência molde será trocado pelo resíduo da seqüência alvo, que encontra-se na mesma posição dentro da seqüência, isto é, que está alinhado com o primeiro.

```
molde:  MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKSELDKAIGRNTN
alvo :  MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKSELDKAIGRNCN

molde:  GVITKDEAEKLFNQDVDAAVRGILRNAKMKPMYDSMDAVRRAAMMNMMFQMGETR
alvo :  GVITKDEAEKLFNQDVDAAVRGILRNAKLKPVYDSLDAVRRCALINMVFQMGETG

molde:  MAGFTNSMRMMQQKRWDEAAVNMAKSRWYNQTPNRAKRVITTFRTGTWDAYK
alvo :  VAGFTNSLRMLQQKRWDAAAAALAAAAWAAATPNRAKRVITTFRTGTWDAYK
```

Além da troca de resíduos, este método realiza inserções e deleções de resíduos nas posições adequadas e realiza cálculo de energia para fazer as operações sobre a estrutura molde chegando a uma estrutura alvo com a menor energia possível. O método utiliza em todo seu processo bibliotecas de *rotamer*, para auxiliar na montagem da estrutura alvo.

Durante todo o processo de construção da estrutura tridimensional da seqüência alvo, a classe Mutate, através de seus métodos de construção como o **actmutate0**, utiliza, entre outras informações, a estrutura secundária da estrutura molde, armazenada no atributo dssp desta classe e transferida para esse atributo na fase de preparação da predição, sendo recalculada em vários pontos da etapa de criação e calculada também no inicio da preparação para construção. Esses dados influenciam na estrutura final da proteína, pois esta é constituída: pelas ligações simples entre os resíduos, formando sua estrutura primária; pelas ligações de pontes de hidrogênio, formando sua estrutura secundária; e pelas demais ligações como Pontes Salinas, Pontes Dissulfídias, ligações CIS Peptídio, formando como um todo a estrutura tridimensional total da proteína.

Após a etapa de criação da estrutura alvo e sua otimização, a ferramenta Nest, realiza a composição de estruturas, caso a entrada do alinhamento para a ferramenta tenha uma composição de moldes. Em seguida, faz a combinação dessas estruturas compostas.

Finalmente, o processo é finalizado com uma verificação da estrutura final resultante que é escrita em um arquivo de saída no formato PDB.

4.2.3 Reengenharia da Nest gerando a Nest2

Ao analisar o processo seguido pela ferramenta Nest para predição da estrutura alvo, verifica-se que os dados de estrutura secundária são calculados em vários pontos dentro do processo, gerando uma certa perda de informação sobre a estrutura secundária. Esta perda corresponde a resíduos da seqüência que deveriam ser identificados como participantes de estruturas secundárias, porém através dos cálculos esses resíduos não são identificados como participantes de nenhuma estrutura secundária. Este trabalho propõe um processo de predição de estrutura de proteína por homologia mais rico em conhecimento, onde dados da estrutura secundária molde anotados por uma ontologia são utilizados, levando a melhorias no desempenho do processo.

Ao determinar quais dados anotados pela ontologia devem ser inseridos no processo de predição, é necessário definir em que parte do processo os dados anotados pela ontologia devem ser inseridos para obter o reuso de conhecimento baseado em ontologia.

Como dito na Seção 2.2.1, a ferramenta Nest implementa os passos 3 e 4 do processo de predição de estrutura de proteína por homologia, que correspondem à construção de um modelo da proteína alvo e à avaliação do modelo, respectivamente. Dentro do passo de construção na ferramenta Nest, nota-se a utilização da estrutura secundária em certos pontos, sendo esta calculada em vários desses pontos, como na preparação para a predição, troca (mutação) de resíduos e em outros. A estrutura secundária é ainda calculada na fase de otimização da estrutura sendo gerada.

Baseados nessa análise, define-se então que nos pontos onde ocorre o cálculo da estrutura secundária da proteína, devem ser inseridos os dados anotados pela ontologia, para que o processo obtenha o ganho desejado.

Para explicar melhor como os dados anotados devem ser inseridos na ontologia vemos a necessidade de mostrar como os dados de estrutura secundária são trabalhados na ferramenta Nest.

A estrutura secundária dentro da ferramenta é classificada com as letra 'h' e 'e' representando *alpha-helix* e *beta-sheet*, respectivamente, e '-' quando não é nenhuma das duas. Essa classificação é feita para cada resíduo da seqüência molde e conseqüentemente em cada resíduo da seqüência alvo, por ser construída com base

na estrutura molde. A seguir um exemplo da forma como a estrutura secundária é classificada na ferramenta. No exemplo temos um alinhamento de entrada com as seqüências molde e alvo e a classificação. O caracter '-' que aparece na seqüência alvo representa *gap* no alinhamento.

```
est sec:  ----eee------eeee-e--eeee----eeee-eeeee---eeeeeeee------
s molde:  CPTLGEAVTDHPDRLWAWEKFVYLDEKQHAWLPLTIEIKDRLQLRVLLRREDVVLG
s alvo :  ---AGEDVGAPPDHLWVHQEGIYRDEYQRTWVAVVEEETSFLRARV--QQIQVPLG

est sec:  ----hhhh-------eeee-----eee----eeee-eeeee--eeeeeeeee---
s molde:  RPMTPTQIGPSLLPIMWQLYPDGRYRSSDSSFWRLVYHIKIDGVEDMLLELLPDD
s alvo :  DAARPSHLLTSQLPLMWQLYPEERYMDNNSRLWQIQHHLMVRGVQELLLKLLPDD
```

No exemplo anterior existem gaps na seqüência alvo onde na estrutura molde existem resíduos participantes de estrutura *beta-sheet*, mostrados em vermelho no exemplo. Durante o processo de predição na Nest, com a retirada dos *gaps* a estrutura secundária para a seqüência alvo fica da forma a seguir, a parte vermelha mostra a área onde ocorre a mudança contendo *beta-sheet*:

```
est sec:  -eee------eeee-e--eeee----eeee-eeeee---eeeeee------
s alvo :  AGEDVGAPPDHLWVHQEGIYRDEYQRTWVAVVEEETSFLRARVQQIQVPLG

est sec:  ----hhhh-------eeee-----eee----eeee-eeeee--eeeeeeeee---
s alvo :  DAARPSHLLTSQLPLMWQLYPEERYMDNNSRLWQIQHHLMVRGVQELLLKLLPDD
```

Para realizar então a predição da estrutura tridimensional com reuso de conhecimento baseado em ontologia, é feito um casamento entre as informações de resíduo da seqüência molde, e consequentemente da seqüência alvo por existir o alinhamento entre elas, e os dados de estrutura secundária da estrutura molde anotados com a ontologia pelos conceitos *HelixStructure* e *Sheets*. Tendo, desta forma, a montagem da estrutura alvo. É claro que este casamento é feito em vários pontos do processo, pois como foi dito anteriormente a estrutura secundária é calculada em

vários pontos do processo.

Para implementar esse casamento no processo, é necessário criar uma interface entre a ferramenta Nest construída em C++, uma linguagem orientada a objeto, e arquivos no formato XML onde encontram-se os dados anotados com a ontologia no padrão OWL. Para isso, foi utilizada a biblioteca libxml2 [20], que contém funções de interface entre C++ e formato XML. A libxml2 é uma biblioteca de funções para manipular dados em XML e trata o arquivo XML como uma estrutura contendo uma árvore de elementos.

Como os métodos para calcular a estrutura secundária na ferramenta Nest encontram-se na classe Chn, pois esta contém a lista de resíduos da proteína, devemos criar então os métodos implementando o reuso de conhecimento baseado em ontologia dentro desta mesma classe. Além disso, todos os pontos onde os métodos para o cálculo são utilizados são modificados para usar os novos métodos criados. Com isso a ferramenta passa a não mais calcular a estrutura secundária e sim reusar dados de estrutura secundária anotados com a ontologia. Este cálculo feito pela ferramenta Nest tem como objetivo definir que resíduos participam de estruturas secundárias e de que tipo (*Helix* ou *Sheet*) são essas estruturas, além de determinar as ligações químicas existentes entre os átomos dos resíduos que compõem a proteína. Com os dados anotados com a ontologia essas informações de estruturas secundárias já podem ser reusadas pela ferramenta para a definição direta das estruturas secundárias existentes na proteína e que resíduos fazem parte delas. Isso é feito através do casamento entre resíduos da seqüência e os resíduos da estrutura anotada.

Os principais métodos criados para o reuso baseado em ontologia são parseDoc e buscaEstSec. A Figura 4.5 mostra parte do processo de predição na ferramenta Nest2. O método parseDoc, que utiliza a libxml2 para interface com o arquivo XML, é responsável por casar os tipos de estrutura secundária com as existentes na estrutura molde e recuperar dados de estrutura secundária anotados com a ontologia através da chamada dos métodos parseHelix e parseStrand, gerando como resultado uma lista com todos os dados recuperados. O método buscaEstSec é responsável por realizar o casamento entre os resíduos da proteína e os dados recuperados para classificar os resíduos, determinando se os resíduos pertencem ou não a alguma estrutura

secundária e qual é esta estrutura, *Helix* ou *Sheet*.

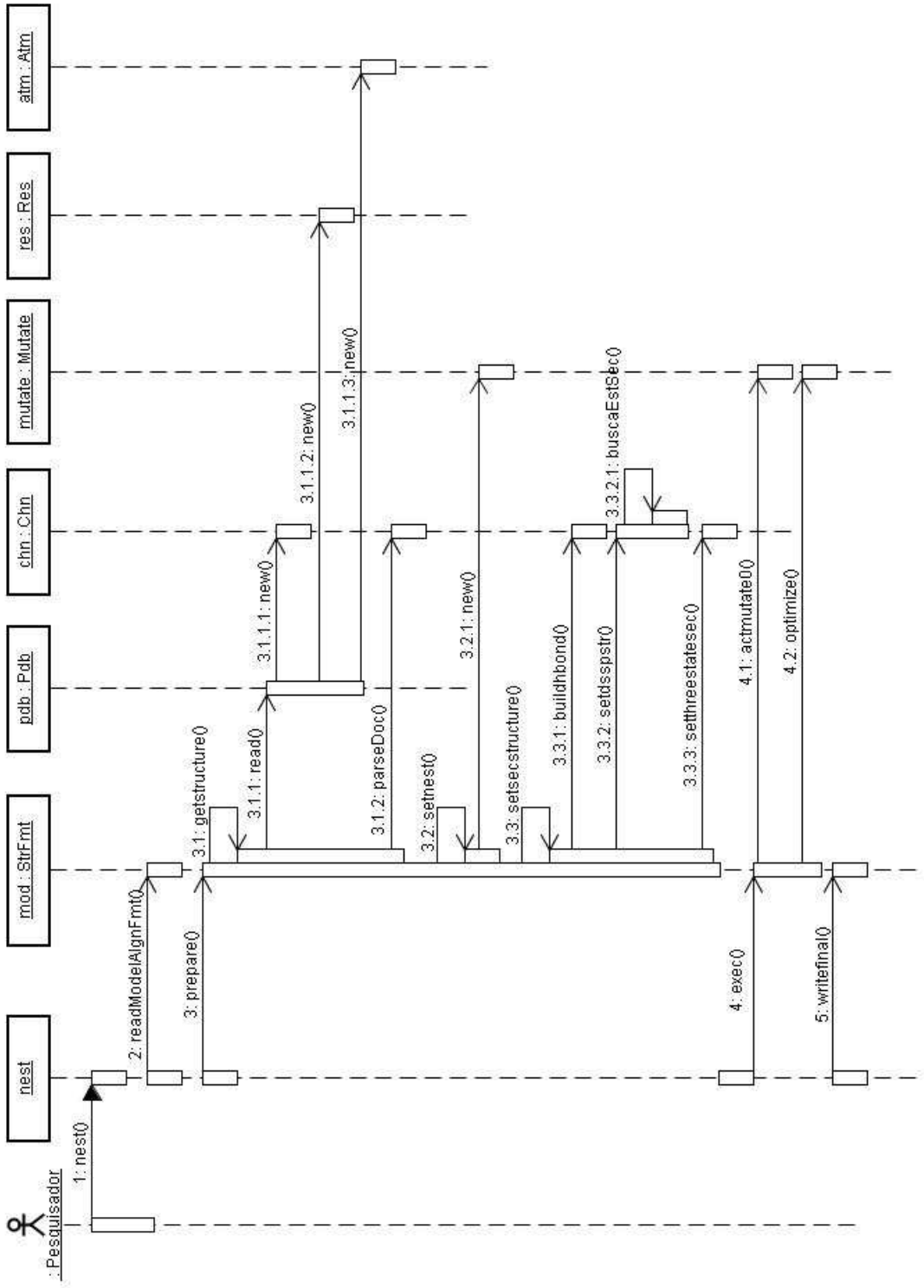

Figura 4.5: Diagrama de seqüência contendo parte do processo de predição na Nest2

Para criar a interface e realizar a comunicações com o arquivo XML, o método parseDoc primeiramente cria um variável do tipo xmlDocPtr que servirá como um ponteiro para a estrutura contendo a árvore de elementos do arquivo XML. Esse

ponteiro passa a apontar para a estrutura no momento em que o arquivo XML é carregado pela função xmlParseFile, que tem como parâmetro o nome do arquivo XML, e retorna um ponteiro para a estrutura. Após carregar o arquivo, o método parseDoc recupera então o elemento raiz da árvore e atribui a uma variável chamada cur, do tipo xmlNodePtr, que corresponde a um ponteiro para um elemento (nó) da árvore. Em seguida percorre a árvore através dos comandos cur->xmlChlindrenNode que recupera o primeiro nó filho do nó corrente e cur->next que recupera o próximo nó irmão do nó corrente. Para recuperar o conteúdo de um determinado elemento é usada a função xmlNodeListGetString e, quando esse conteúdo não é mais necessário, o método utiliza a função xmlFree para liberar a memória usada por esse conteúdo recuperado. Após finalizar suas operações com o arquivo XML, o método libera o arquivo com a função xmlFreeDoc.

Seguindo a seqüência de passos do diagrama mostrado na Figura 4.5, o processo começa com uma requisição do pesquisador tendo como entrada o arquivo de alinhamento, o arquivo pdb, contendo a estrutura molde, e o arquivo owl, gerado pela ferramenta anotaPO, contendo os dados da estrutura molde anotados com a *Protein Ontology*. A ferramenta Nest2 realiza então a leitura do arquivo de alinhamento através do método (readModelAlgnFmt). Em seguida prepara os dados para o processo de predição, através do método **prepare**, descrito na Seção 4.2.3, recuperando a estrutura molde do arquivo pdb. Após esse passo é chamado o método **parseDoc**, que realiza um *parse* no arquivo contendo os dados anotados com a PO, realizando um casamento entre os tipos de estruturas secundárias existentes e as estruturas secundárias da estrutura molde, gerando como resultado uma lista com os dados recuperados. Esses dados são usados posteriormente na definição das estruturas secundárias da seqüência alvo através do método **buscaestSec**.

Após a chamada ao **parseDoc** é preparada a classe Mutate onde a construção da estrutura alvo é realizada, através do método **setnest**. Em seguida é setada, num primeiro momento, a estrutura secundária da seqüência alvo, começando com cálculo das pontes de hidrogênio existentes e depois a determinação da estrutura secundária da seqüência alvo através do método textbfbuscaEstSec. O **buscaEstSec** realiza o casamento entre os resíduos da proteína e os dados anotados com a PO, recuperados pelo método **parseDoc** e desta forma classifica os resíduos quanto ao tipo de estru-

tura secundária participante. Este método, juntamento com o parseDoc, substituem o cálculo realizado pela ferramenta Nest para determinar a estrutura secundária da estrutura alvo. Após esse passo ocorre a execução da lista de operações de mutação, a evolução artificial citada na Seção 2.2.1, para se alcançar a estrutura alvo final, que seja mais próxima da estrutura nativa. Isso é feito através do método **exec** e os métodos chamados por ele, como **acmutate0** e **optimize**. Durante esse processo os métodos que implementam o reuso de conhecimento baseado em ontologia são utilizados novamente substituindo os cálculos realizados durante o processo para determinação da estrutura secundária. O processo termina então com a escrita do arquivo final no formato pdb contendo a estrutura alvo final.

Resumindo as principais diferenças entre a Nest e a Nest2, temos os cálculos para determinação dos dados de entrutura secundária, que são realizados em vários pontos na Nest, sendo substituídos pelos dados anotados com a ontologia na Nest2. Esses cálculos realizados pela Nest acarretam em perda de informação, pois resíduos que deveriam ser identificados como participantes de estruturas secundárias não são. Na Nest2 tem-se uma definição direta das estruturas secundárias com a utilização de dados anotados com a PO e com isso uma melhoria no desempenho e redução da perda de informação.

A Figura 4.6 mostra a instânciação da Figura 4.2 para o mecanismo da Nest2. Nela podemos ver os dados do PDB como o conjunto de dados sendo anotados pela *Protein Ontology*. Estes dados anotados pela *Protein Ontology* são utilizados pelo mecanismo de modelagem da Nest2 para gerar a estrutura da proteína alvo, tendo como base a estrutura molde sendo reusada.

Diferentemente do que foi implementado em [3], onde o processo de reuso é realizado em prolog, que facilita o casamento de informação, no caso do reuso por ontologia no processo de predição, a técnica tem que ser adaptada para realizar de forma mais eficiente possível esse casamento de informações em linguagem orientada a objeto. Por isso, a adaptação da técnica para o processo de predição de estrutura não pode ser feito de forma trivial, tendo que ser adequado a tecnologia usada pela ferramenta Nest e a ontologia PO.

Após a modificação na ferramenta, criando então a ferramenta Nest2, foram realizados testes e comparações entre os resultados da predição fornecidos por ambas

as ferramentas. Esta avaliação será descrita no Capítulo a seguir.

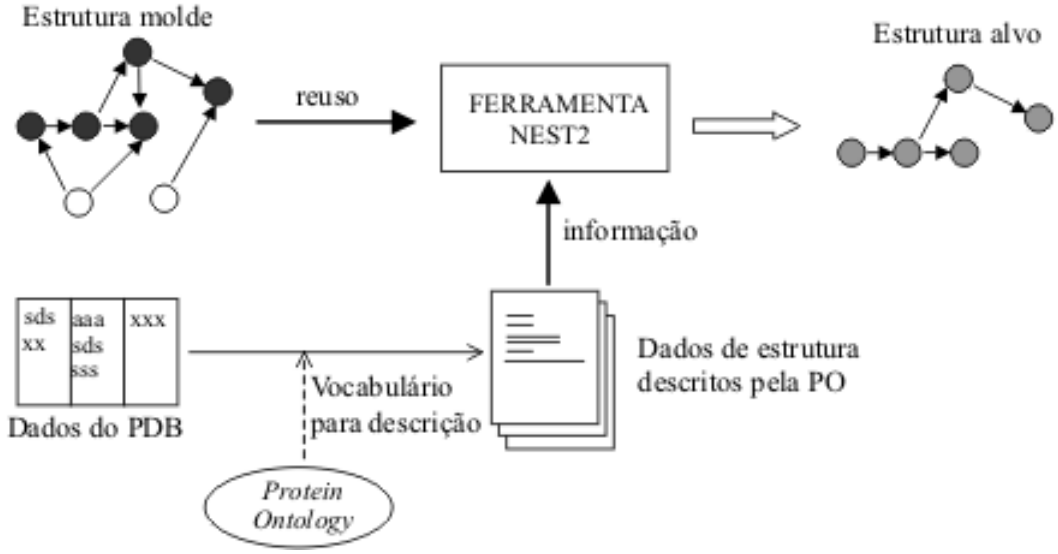

Figura 4.6: Mecanismo de modelagem da Nest2

Capítulo 5

Avaliação da ferramenta Nest2

No Capítulo anterior foi mostrado como a *Protein Ontology* é aplicada na ferramenta de predição Nest, para implementar o reuso de conhecimento baseado em ontologia, gerando a ferramenta Nest2. Nosso objetivo neste Capítulo é mostrar os testes realizados com a Nest e Nest2, comparando as duas ferramentas, e a avaliação dos resultados destes testes. Estes testes correspondem à execução das ferramentas Nest e Nest2, para comparação de tempo de execução entre elas e comparação das estruturas geradas com as duas ferramentas.

A hipótese experimental deste trabalho é que a Nest2, com o reuso de conhecimento baseado em ontologia, tenha ganho de desempenho (tempo de execução) em relação a Nest e um ganho de qualidade nas estruturas geradas.

Para realizar os testes foram criados *scripts* (arquivos .sh) contendo as chamadas de execução da ferramenta de predição para cada alinhamento de entrada, registrando para cada execução da ferramenta o horário de início e término de execução, através de uma função do sistema operacional, tanto com a ferramenta Nest quanto com a ferramenta Nest2. Com esse registro de horários, foi calculado o tempo de execução de cada predição realizada pela Nest e pela Nest2. Em seguida foi feita a comparação entre o tempo de execução de cada predição de estrutura da seqüência alvo na Nest2 com o tempo de predição da mesma seqüência alvo na Nest. Com essa comparação foi registrado se a Nest2 executou cada predição num tempo menor, maior ou igual à Nest. O computador usado para executar os *scripts* e realizar a avaliação foi um NoteBook HP AMD Athlon(tm) XP2500+, 519 Mhz, 192MB de RAM, com sistema operacional Linux Red Hat.

Os alinhamentos de entrada selecionados para a realização dos testes foram os mesmos utilizados no artigo [18], que realizou uma avaliação de várias ferramentas colocando a Nest como umas das três melhores. Estes alinhamentos são de proteínas de uma mesma família. Foi selecionada uma amostra de 290 alinhamentos entre os usados pelo artigo [18], porém, como 4 arquivos de alinhamento estavam com problema, somente 286 alinhamentos foram usados nos testes.

5.1 Avaliação de desempenho

Avaliando os resultados dos testes, em relação ao desempenho, verificamos que a Nest2 teve um tempo de execução menor em 48% da amostra de 286 seqüências alvo, onde 45 estruturas foram geradas pela Nest2 em tempo maior que a Nest, 103 foram geradas em tempo igual e 138 em tempo menor, resultando no gráfico mostrado na Figura 5.1.a. Todos os tempos de execução medidos nos testes podem ser vistos na Tabela do Apêndice B onde encontram-se os nomes dos arquivos de alinhamento, os tempos de execução da Nest, da Nest2, e indicações de tempo de execução menor, maior ou igual da Nest2 em relação à Nest.

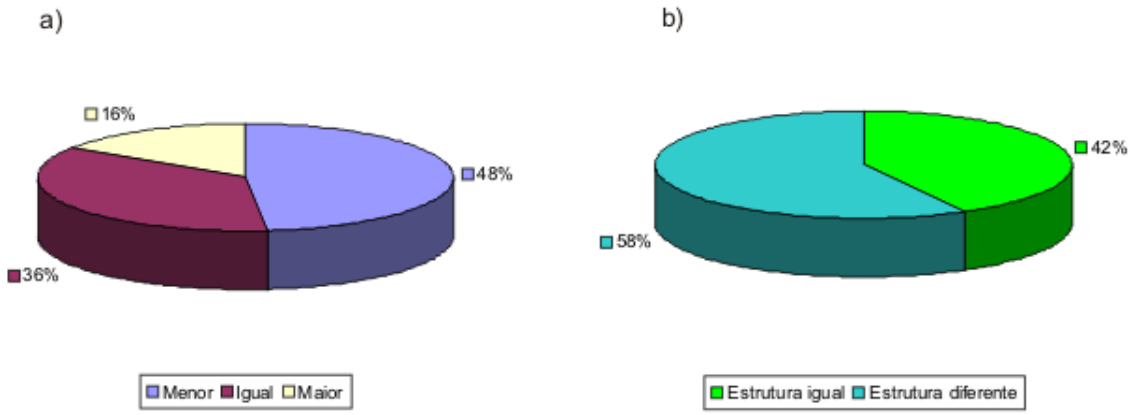

Figura 5.1: a) Gráfico percentual comparativo de tempo de execução da Nest2 em relação à Nest b) Gráfico de comparação entre estruturas geradas pela Nest2 e Nest

O gráfico na Figura 5.1.b, mostra que 42% das 286 seqüências da amostra tiveram estruturas geradas pela Nest2 iguais às estruturas geradas pela Nest e, 58% tiveram estruturas diferentes. Dentre as estruturas diferentes geradas pela Nest2, podem existir estruturas com qualidade inferior ou superior às geradas pela Nest. Fizemos então uma avaliação de desempenho (tempo de execução) considerando

apenas as estruturas iguais, por ser mais acurada a comparação entre tempos de execução na geração, pelas duas ferramentas, de estruturas iguais. Como resultado, tivemos, das 120 estruturas iguais, 8 geradas pela Nest2 em tempo maior que a Nest, 68 geradas em tempo igual e 48 geradas em tempo menor, resultando no gráfico mostrado na Figura 5.2, onde podemos ver que 40% das estruturas iguais foram geradas em tempo menor pela Nest2. Este gráfico também nos mostra que, para as estruturas iguais, a ferramenta Nest2 realizou a predição de estruturas de proteínas com tempo igual ou menor do que a Nest em 93% da amostra.

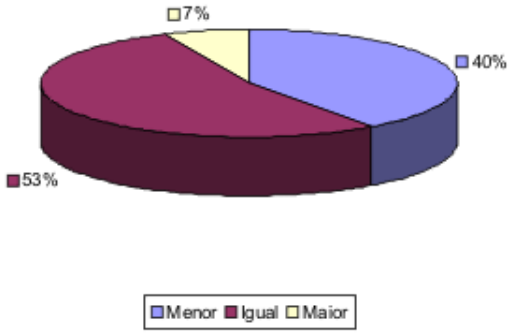

Figura 5.2: Gráfico percentual comparativo de tempo de execução da Nest2 em relação a Nest para estruturas iguais

Este comparativo entre as ferramentas Nest e Nest2, onde a segunda inclui dados anotados por uma ontologia do domínio de estrutura de proteína no processo de predição, demonstra que agregar a ontologia ao processo leva a uma melhoria de desempenho em termos de tempo de execução.

Quanto às estruturas diferentes geradas pela Nest2, isso ocorre por fatores como perda de informação. Como dito no capítulo anterior, nós observamos que em relação à estrutura secundária ocorre uma certa perda de informação quando a ferramenta Nest calcula a estrutura secundária ao invés de reusar o conhecimento sobre a estrutura secundária da estrutura molde. Com a aplicação do reuso de conhecimento baseado em ontologia, através da utilização dos dados de estrutura secundária anotados pela ontologia, a Nest2 gera estruturas diferentes com possíveis mudanças na qualidade das estruturas geradas em relação àquelas da Nest. Para as estruturas iguais, citadas anteriormente, a qualidade delas é a mesma.

Das 166 estruturas diferentes, correspondendo a 58% do total da amostra, 37

foram geradas pela Nest2 em tempo maior que a Nest, 39 foram geradas em tempo igual e 90 foram geradas em tempo menor, resultando no gráfico mostrado na Figura 5.3.

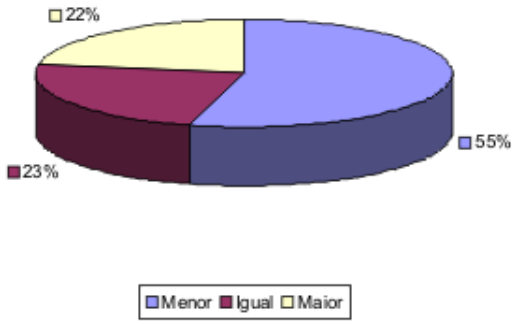

Figura 5.3: Gráfico percentual comparativo de tempo de execução da Nest2 em relação à Nest para estruturas diferentes

Verificando o tempo de cada execução da Nest e Nest2, calculando o $\triangle$t para cada execução e tirando a média geral dos $\triangle$t temos como resultado que a Nest2 realiza a predição em média 0,81s mais rápido que a Nest. Vale ressaltar que a maioria das predições foram realizadas tanto pela Nest quanto pela Nest2 em segundos, pelo fato das estruturas selecionadas para avaliação não serem grandes/complexas. A predição que levou mais tempo para ser realizada, dentre a amostra utilizada, finalizou em 26m31s com a Nest e com a Nest2 finalizou em 23m54s, como mostra a Tabela do Apêndice B, tendo uma diminuição de 2m37s no tempo de execução da Nest2 em relação a Nest.

5.2 Avaliação de qualidade

O critério de qualidade considerado para uma estrutura resultante de um predição é sua similaridade em relação á estrutura nativa, a estrutura em que a proteína realiza sua função na célula. Em [18] foram usadas as medidas RMSD e MaxSub para avaliar a qualidade das estruturas resultantes em relação às estruturas nativas.

O RMSD (*Root Mean Square Deviation*) é a medida mais comum para verificação de similaridade estrutural entre um modelo e uma estrutura correta (a

estrutura nativa). Mede a distância entre dois átomos equivalentes, depois de uma superposição das estruturas molde e nativa. Quanto menor o desvio, melhor a estrutura [18]. O CA-RMSD é a media sobre os átomos Cα.

O MaxSub calcula um escalar no intervalo de 0 a 1 que mede a similaridade de um modelo e sua estrutura correta (nativa) correspondente, determinada através de experimentos, onde 0 indica um modelo completamente errado e 1, um modelo perfeito [16]. Dessa forma, quanto maior o valor de MaxSub, melhor a estrutura.

Para avaliar a qualidade das estruturas, calculamos os valores RMSD de cada estrutura resultante em relação a estrutura nativa, tanto para as estruturas geradas pela Nest2 quanto para as geradas pela Nest. Após esses cálculos foi feita a comparação de cada valor para cada estrutura, registrando se a Nest2 gerou uma estrutura mais similar à estrutura nativa do que a Nest, se gerou uma estrutura igual, ou menos similar. Esse registro pode ser visto na Tabela do Apêndice B. Assim como em [18], foi usado o CA-RMSD. Além dos valores de CA-RMSD, foi calculado também o MaxSub de cada estrutura pois, como dito em [18], o RMSD sozinho não é suficiente para determinar a qualidade de uma estrutura, sendo o MaxSub utilizado como uma informação complementar. O resultado dos cálculos de MaxSub e RMSD, após a comparação entre os valores da Nest e Nest2 podem ser vistos na Tabela do Apêndice B.

Analisando os resultados dos cálculos de RMSD e MaxSub, sobre as 166 estruturas diferentes, e levando em consideração que quanto menor o desvio (valor do RMSD) e maior o MaxSub melhor a qualidade da estrutura, verificamos:

- 51 estruturas geradas pela Nest2 com qualidade igual as das estruturas correspondentes geradas pela Nest, dado que os valores de RMSD e MaxSub foram iguais aos das estruturas correspondentes geradas pela Nest.

- 38 estruturas geradas pela Nest2 com qualidade inferior as das estruturas geradas pela Nest, dado que para 26 destas o desvio aumentou e o MaxSub diminuiu e para demais 12 estruturas o MaxSub manteve-se, porém houve aumento do desvio.

- 23 estruturas de difícil análise de qualidade, pois destas 14 tiveram diminuição do desvio e diminuição do MaxSub e 9 tiveram aumento do desvio e aumento

do MaxSub.

- 54 estruturas geradas pela Nest2 com qualidade superior as das estruturas geradas pela Nest, dado que para 28 destas o desvio diminuiu e o MaxSub aumentou e para as demais 26 estruturas o MaxSub manteve-se, mas houve diminuição do desvio.

Com isso, temos como resultado de qualidade os percentuais mostrados no gráfico da Figura 5.4.a. Se considerarmos a parte duvidosa como perda da qualidade, temos os percentuais mostrados na Figura 5.4.b.

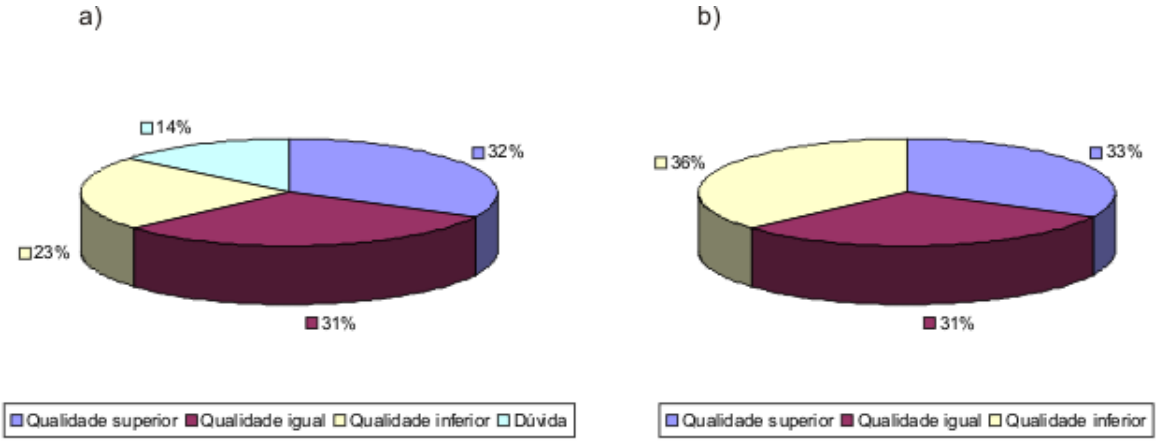

Figura 5.4: Gráficos percentuais comparativos de qualidade da Nest2 em relação a Nest para as 166 estruturas diferentes

5.3 Avaliação geral dos resultados

Analisando os tempos e os resultados de medidas de qualidade das 166 estruturas diferentes, verificamos que 59 dessas estruturas, sendo 33 de qualidade igual e 26 de qualidade superior, tiveram o tempo de execução menor. Unindo esta informação com a informação inicial das 48 estruturas iguais com tempo menor, temos então 107 estruturas que podem ser realmente aceitas com tempo menor, o que nos dá 38% do total da amostra, que é de 286 estruturas. O gráfico da Figura 5.5 mostra um resumo geral das análises e percentuais calculados, onde pode ser visto os 38% das estruturas que podem ser realmente aceitas com tempo menor.

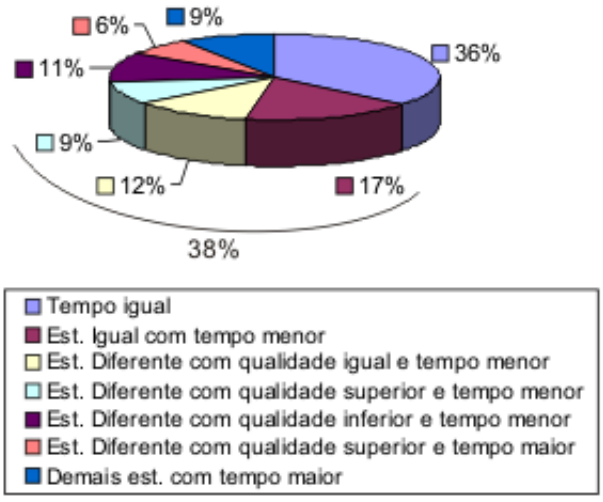

Figura 5.5: Gráfico percentual comparativo de tempo e qualidade da Nest2 em relação a Nest (resumo)

Capítulo 6

Discussão, conclusões e trabalhos futuros

Quando nos propusemos a realizar este trabalho tínhamos como meta mostrar a aplicabilidade da técnica usada em [3] no domínio ecológico, de reuso de conhecimento baseado em ontologia, no domínio da biologia molecular, tendo com isso um ganho de desempenho, porém realizando as devidas adaptações para o domínio em questão. Acrescentamos a ontologia *Protein Ontology* no processo de predição de estrutura de proteína realizado pela Nest, gerando a Nest2. O acréscimo foi feito na etapa de definição da estrutura secundária da estrutura alvo. Antes da inclusão da ontologia no processo a ferramenta Nest realizava cálculos para definir os resíduos participantes de estruturas secundárias e de qual tipo seriam essas estruturas (*beta-sheet* ou *alpha-helix*). Com o acréscimo da ontologia no processo esses cálculos foram retirados e os dados de estrutura secundária anotados com a PO foram utilizados para definir os resíduos participantes de estruturas secundárias. Verificando os resultados obtidos no capítulo anterior, podemos confirmar que a meta foi alcançada, mostramos a aplicabilidade da técnica no domínio da biologia molecular, tendo ganho de desempenho. Outra meta alcançada por nós neste trabalho foi a de tornar explícito o conhecimento sobre o processo de predição de estrutura de proteína por homologia.

6.1 Discussão

O uso da ontologia no processo de predição facilitou a obtenção de informações utilizadas no processo. Como os cálculos realizados para obter informações classificativas de estruturas secundárias foram retirados do processo e os dados anotados com a PO foram utilizados para obtenção dessas informações classificativas, ocorreu ganho de performance no processo de predição, onde 38% da amostra teve o modelo de estrutura gerado pela Nest2 em tempo menor que a Nest. Além disso foi observado um ganho de qualidade das estruturas geradas, pelo fato de que antes ocorria perda de informação na definição das estruturas secundárias com os cálculos realizado pela Nest. Vemos então a substituição da informação procedimental, onde tínhamos os cálculos, pela informação declarativa, os dados anotados com a PO, proporcionando ganho de eficiência e qualidade no processo de predição de estrutura de proteína por homologia.

Esse ganho de eficiência e qualidade tem grande relevência nesse processo de predição, pois pesquisadores envolvidos no processo de descobrir as estruturas das proteínas trabalham com uma grande massa de dados, onde esses dados precisam ser processados num menor tempo possível e com melhor qualidade dos resultados. Pois os resultados desse processamento são depois avaliados para obtenção de informações importantes como a função da proteína dentro da célula, que auxiliam os pesquisadores na criação de drogas para combater doenças.

Em se tratando da avaliação da qualidade das estruturas geradas, é necessário ampliar o estudo dos resultados, usando outras medidas de avaliação de similaridade. Tivemos ganho de qualidade comprovado para 18% da amostra de seqüências e possivelmente perda de qualidade para uma porção equivalente da amostra. O que poderia diminuir esta possível perda de qualidade (e aumentar o ganho), seria o acréscimo de outros dados anotados com a ontologia, como, por exemplo, dados de pontes de hidrogênio que estão relacionados aos dados anotados de estrutura secundária.

Tivemos várias dificuldades no decorrer deste trabalho. Não obtivemos acesso ao código fonte de nossa 1a. escolha de ferramenta de predição de estrutura de proteína. Avaliamos algumas antes porém não eram de código aberto, até que conseguimos acesso à Nest, considerada uma das melhores na avaliação em [18].

Além disso, o conhecimento que tínhamos em biologia molecular era pouco, exigindo-nos um esforço ainda maior na pesquisa para adquirir o conhecimento em biologia molecular necessário para a realização deste trabalho.

6.2 Contribuições

Temos como contribuição o processo proposto de predição por homologia baseado em ontologia, onde a *Protein Ontology* é aplicada no processo de predição por homologia resultando no ganho de desempenho. A área de Biologia Molecular necessitada de nossos conhecimentos em informática para automatizar e melhorar seus processos, como o processo de predição, que tem um custo alto quando realizado por experimentos laboratoriais.

Outra contribuição deste trabalho foi tornar explícito o conhecimento sobre o processo de predição de estrutura de proteína por homologia, onde estruturas de proteínas já conhecidas são utilizadas como molde pelo processo para gerar a estrutura de proteína alvo. Todas as estruturas que possuem similaridade de seqüência com a sequüência da estrutura alvo são selecionas e informações como ligações químicas e estruturas secundárias dessas proteínas são usadas pelo processo para chegar na estrutura alvo, a estrutura mais próxima da estrutura nativa. Este processo foi tornado explícito com a reengenharia realizada na ferramenta Nest para gerar a Nest2, onde foi feito todo um estudo e análise do processo para acrescentar a ontologia ao mesmo.

Além das contribuições citadas anteriormente temos também como contribuíção deste trabalho a ferramenta anotaPO criada para anotar os dados de proteína com a *Protein Ontology*. A anotaPO recebe dados da base PDB e anota esses dados com a ontologia *Protein Ontology* gerando um arquivo com os dados em OWL.

6.3 Trabalhos futuros

Como trabalhos futuros, propomos uma melhoria na ferramenta anotaPO, para que esta possa realizar a anotação de dados oriundos das outras bases nas quais a construção da *Protein Ontology* se baseou, pois a anotaPO realiza a anotação de dados oriundos apenas do PDB; uma análise mais detalhada da qualidade das

estruturas geradas pela Nest2; e a utilização de mais dados anotados com a PO no processo de predição de estrutura, como os dados de ligações químicas.

Referências Bibliográficas

[1] Anfinsen, C. B. Principles that govern the folding of protein chains. Science 181 (1973), 223–230.

[2] Bajorath, J., Stenkamp, R., and Aruffo, A. Knowledge-based model building of proteins: Concepts and examples. Protein Science 2 (1993), 1798–1810.

[3] Brilhante, V. Ontology and Reuse in Model Synthesis. Ph.d. thesis, School of Informatics, University of Edinburgh, 2003.

[4] Brilhante, V. Ontology-enabled reuse of structural data-based model. In Workshop on Ontologies and their Applications (WONTO'04) at the 17th Brazilian Symposium on Artificial Intelligence (São Luís, Brazil, 2004).

[5] Brilhante, V. Ecolingua: a formal ontology for data in ecology. Journal of the Brazilian Computer Society 11, 2 (2005), 61–78.

[6] de Brito, E. A. Biologia: volume único, 1 ed. Moderna, São Paulo, 1999.

[7] Gutwin, K. A parallel implementation of a higher-order self consistent mean field. May 2005.

[8] Madhusudhan, M. S., Martí-Renom, M. A., Eswar, N., John, B., Pieper, U., Karchin, R., Shen, M.-Y., and Sali, A. Comparative protein structure modeling. The Proteomics Protocols Handbook, 831–860.

[9] Martí-Renom, M. A., Stuart, A. C., Fiser, A., Sánchez, R., Melo, F., and Sali, A. Comparative protein structure modeling of genes and genomes. Annu. Rev. Biophys. Biomol. Struct 29 (2000), 291–325.

[10] Meidanes, J., and Setúbal, J. C. Uma Introdução à Biologia Computacional. IX Escola de Computação, Recife, 1994.

[11] Pinagé, K., and Brilhante, V. Protein structure homology modelling assisted by ontology. In 14th Annual International Conference On Intelligent Systems For Molecular Biology (ISMB 2006). Poster.

[12] Schmidt, T. S. Backing on strucutre. Bio-IT World 1 (October 2002).

[13] Sidhu, A. S., Dillon, T. S., and Chang, E. Creating a protein ontology resource. CSB Workshops (2005), 220–221.

[14] Sidhu, A. S., Dillon, T. S., and Chang, E. Protein Ontology Guide. BIOMAP Project - EXEL Research Lab, December 2006.

[15] Sidhu, A. S., Dillon, T. S., Chang, E., and Sidhu, B. S. Protein ontology: Vocabulary for protein data. In Third International Conference on Information Technology and Applications (ICITA'05). IEEE Computer Society (2005).

[16] Siew, N., Elofsson, A., Rychlewski, L., and Fischer, D. Maxsub: An automated measure for the assessment of protein structure prediction quality. Bioinformatics 16 (September 2000), 776–785.

[17] Smith, M. K., Welty, C., and McGuinness, D. L. OWL Web Ontology Language Guide. W3C, February 2004.

[18] Wallner, B., and Elofsson, A. All are not equal: A benchmark of different homology programs. Protein Science 14 (2005), 1315–1327.

[19] Xiang, J. Z. JACKAL: A Protein Structure Modeling Package. Columbia University and Howard Hughes Medical Institute, July 2002.

[20] Biblioteca de funções libxml2. http://xmlsoft.org. Acessado em julho de 2007.

[21] Deepview - ferramenta de visualização e manipulação de estruturas de proteínas. http://expasy.org/spdbv. Acessado em julho de 2007.

[22] Protégé - editor de ontologias. http://protégé.standford.edu. Acessado em julho de 2007.

[23] Protein ontolgy. http://www.proteinontology.info. Acessado em julho de 2007.

[24] Rasmol - ferramenta para visualização de estruturas de proteínas. http://www.usmass.edu/microbio/rasmol/index2.html. Acessado em julho de 2007.

[25] Web-ontology (web onto). http://www.w3.org/2001/sw/webont. Acessado em julho de 2007.

Apêndice A

Anotação de dados de proteína com a *Protein Ontology*

Abaixo demais dados anotados com a ferramenta anotaPO mostrada na Seção 4.1.

Descrição do registro da proteína

O PDB guarda também informações referentes ao experimento realizado para obter a estrutura da proteína e sobre as macromoleculas biológicas presentes. Entre as informações sobre o experimento temos uma descrição, onde encontramos: nome dos responsáveis pelas informações da proteína registradas no PDB, um resumo do experimento, descrição da técnica experimental usada e um conjunto de termos relevantes no registro da proteína. Essas informações são descritas no PDB pelos registros identificados pelas siglas AUTHOR, TITLE, EXPDTA, KEYWDS respectivamente. A anotação desses dados com a PO é feita pelo conceito *Description*. Abaixo um exemplo.

```
<Description rdf:ID="DescriptionInstance_13_1">
<Authors rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
D.J.RIGDEN,S.E.V.PHILLIPS,P.A.M.MICHELS,L.A.FOTHERGILL- GILMORE
</Authors>
<Title rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
THE STRUCTURE OF LEISHMANIA PYRUVATE KINASE
</Title>
```

```
<Keywords rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PYRUVATE KINASE, GLYCOLYTIC ENZYME, HOMOTETRAMER, TRANSFERASE
</Keywords>
<Experiment rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
X-RAY DIFFRACTION
</Experiment>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PO_13</ProteinOntologyID>
</Description>
```

Outras informações referentes ao experimento é a identificação do registro da proteína, onde encontramos um código identificador, uma classificação para o registro e a data em que este foi realizado. Essas informações estão descritas no PDB através do registro identificado pela sigla HEADER. Na PO esses dados são anotados com o conceito *Entry*. Abaixo um exemplo.

```
<Entry rdf:ID="EntryInstance_13_1">
<SourceDatabaseID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
1PKL</SourceDatabaseID>
<SourceDatabaseName rdf:datatype="http://www.w3.org/2001/XMLSchema
♯string">PROTEIN DATA BANK</SourceDatabaseName>
<SubmissionDate rdf:datatype="http://www.w3.org/2001/XMLSchema
♯string">15-SEP-98</SubmissionDate>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PO_13</ProteinOntologyID>
</Entry>
```

Além dessas informações já citadas sobre o experimento, o PDB contém também as referências literárias que descrevem o experimento que resultou na estrutura da proteína. Essas referências estão descritas no PDB pelos registros identificados com a sigla JRNL e contém informações como: lista de autores da referência; título da referência; editores; informações da publicação como status, volume, página

e ano; nome da editora e local de publicação, se a referência for de um livro; e referência codificada para citações. Todos esses dados são anotados pelo conceito *Reference* da PO. Abaixo um exemplo.

```
<Reference rdf:ID="RefInstance_13_1">
<CitationAuthors rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
D.J.RIGDEN,S.E.PHILLIPS,P.A.MICHELS, L.A.FOTHERGILL-GILMORE
</CitationAuthors>
<CitationEditors rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
</CitationEditors>
<CitationPublication rdf:datatype="http://www.w3.org/2001/XMLSchema
♯string">
</CitationPublication>
<CitationReference rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
J.MOL.BIOL. V. 291 615 1999 ASTM JMOBAK UK ISSN 0022-2836
</CitationReference>
<CitationReferenceNumbers rdf:datatype="http://www.w3.org/2001/XMLSchema
♯string">
ASTM JMOBAK UK ISSN 0022-2836
</CitationReferenceNumbers>
<CitationTitle rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
THE STRUCTURE OF PYRUVATE KINASE FROM LEISHMANIA
MEXICANA REVEALS DETAILS OF THE ALLOSTERIC
TRANSITION AND UNUSUAL EFFECTOR SPECIFICITY
</CitationTitle>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PO_13</ProteinOntologyID>
</Reference>
```

Quantos às informações da macromolécula temos: cadeias existentes na macromolécula; unidade biológica; indicação se a molécula foi produzida ou não usando tecnologia recombinante; especificação de um domínio ou região da molécula; nome

da molécula; descrição de mutações e uma lista de sinônimos para a molécula. Esses dados estão descritos no PDB através do resgistro identificado pela sigla COMPND. Com a PO esses dados são anotados pelo conceito *Molecule*. Abaixo um exemplo.

```
<Molecule rdf:ID="MoleculeInstance_13_1">
< _Molecule_Chain rdf:resource="♯ChainInstance_13_A"/>
< _Molecule_Chain rdf:resource="♯ChainInstance_13_B"/>
< _Molecule_Chain rdf:resource="♯ChainInstance_13_C"/>
< _Molecule_Chain rdf:resource="♯ChainInstance_13_D"/>
< _Molecule_Chain rdf:resource="♯ChainInstance_13_E"/>
< _Molecule_Chain rdf:resource="♯ChainInstance_13_F"/>
< _Molecule_Chain rdf:resource="♯ChainInstance_13_H"/>
< _Molecule_Chain rdf:resource="♯ChainInstance_13_G"/>
<BiologicalUnit rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
</BiologicalUnit>
<Engineered rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
YES</Engineered>
<Fragment rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
</Fragment>
<MoleculeID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
1</MoleculeID>
<MoleculeName rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PYRUVATE KINASE</MoleculeName>
<Mutations rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
</Mutations>
<OtherDetails rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
</OtherDetails>
<Snonyms rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
</Snonyms>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema♯string">
PO_13</ProteinOntologyID>
</Molecule>
```

Dados de coordenadas e cristalografia

O PDB registra informações de parâmetros de unidade celular, grupo espacial e valor Z da proteína. Esses dados são descritos no registro identificado pela sigla CRYST1. Na PO esses dados são anotados pelo conceito UnitCell. Abaixo um exemplo.

```
<UnitCell rdf:ID="UnitCellInstance_13_1">
<a rdf:datatype="http://www.w3.org/2001/XMLSchema#float">121.649</a>
<b rdf:datatype="http://www.w3.org/2001/XMLSchema#float">132.638</b>
<c rdf:datatype="http://www.w3.org/2001/XMLSchema#float">180.996</c>
<z rdf:datatype="http://www.w3.org/2001/XMLSchema#string">16</z>
<alpha rdf:datatype="http://www.w3.org/2001/XMLSchema#float">
90.00</alpha>
<beta rdf:datatype="http://www.w3.org/2001/XMLSchema#float">97.11</beta>
<gamma rdf:datatype="http://www.w3.org/2001/XMLSchema#float">
90.00</gamma>
<SpaceGroup rdf:datatype="http://www.w3.org/2001/XMLSchema#string"> P 1 21
1</SpaceGroup>
<ProteinOntologyID rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
PO_13</ProteinOntologyID>
</UnitCell>
```

Apêndice B

Tabela de resultados

Neste Apêndice temos uma Tabela contendo todos os resultados dos testes realizados com as ferramentas Nest e Nest2. As colunas na Tabela correspondem nesta ordem, a: o nome dos arquivos de alinhamento usados como entrada para as ferramentas, o tempo de execução para cada alinhamento usando a ferramenta Nest, o tempo de execução para cada alinhamento usando a ferramenta Nest2, e para o mesmo arquivo de alinhamento, uma indicação de a estrutura gerada pela Nest2 ter sido diferente ou igual à gerada pela ferramenta Nest, uma indicação de o valor do RMSD da estrutura gerada pela Nest2 ter sido maior, menor ou igual ao da estrutura gerada pela Nest, e uma indicação de o valor do MaxSub da estrutura gerada pela Nest2 ter sido maior, menor ou igual ao da estrutura gerada pela Nest. Os tempos da Nest2 são indicados em vermelho para tempo maior, azul para tempo igual e verde para tempo menor que o da ferramenta Nest. As indicações D, I, M e N que aparecem na Tabela, correspondem respectivamente à diferente, igual, maior e menor.

	Alinhamento	Nest	Nest2	Est	RMSD	Msub
1	d119l__.d.2.1.3_on_d169la_.d.2.1.3.ali	5s	**5s**	I	I	I
2	d169la_.d.2.1.3_on_d1lpya_.d.2.1.3.ali	5s	**8s**	D	M	N
3	d16pk__.c.86.1.1_on_d1qpg__.c.86.1.1.ali	1m30s	**1m33s**	D	M	N
4	d1a05a_.c.77.1.1_on_d1cnza_.c.77.1.1.ali	5s	**5s**	I	I	I
5	d1a17__.a.118.8.1_on_d1elwa_.a.118.8.1.ali	1s	**1s**	I	I	I
6	d1a1s_1.c.78.1.1_on_d1dxha1.c.78.1.1.ali	12s	**11s**	D	I	I
7	d1a1x__.b.63.1.1_on_d1jsg__.b.63.1.1.ali	7s	**9s**	D	N	N
8	d1a2oa1.c.23.1.1_on_d1dz3a_.c.23.1.1.ali	36s	**33s**	D	N	N
9	d1a2za_.c.56.4.1_on_d1auga_.c.56.4.1.ali	40s	**38s**	I	I	I
10	d1a31a1.d.163.1.2_on_d1k4ta2.d.163.1.2.ali	17s	**14s**	D	I	I
11	d1a3qa1.b.1.18.1_on_d1bfs__.b.1.18.1.ali	4s	**4s**	I	I	I
12	d1a3qa1.b.1.18.1_on_d1gjia1.b.1.18.1.ali	12s	**12s**	I	I	I
13	d1a3wa1.b.58.1.1_on_d1liua1.b.58.1.1.ali	25s	**22s**	D	I	I
14	d1a3wa1.b.58.1.1_on_d1pkla1.b.58.1.1.ali	19s	**19s**	D	I	I
15	d1a3wa2.c.1.12.1_on_d1a49a2.c.1.12.1.ali	31s	**28s**	D	N	I
16	d1a44__.b.17.1.1_on_d1beha_.b.17.1.1.ali	3s	**3s**	I	I	I
17	d1a44__.b.17.1.1_on_d1kn3a_.b.17.1.1.ali	7s	**7s**	I	I	I
18	d1a48_.d.143.1.1_on_d1kutb_.d.143.1.1.ali	3m59s	**4m**	D	N	M
19	d1a49a1.b.58.1.1_on_d1e0ta1.b.58.1.1.ali	21s	**20s**	D	M	N
20	d1a49a1.b.58.1.1_on_d1pkm_1.b.58.1.1.ali	13s	**14s**	I	I	I
21	d1a49a2.c.1.12.1_on_d1pkla2.c.1.12.1.ali	49s	**43s**	D	M	I
22	d1a4ma_.c.1.9.1_on_d1krma_.c.1.9.1.ali	10s	**9s**	D	I	I
23	d1a4ya_.c.10.1.1_on_d2bnh__.c.10.1.1.ali	13s	**13s**	I	I	I
24	d1a58__.b.62.1.1_on_d1lopa_.b.62.1.1.ali	56s	**41s**	D	N	N
25	d1a5da2.b.11.1.1_on_d1ha4a_.b.11.1.1.ali	5s	**4s**	D	I	I
26	d1a5ma_.d.8.1.1_on_d4ubpa_.d.8.1.1.ali	3s	**3s**	I	I	I
27	d1a5mb_.b.85.3.1_on_d4ubpb_.b.85.3.1.ali	4s	**3s**	I	I	I
28	d1a5mc1.b.92.1.1_on_d4ubpc1.b.92.1.1.ali	57s	**58s**	D	N	M
29	d1a5mc2.c.1.9.2_on_d4ubpc2.c.1.9.2.ali	29s	**53s**	D	N	I
30	d1a5z_1.c.2.1.5_on_d6ldh_1.c.2.1.5.ali	9s	**8s**	I	I	I
31	d1a68__.d.42.1.2_on_d1qdva_.d.42.1.2.ali	2s	**2s**	I	I	I
32	d1a6da1.a.129.1.2_on_d1a6db1.a.129.1.2.ali	23s	**25s**	D	N	I
33	d1a6da3.d.56.1.2_on_d1a6db3.d.56.1.2.ali	12s	**12s**	I	I	I
34	d1a6db1.a.129.1.2_on_d1a6da1.a.129.1.2.ali	17s	**15s**	D	I	I
35	d1a6db2.c.8.5.2_on_d1ass__.c.8.5.2.ali	4s	**4s**	I	I	I
36	d1a6i_1.a.4.1.9_on_d2tct_1.a.4.1.9.ali	3s	**3s**	I	I	I
37	d1a75a_.a.39.1.4_on_d5pal__.a.39.1.4.ali	4s	**4s**	I	I	I

	Alinhamento	Nest	Nest2	Est	RMSD	Msub
38	d1a77_1.a.60.7.1_on_d1b43a1.a.60.7.1.ali	9s	**9s**	I	I	I
39	d1a7ge_.d.58.8.1_on_d1by9__.d.58.8.1.ali	6s	**5s**	I	I	I
40	d1a7w__.a.22.1.2_on_d1b67a_.a.22.1.2.ali	2s	**2s**	I	I	I
41	d1a88a_.c.69.1.12_on_d1a8s__.c.69.1.12.ali	12s	**10s**	D	I	I
42	d1a8d_2.b.42.4.2_on_d1epwa2.b.42.4.2.ali	3m57s	**3m53s**	D	N	M
43	d1a8o__.a.28.3.1_on_d2eiaa1.a.28.3.1.ali	4s	**4s**	I	I	I
44	d1a8q__.c.69.1.12_on_d1a8s__.c.69.1.12.ali	12s	**11s**	I	I	I
45	d1aa8a2.d.16.1.3_on_d1c0pa2.d.16.1.3.ali	20s	**19s**	I	I	I
46	d1abrb2.b.42.2.1_on_d2aaib2.b.42.2.1.ali	6s	**6s**	I	I	I
47	d1ads__.c.1.7.1_on_d1afsa_.c.1.7.1.ali	25s	**21s**	D	I	I
48	d1ag8a_.c.82.1.1_on_d1cw3a_.c.82.1.1.ali	16s	**15s**	D	I	I
49	d1agx__.c.88.1.1_on_d4pgaa_.c.88.1.1.ali	15s	**14s**	I	I	I
50	d1ah4__.c.1.7.1_on_d1j96a_.c.1.7.1.ali	16s	**16s**	D	N	I
51	d1ahsa_.b.19.1.1_on_d1bvp12.b.19.1.1.ali	17s	**17s**	D	N	I
52	d1aipc1.a.5.2.2_on_d1efub3.a.5.2.2.ali	9s	**7s**	D	M	I
53	d1aisa1.d.129.1.1_on_d1cdwa2.d.129.1.1.ali	33s	**41s**	D	M	N
54	d1aisa2.d.129.1.1_on_d1cdwa2.d.129.1.1.ali	4s	**4s**	D	N	I
55	d1aj8a_.a.103.1.1_on_d1o7xa_.a.103.1.1.ali	57s	**1m13s**	D	N	M
56	d1ajka_.b.29.1.2_on_d1gbg__.b.29.1.2.ali	4m15s	**3m58s**	D	N	N
57	d1akz__.c.18.1.1_on_d4euga_.c.18.1.1.ali	14s	**14s**	I	I	I
58	d1alva_.a.39.1.8_on_d1gjya_.a.39.1.8.ali	19s	**17s**	D	N	I
59	d1amm_1.b.11.1.1_on_d1a5da1.b.11.1.1.ali	2s	**3s**	I	I	I
60	d1an7a_.d.140.1.1_on_d1seia_.d.140.1.1.ali	22s	**29s**	D	N	M
61	d1an9a2.d.16.1.3_on_d1c0pa2.d.16.1.3.ali	20s	**18s**	I	I	I
62	d1aora2.d.152.1.1_on_d1b25a2.d.152.1.1.ali	48s	**48s**	I	I	I
63	d1aoxa_.c.62.1.1_on_d1ck4a_.c.62.1.1.ali	11s	**10s**	D	I	I
64	d1aqt_1.a.2.10.1_on_d1fs0e1.a.2.10.1.ali	5s	**5s**	I	I	I
65	d1ar1b1.b.6.1.2_on_d1m56b1.b.6.1.2.ali	7s	**7s**	D	I	I
66	d1ar1b1.b.6.1.2_on_d2occb1.b.6.1.2.ali	20s	**18s**	D	I	I
67	d1art__.c.67.1.1_on_d2csta_.c.67.1.1.ali	27s	**27s**	D	N	I
68	d1au7a1.a.4.1.1_on_d1jgga_.a.4.1.1.ali	4s	**4s**	I	I	I
69	d1au7a2.a.35.1.1_on_d1e3oc2.a.35.1.1.ali	3s	**3s**	I	I	I
70	d1aun__.b.25.1.1_on_d1du5a_.b.25.1.1.ali	10s	**14s**	D	N	M
71	d1aun__.b.25.1.1_on_d1kwna_.b.25.1.1.ali	18s	**17s**	D	N	M
72	d1avc_1.a.65.1.1_on_d1ala__.a.65.1.1.ali	16s	**15s**	D	I	I
73	d1avwb_.b.42.4.1_on_d1tie__.b.42.4.1.ali	26s	**23s**	D	N	N
74	d1ayfa_.d.15.4.1_on_d1e9ma_.d.15.4.1.ali	6s	**6s**	I	I	I

	Alinhamento	Nest	Nest2	Est	RMSD	Msub
75	d1ayl_1.c.91.1.1_on_d1j3ba1.c.91.1.1.ali	26s	**24s**	D	I	I
76	d1ayoa_.b.2.4.1_on_d1edya_.b.2.4.1.ali	3s	**3s**	I	I	I
77	d1azpa_.d.9.2.1_on_d1c8ca_.d.9.2.1.ali	5s	**4s**	I	I	I
78	d1azsc1.a.66.1.1_on_d1tada1.a.66.1.1.ali	11s	**11s**	D	N	I
79	d1azwa_.c.69.1.7_on_d1qtra_.c.69.1.7.ali	14s	**12s**	D	I	I
80	d1b04b_.d.142.2.2_on_d1dgsa3.d.142.2.2.ali	56s	**54s**	D	M	N
81	d1b06a1.a.2.11.1_on_d1idsa1.a.2.11.1.ali	19s	**26s**	D	M	N
82	d1b06a2.d.44.1.1_on_d1coja2.d.44.1.1.ali	15s	**18s**	D	I	I
83	d1b09a_.b.29.1.5_on_d1saca_.b.29.1.5.ali	9s	**9s**	D	I	I
84	d1b0aa2.c.58.1.2_on_d1a4ia2.c.58.1.2.ali	10s	**10s**	I	I	I
85	d1b2pa_.b.78.1.1_on_d1bwua_.b.78.1.1.ali	18s	**16s**	I	I	I
86	d1b2pa_.b.78.1.1_on_d1kj1d_.b.78.1.1.ali	21s	**20s**	I	I	I
87	d1b3qa2.b.40.7.1_on_d1b3qb2.b.40.7.1.ali	16s	**13s**	I	I	I
88	d1b41a_.c.69.1.1_on_d1n5ma_.c.69.1.1.ali	28s	**26s**	D	N	I
89	d1b43a1.a.60.7.1_on_d1a76_1.a.60.7.1.ali	33s	**33s**	D	N	M
90	d1b4aa1.a.4.5.3_on_d1f9na1.a.4.5.3.ali	3s	**3s**	I	I	I
91	d1b4aa1.a.4.5.3_on_d1f9nb1.a.4.5.3.ali	2s	**2s**	I	I	I
92	d1b4ba_.d.74.2.1_on_d1f9na2.d.74.2.1.ali	3s	**3s**	I	I	I
93	d1b67a_.a.22.1.2_on_d1a7w__.a.22.1.2.ali	2s	**1s**	I	I	I
94	d1b67a_.a.22.1.2_on_d1f1ea_.a.22.1.2.ali	5s	**6s**	D	I	I
95	d1b74a1.c.78.2.1_on_d1jfla1.c.78.2.1.ali	9s	**8s**	D	N	I
96	d1b7ba_.c.73.1.1_on_d1e19a_.c.73.1.1.ali	21s	**20s**	D	N	M
97	d1b8oa_.c.56.2.1_on_d1g2oa_.c.56.2.1.ali	48s	**58s**	D	M	M
98	d1b9ma3.b.40.6.2_on_d1b9mb3.b.40.6.2.ali	8s	**8s**	I	I	I
99	d1b9ma3.b.40.6.2_on_d1h9ma1.b.40.6.2.ali	11s	**11s**	I	I	I
100	d1baia_.b.50.1.1_on_d1mtra_.b.50.1.1.ali	52s	**46s**	D	N	M
101	d1bcpb1.b.40.2.1_on_d1bcpc1.b.40.2.1.ali	5s	**3s**	I	I	I
102	d1bcpb2.d.169.1.2_on_d1bcpc2.d.169.1.2.ali	3s	**3s**	I	I	I
103	d1bcpc2.d.169.1.2_on_d1bcpb2.d.169.1.2.ali	4s	**4s**	I	I	I
104	d1bdfb1.d.74.3.1_on_d1iw7a1.d.74.3.1.ali	15s	**14s**	I	I	I
105	d1bdqa_.b.50.1.1_on_d1kzka_.b.50.1.1.ali	2s	**2s**	I	I	I
106	d1bed_1.a.44.1.1_on_d1dyva1.a.44.1.1.ali	3s	**3s**	I	I	I
107	d1bed_2.c.47.1.4_on_d1fvka2.c.47.1.4.ali	11s	**10s**	D	N	I
108	d1beha_.b.17.1.1_on_d1kn3a_.b.17.1.1.ali	4s	**3s**	I	I	I
109	d1bf5a2.b.2.5.5_on_d1bg1a2.b.2.5.5.ali	21s	**22s**	D	M	M
110	d1bg1a2.b.2.5.5_on_d1bf5a2.b.2.5.5.ali	22s	**22s**	D	N	I
111	d1bg3b3.c.55.1.3_on_d1czan3.c.55.1.3.ali	46s	**44s**	I	I	I

	Alinhamento	Nest	Nest2	Est	RMSD	Msub
112	d1bgp__.a.93.1.1_on_d1qgja_.a.93.1.1.ali	55s	**46s**	D	N	M
113	d1bif_1.c.37.1.7_on_d1k6ma1.c.37.1.7.ali	16s	**15s**	I	I	I
114	d1bif_2.c.60.1.4_on_d1fbta_.c.60.1.4.ali	6s	**5s**	I	I	I
115	d1bif_2.c.60.1.4_on_d1k6ma2.c.60.1.4.ali	9s	**9s**	D	I	I
116	d1bio__.b.47.1.2_on_d1klt__.b.47.1.2.ali	33s	**43s**	D	M	N
117	d1bjya2.a.121.1.1_on_d2tct_2.a.121.1.1.ali	5s	**5s**	I	I	I
118	d1bkb_1.b.34.5.2_on_d2eifa1.b.34.5.2.ali	4s	**4s**	I	I	I
119	d1blu__.d.58.1.1_on_d1dura_.d.58.1.1.ali	15s	**10s**	D	M	N
120	d1bnla_.d.169.1.5_on_d1koe__.d.169.1.5.ali	3s	**2s**	D	I	I
121	d1bpb_1.a.60.12.1_on_d1bpya3.a.60.12.1.ali	5s	**4s**	D	N	M
122	d1bpya1.a.60.6.1_on_d1huoa1.a.60.6.1.ali	7s	**8s**	I	I	I
123	d1bpya3.a.60.12.1_on_d1bpb_1.a.60.12.1.ali	2s	**2s**	D	M	N
124	d1bpya3.a.60.12.1_on_d1rpl_1.a.60.12.1.ali	3s	**4s**	D	M	I
125	d1bpya4.d.218.1.2_on_d1rpl_2.d.218.1.2.ali	3s	**3s**	I	I	I
126	d1bqba_.d.92.1.2_on_d1keia_.d.92.1.2.ali	29s	**27s**	D	M	N
127	d1bqg_1.c.1.11.2_on_d1ec7a1.c.1.11.2.ali	13s	**12s**	I	I	I
128	d1brt__.c.69.1.12_on_d1a8q__.c.69.1.12.ali	22s	**24s**	D	M	M
129	d1brwa2.c.27.1.1_on_d2tpt_2.c.27.1.1.ali	24s	**23s**	D	I	I
130	d1bt0a_.d.15.1.1_on_d1ubi__.d.15.1.1.ali	3s	**3s**	I	I	I
131	d1bu8a2.c.69.1.19_on_d1etha2.c.69.1.19.ali	18s	**17s**	D	I	I
132	d1bu8a2.c.69.1.19_on_d1lpbb2.c.69.1.19.ali	17s	**17s**	D	I	I
133	d1bu8a2.c.69.1.19_on_d1rp1_2.c.69.1.19.ali	26m31s	**23m54s**	D	N	M
134	d1buca1.a.29.3.1_on_d1jqia1.a.29.3.1.ali	9s	**8s**	I	I	I
135	d1buhb_.d.97.1.1_on_d1cksa_.d.97.1.1.ali	2s	**1s**	I	I	I
136	d1buhb_.d.97.1.1_on_d1cksb_.d.97.1.1.ali	2s	**2s**	I	I	I
137	d1bv1__.d.129.3.1_on_d1fm4a_.d.129.3.1.ali	2s	**1s**	I	I	I
138	d1bvua2.c.58.1.1_on_d1gtma2.c.58.1.1.ali	9s	**9s**	I	I	I
139	d1bwoa_.a.52.1.1_on_d1fk5a_.a.52.1.1.ali	4s	**4s**	I	I	I
140	d1bwoa_.a.52.1.1_on_d1rzl__.a.52.1.1.ali	4s	**4s**	I	I	I
141	d1bwva1.c.1.14.1_on_d1geha1.c.1.14.1.ali	2m8s	**1m48s**	D	M	M
142	d1bwva2.d.58.9.1_on_d1rbla2.d.58.9.1.ali	15s	**18s**	D	N	I
143	d1bxba_.c.1.15.3_on_d1muwa_.c.1.15.3.ali	22s	**21s**	D	N	I
144	d1bxca_.c.1.15.3_on_d1muwa_.c.1.15.3.ali	23s	**24s**	D	I	I
145	d1byla_.d.32.1.2_on_d1qtoa_.d.32.1.2.ali	4s	**4s**	D	I	I
146	d1c3ab_.d.169.1.1_on_d1qdda_.d.169.1.1.ali	10s	**13s**	D	N	M
147	d1c3d__.a.102.4.4_on_d1qqfa_.a.102.4.4.ali	7s	**6s**	I	I	I
148	d1c3ma_.b.77.3.1_on_d1j4ta_.b.77.3.1.ali	27s	**33s**	D	M	M

	Alinhamento	Nest	Nest2	Est	RMSD	Msub
149	d1c44a_.d.106.1.1_on_d1ikta_.d.106.1.1.ali	8s	**7s**	D	M	I
150	d1c6ya_.b.50.1.1_on_d1idaa_.b.50.1.1.ali	4s	**4s**	I	I	I
151	d1c8ca_.d.9.2.1_on_d1azpa_.d.9.2.1.ali	2s	**2s**	I	I	I
152	d1c9ba2.a.74.1.2_on_d1aisb2.a.74.1.2.ali	19s	**20s**	D	N	M
153	d1c9ha_.d.26.1.1_on_d1fd9a_.d.26.1.1.ali	10s	**9s**	D	M	N
154	d1ca1_1.a.124.1.1_on_d1khoa1.a.124.1.1.ali	8s	**8s**	D	M	N
155	d1ca1_2.b.12.1.3_on_d1khoa2.b.12.1.3.ali	5s	**5s**	D	I	I
156	d1cb6a2.c.94.1.2_on_d1a8e__.c.94.1.2.ali	2m5s	**2m14s**	D	N	N
157	d1cb6a2.c.94.1.2_on_d1dtza2.c.94.1.2.ali	35s	**33s**	D	I	I
158	d1cb6a2.c.94.1.2_on_d1iq7a_.c.94.1.2.ali	39s	**34s**	D	M	I
159	d1cbia_.b.60.1.2_on_d1ggla_.b.60.1.2.ali	16s	**15s**	D	N	N
160	d1cbja_.b.1.8.1_on_d1xsoa_.b.1.8.1.ali	4s	**4s**	I	I	I
161	d1cbka_.d.58.30.1_on_d1eqma_.d.58.30.1.ali	7s	**7s**	D	N	I
162	d1ccd__.a.101.1.1_on_d1utg__.a.101.1.1.ali	5s	**4s**	D	N	N
163	d1cdoa2.c.2.1.1_on_d1e3ia2.c.2.1.1.ali	9s	**9s**	I	I	I
164	d1cdoa2.c.2.1.1_on_d1ht0a2.c.2.1.1.ali	9s	**8s**	I	I	I
165	d1cdwa1.d.129.1.1_on_d1aisa1.d.129.1.1.ali	5s	**5s**	I	I	I
166	d1cdwa2.d.129.1.1_on_d1ytba2.d.129.1.1.ali	3s	**2s**	I	I	I
167	d1ce2a2.c.94.1.2_on_d1ce2a1.c.94.1.2.ali	4m21s	**4m37s**	D	I	I
168	d1cero2.d.81.1.1_on_d1gd1o2.d.81.1.1.ali	7s	**6s**	I	I	I
169	d1cero2.d.81.1.1_on_d1i32a2.d.81.1.1.ali	12s	**12s**	D	N	I
170	d1cewi_.d.17.1.2_on_d1g96a_.d.17.1.2.ali	4s	**4s**	I	I	I
171	d1cf1a1.b.1.18.11_on_d1g4ma1.b.1.18.11.ali	19s	**17s**	D	M	I
172	d1cfr__.c.52.1.7_on_d1knva_.c.52.1.7.ali	27s	**26s**	D	M	M
173	d1chma2.d.127.1.1_on_d1kp0a2.d.127.1.1.ali	34s	**33s**	D	N	I
174	d1ci3m1.b.2.6.1_on_d1e2wa1.b.2.6.1.ali	6s	**6s**	I	I	I
175	d1ci3m1.b.2.6.1_on_d1hcz_1.b.2.6.1.ali	6s	**6s**	D	N	I
176	d1cipa1.a.66.1.1_on_d1tada1.a.66.1.1.ali	7s	**6s**	I	I	I
177	d1ciu_2.b.3.1.1_on_d1cgt_2.b.3.1.1.ali	25s	**24s**	I	I	I
178	d1cjca2.c.4.1.1_on_d1lqta2.c.4.1.1.ali	37s	**30s**	D	M	N
179	d1ck1a1.b.40.2.2_on_d1klud1.b.40.2.2.ali	9s	**8s**	I	I	I
180	d1ck4a_.c.62.1.1_on_d1aoxa_.c.62.1.1.ali	10s	**9s**	I	I	I
181	d1ck4b_.c.62.1.1_on_d1qc5a_.c.62.1.1.ali	4s	**4s**	I	I	I
182	d1cksa_.d.97.1.1_on_d1buhb_.d.97.1.1.ali	2s	**2s**	I	I	I
183	d1ckta_.a.21.1.1_on_d1gt0d_.a.21.1.1.ali	6s	**5s**	D	I	I
184	d1cm7a_.c.77.1.1_on_d1cnza_.c.77.1.1.ali	6s	**5s**	I	I	I
185	d1coja1.a.2.11.1_on_d1sssa1.a.2.11.1.ali	30s	**23s**	D	M	N

	Alinhamento	Nest	Nest2	Est	RMSD	Msub
186	d1coja2.d.44.1.1_on_d3sdpa2.d.44.1.1.ali	30s	**24s**	D	N	M
187	d1cpm__.b.29.1.2_on_d1gbg__.b.29.1.2.ali	3s	**3s**	I	I	I
188	d1cqwa_.c.69.1.8_on_d1iz7a_.c.69.1.8.ali	20s	**20s**	D	M	N
189	d1crb__.b.60.1.2_on_d1ifc__.b.60.1.2.ali	25s	**30s**	D	N	M
190	d1crka1.a.83.1.1_on_d1g0wa1.a.83.1.1.ali	9s	**8s**	I	I	I
191	d1ctf__.d.45.1.1_on_d1dd3a2.d.45.1.1.ali	4s	**3s**	I	I	I
192	d1cvl__.c.69.1.18_on_d4lipd_.c.69.1.18.ali	8s	**11s**	D	N	N
193	d1cvl__.c.69.1.18_on_d4lipe_.c.69.1.18.ali	11s	**10s**	D	M	N
194	d1cvua1.a.93.1.2_on_d1eqga1.a.93.1.2.ali	32s	**29s**	D	I	I
195	d1cwva2.b.1.14.1_on_d1f00i1.b.1.14.1.ali	13s	**12s**	D	M	N
196	d1cx7a_.d.2.1.3_on_d1l65__.d.2.1.3.ali	2s	**3s**	I	I	I
197	d1cxva_.d.92.1.11_on_d1jk3a_.d.92.1.11.ali	14s	**14s**	D	I	I
198	d1cyg_2.b.3.1.1_on_d1ciu_2.b.3.1.1.ali	22s	**21s**	I	I	I
199	d1cyna_.b.62.1.1_on_d1lopa_.b.62.1.1.ali	34s	**33s**	D	N	M
200	d1czna_.c.23.5.1_on_d1rcf__.c.23.5.1.ali	5s	**5s**	D	M	I
201	d1czta_.b.18.1.2_on_d1d7pm_.b.18.1.2.ali	8s	**8s**	D	N	M
202	d1czya1.b.8.1.1_on_d1flka1.b.8.1.1.ali	7s	**6s**	D	M	N
203	d1d1ja_.d.110.1.1_on_d1fil__.d.110.1.1.ali	4s	**4s**	I	I	I
204	d1d1qa_.c.44.1.1_on_d5pnt__.c.44.1.1.ali	16s	**16s**	D	M	M
205	d1d2ea1.b.43.3.1_on_d1efca1.b.43.3.1.ali	8s	**8s**	D	N	M
206	d1d2ea2.b.44.1.1_on_d1efca2.b.44.1.1.ali	27s	**24s**	D	M	N
207	d1d2ea2.b.44.1.1_on_d1exma2.b.44.1.1.ali	8s	**7s**	I	I	I
208	d1d2oa1.b.3.5.1_on_d1d2oa2.b.3.5.1.ali	9s	**8s**	I	I	I
209	d1d2oa2.b.3.5.1_on_d1d2oa1.b.3.5.1.ali	11s	**12s**	D	M	N
210	d1d3ca2.b.3.1.1_on_d1cyg_2.b.3.1.1.ali	20s	**20s**	I	I	I
211	d1d3ca2.b.3.1.1_on_d1qhoa2.b.3.1.1.ali	2m1s	**1m45s**	D	N	M
212	d1d4aa_.c.23.5.3_on_d1dxqa_.c.23.5.3.ali	6s	**5s**	I	I	I
213	d1d4aa_.c.23.5.3_on_d1qrda_.c.23.5.3.ali	5s	**5s**	I	I	I
214	d1d7ka2.c.1.6.1_on_d1f3tc2.c.1.6.1.ali	17s	**17s**	D	I	I
215	d1d7ka2.c.1.6.1_on_d2toda2.c.1.6.1.ali	20s	**20s**	D	N	M
216	d1d7pm_.b.18.1.2_on_d1czta_.b.18.1.2.ali	8s	**8s**	D	M	I
217	d1d9ca_.a.26.1.3_on_d1fyha1.a.26.1.3.ali	8s	**8s**	I	I	I
218	d1dar_4.d.58.11.1_on_d1n0ua5.d.58.11.1.ali	43s	**40s**	D	I	I
219	d1dbbh1.b.1.1.1_on_d1b2wl1.b.1.1.1.ali	37s	**38s**	D	M	M
220	d1dd3a2.d.45.1.1_on_d1ctf__.d.45.1.1.ali	5s	**5s**	D	I	I
221	d1dd5a_.d.67.3.1_on_d1eh1a_.d.67.3.1.ali	9s	**9s**	I	I	I
222	d1dd5a_.d.67.3.1_on_d1ek8a_.d.67.3.1.ali	12s	**11s**	I	I	I

	Alinhamento	Nest	Nest2	Est	RMSD	Msub
223	d1ddga1.b.43.4.1_on_d1f20a1.b.43.4.1.ali	1m5s	**1m5s**	D	N	N
224	d1ddga2.c.25.1.4_on_d1f20a2.c.25.1.4.ali	16s	**15s**	D	I	I
225	d1ddga2.c.25.1.4_on_d1ja1a3.c.25.1.4.ali	25s	**24s**	D	N	M
226	d1ddwa_.b.55.1.4_on_d1i7aa_.b.55.1.4.ali	6s	**6s**	I	I	I
227	d1ddza1.c.53.2.1_on_d1ddza2.c.53.2.1.ali	51s	**54s**	I	I	I
228	d1ddza1.c.53.2.1_on_d1i6pa_.c.53.2.1.ali	11s	**10s**	D	N	I
229	d1df0a1.a.39.1.8_on_d1alva_.a.39.1.8.ali	22s	**22s**	D	N	I
230	d1df0a1.a.39.1.8_on_d1kful1.a.39.1.8.ali	28s	**26s**	I	I	I
231	d1dg9a_.c.44.1.1_on_d5pnt__.c.44.1.1.ali	4s	**3s**	I	I	I
232	d1dgja1.a.56.1.1_on_d1jroa1.a.56.1.1.ali	42s	**36s**	D	N	I
233	d1dgja3.d.41.1.1_on_d1hlra3.d.41.1.1.ali	1m9s	**1m4s**	D	I	I
234	d1dgja4.d.133.1.1_on_d1hlra4.d.133.1.1.ali	1m24s	**1m20s**	D	N	I
235	d1dgsa3.d.142.2.2_on_d1b04a_.d.142.2.2.ali	33s	**31s**	D	N	M
236	d1dgwa_.b.82.1.2_on_d2phla1.b.82.1.2.ali	18s	**20s**	D	N	I
237	d1dkgd2.c.55.1.1_on_d1hjoa2.c.55.1.1.ali	21s	**19s**	D	I	I
238	d1dlc_1.b.18.1.3_on_d1ciy_1.b.18.1.3.ali	21s	**20s**	D	I	I
239	d1dlwa_.a.1.1.1_on_d1idra_.a.1.1.1.ali	4s	**5s**	D	M	N
240	d1dnpa1.a.99.1.1_on_d1iqra1.a.99.1.1.ali	1m55s	**1m55s**	D	M	N
241	d1dpja_.b.50.1.2_on_d1lf2a_.b.50.1.2.ali	51s	**51s**	D	N	N
242	d1dqza_.c.69.1.3_on_d1f0na_.c.69.1.3.ali	8s	**9s**	I	I	I
243	d1ds6b_.b.1.18.8_on_d1kmta_.b.1.18.8.ali	3s	**3s**	I	I	I
244	d1dssg2.d.81.1.1_on_d1gado2.d.81.1.1.ali	5s	**4s**	I	I	I
245	d1dsxa_.d.42.1.2_on_d1t1da_.d.42.1.2.ali	2s	**2s**	I	I	I
246	d1dt4a_.d.51.1.1_on_d1dtja_.d.51.1.1.ali	1s	**2s**	I	I	I
247	d1dtja_.d.51.1.1_on_d1dt4a_.d.51.1.1.ali	2s	**1s**	I	I	I
248	d1dtwb1.c.36.1.3_on_d1qs0b1.c.36.1.3.ali	18s	**18s**	D	N	I
249	d1du5a_.b.25.1.1_on_d1aun__.b.25.1.1.ali	13s	**12s**	D	I	I
250	d1du5a_.b.25.1.1_on_d1kwna_.b.25.1.1.ali	14s	**13s**	D	N	M
251	d1dura_.d.58.1.1_on_d1fca__.d.58.1.1.ali	2s	**3s**	I	I	I
252	d1duxc_.a.4.5.21_on_d1md0a_.a.4.5.21.ali	5s	**3s**	D	M	M
253	d1dvpa1.a.118.9.2_on_d1elka_.a.118.9.2.ali	11s	**11s**	D	I	I
254	d1dxqa_.c.23.5.3_on_d1qrda_.c.23.5.3.ali	5s	**4s**	I	I	I
255	d1dxrh1.b.41.1.1_on_d1eysh1.b.41.1.1.ali	1m26s	**1m21s**	D	I	I
256	d1dxxa1.a.40.1.1_on_d1qaga1.a.40.1.1.ali	5s	**5s**	I	I	I
257	d1dy5a_.d.5.1.1_on_d1gv7a_.d.5.1.1.ali	8s	**7s**	D	N	N
258	d1dysa_.c.6.1.1_on_d1qjwa_.c.6.1.1.ali	40s	**43s**	D	N	N
259	d1dzaa_.d.5.1.1_on_d1dyta_.d.5.1.1.ali	9s	**8s**	D	I	I

	Alinhamento	Nest	Nest2	Est	RMSD	Msub
260	d1e2wa1.b.2.6.1_on_d1ci3m1.b.2.6.1.ali	8s	**7s**	D	I	I
261	d1e2wa1.b.2.6.1_on_d1hcz_1.b.2.6.1.ali	5s	**5s**	D	I	I
262	d1e2wa2.b.84.2.2_on_d1ci3m2.b.84.2.2.ali	5s	**4s**	D	M	N
263	d1e3oc2.a.35.1.1_on_d1au7a2.a.35.1.1.ali	3s	**3s**	I	I	I
264	d1e4ia_.c.1.8.4_on_d1cbg__.c.1.8.4.ali	2m15s	**3m11s**	D	N	M
265	d1e51a_.c.1.10.3_on_d1h7na_.c.1.10.3.ali	19s	**17s**	D	N	N
266	d1e5ma2.c.95.1.1_on_d1kas_2.c.95.1.1.ali	10s	**10s**	I	I	I
267	d1e6va2.d.58.31.2_on_d1e6ya2.d.58.31.2.ali	33s	**32s**	D	N	I
268	d1e6va2.d.58.31.2_on_d1hbna2.d.58.31.2.ali	27s	**23s**	D	M	I
269	d1e6vb1.a.89.1.1_on_d1e6yb1.a.89.1.1.ali	13s	**12s**	D	I	I
270	d1e6vb1.a.89.1.1_on_d1hbnb1.a.89.1.1.ali	11s	**10s**	D	I	I
271	d1e6vb2.d.58.31.2_on_d1hbnb2.d.58.31.2.ali	15s	**16s**	D	I	I
272	d1e6ya2.d.58.31.2_on_d1hbna2.d.58.31.2.ali	2m45s	**2m39s**	D	N	M
273	d1e6yc_.d.58.31.1_on_d1e6vc_.d.58.31.1.ali	15s	**14s**	D	I	I
274	d1e6yc_.d.58.31.1_on_d1hbnc_.d.58.31.1.ali	18s	**15s**	D	M	N
275	d1e85a_.a.24.3.2_on_d1jafa_.a.24.3.2.ali	7s	**6s**	D	I	I
276	d1e89a_.c.69.1.20_on_d1qj4a_.c.69.1.20.ali	11s	**11s**	D	M	I
277	d1e9ia1.c.1.11.1_on_d1onea1.c.1.11.1.ali	21s	**19s**	I	I	I
278	d1e9ia1.c.1.11.1_on_d1pdz_1.c.1.11.1.ali	25s	**27s**	D	M	I
279	d1e9ya1.b.85.3.1_on_d1ejrb_.b.85.3.1.ali	1m29s	**1m24s**	I	I	I
280	d1e9ya1.b.85.3.1_on_d4ubpb_.b.85.3.1.ali	26s	**24s**	D	M	N
281	d1e9ya2.d.8.1.1_on_d1ejra_.d.8.1.1.ali	7s	**6s**	I	I	I
282	d1e9ya2.d.8.1.1_on_d4ubpa_.d.8.1.1.ali	7s	**7s**	D	I	I
283	d1e9yb1.b.92.1.1_on_d1fwfc1.b.92.1.1.ali	49s	**47s**	I	I	I
284	d1e9yb1.b.92.1.1_on_d4ubpc1.b.92.1.1.ali	47s	**47s**	D	M	N
285	d1e9yb2.c.1.9.2_on_d4ubpc2.c.1.9.2.ali	33s	**1m6s**	D	M	I
286	d1ea5a_.c.69.1.1_on_d1b41a_.c.69.1.1.ali	1m11s	**55s**	D	N	M