HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERARBEIT

Diplom-Informatiker(FH)
Knut Altroggen

BioJava für den Einsatz in Forschung und Lehre

2011

Fakultät **Mathematik Naturwissenschaften Informatik**

MASTERARBEIT

BioJava für den Einsatz in Forschung und Lehre

Autor:
Knut Altroggen

Studiengang:
Master Informatik

Seminargruppe:
IF09w1-M

Erstprüfer:
Professor Dr. rer. nat. Dirk Labudde

Zweitprüfer:
Professor Dr.-Ing. Mario Geissler

Mittweida, 2011

Bibliografische Angaben

Altroggen, Knut: BioJava für den Einsatz in Forschung und Lehre, 103 Seiten, 18 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik Naturwissenschaften Informatik

Masterarbeit, 2011

Referat

Ziel der Masterarbeit ist es einen Überblick über die freie Bibliothek „BioJava“ zu erstellen und diese dann exemplarisch in einen Tool für die Forschung und Lehre darzustellen. Die Reduzierung des Arbeitsaufwandes bei der Verarbeitung biologischer Daten soll durch das entstehende Tool minimiert werden. Durch den Einsatz in der Lehre sollen die Studentinnen und Studenten eine Grundlage erhalten für den Umgang mit wissenschaftlichen Daten.
Der Einsatz moderner Technolgien soll zudem gewährleisten, dass das Tool viele Jahre eingesetzt werden kann.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

CDS	coding sequences
CE	Combinatorial Extension Algorithm
FATCAT	Flexible structure alignment by chaining aligned fragment pairs allowing twists
GFF	General Feature Format
GTF	Gene Transfer Format
LGPL 2.1.	GNU Lesser General Public License, version 2.1
MSA	Multiple Sequence Alignment
NCBI	National Center for Biotechnology Information
SCOP	Sturctural Classification of Proteins
UPGMA	Unweighted Pair Group Method with Arithmetic mean
WPGMA	weighted pair group method with averaging
XML	Extensible Markup Language

V. Vorwort

Diese wissenschaftliche Arbeit hat das Ziel, die freie Bibliothek „BioJava“ in der Version 3.0.1 für die Anwendung in Forschung und Lehre aufzubereiten und dies über ein entsprechendes Tool, im Status eines Prototypen, zu realisieren.
Mit der Entwicklung des Tools soll zudem eine Grundlage für die Ausbildung der Studentinnen und Studenten der Studiengänge Biotechnolgie/Bioinformatik und Molekularbiologie/Bioinformatik schaffen. Ein weiterer Punkt den das Tool abdecken soll, ist die Kombination zahlreicher Einzeltools in einem, was zu einer deutlichen Vereinfachung der Arbeit führen soll.
Führende Wissenschaftler benutzen dieses Framework bereits für ihre Arbeiten.[1] Durch diese Arbeit soll dies nun auch an der Hochschule Mittweida Einzug halten.
Mein persönlicher Dank gilt Herrn Prof. Dirk Labudde für die Idee für diese Masterarbeit. Herrn Prof. Mario Geissler, Frau B.Sc. Bianca Liebscher, Frau B.Sc. Marleen Kreuzer, Frau Stefanie Geske, Frau Sandy Weissflog, Frau Kristin Uth und Frau Christin Lembke danke ich für die fachliche Unterstützung während der gesamten Zeit und dies unter Aufopferung ihrer kostenbaren freien Zeit.

[1] [1] Vgl. Abstract S. 2096

1 Grundlagen

1.1 Biologie

1.1.1 Genetik

Das Wort Genetik stammt aus den griechischen und bedeutet soviel wie Abstammung oder Ursprung. Als Teilgebiet der Biologie befasst sich die Genetik mit Grundlagen der Erzeugung und Vererbung der Erbinformationen. [2]

1.1.2 NCBI

Das National Center for Biotechnology Information (NCBI) stellt eine sehr große Auswahl von Datenbanken zur Verfügung. Diese beinhalteten unter anderen die Genomedatenbank, für die Suche nach Genomen unterschiedlichster Spezies, oder die PubMed in welcher wissenschaftliche Artikel veröffentlicht werden. Die wissenschaftlichen Artikel sind auf der NCBI meistens nur mit einen Abstract vorhanden und verweisen auf andere Seiten, wo das Dokument als pdf verfügbar ist.

[2] [3] Vgl. Seite 2

1.1.3 Codon

Jeweils drei Basenpaare der Nukleinsäure legen eine Aminosäure fest. Diese drei Basenpaare werden als Codon bezeichnet. Die Codonen überlappen sich dabei nicht, sondern folgen in einer für jedes Lebewesen spezifischen Reihenfolge. Es gilt dabei die Regel: Jedes Codon kodiert genau eine Aminosäure, aber eine Aminosäuren kann durch mehrere Codonen dargestellt werden. Die Codonen werden meistens in einer Codonhäufigkeitstabelle(siehe Tabelle 1.1)[3] angeordnet. Es gibt insgesamt 64 Codonen, wobei es für Methionin in der Regel nur ein Codon gibt. Ausnahme bildet unteranderem Escherichia Coli, hier können auch noch andere Codonen Methionin kodieren.[4]

	T		C		A		G		
T	TTT	2.08	TCT	0.89	TAT	1.53	TGT	0.49	T
	TTC	1.78	TCC	0.90	TAC	1.30	TGC	0.65	C
	TTA	1.22	TCA	0.64	TAA	0.19	TGA	0.09	A
	TTG	1.28	TCG	0.86	TAG	0.02	TGG	1.48	G
C	CTT	1.00	CCT	0.65	CAT	1.23	CGT	2.29	T
	CTC	1.06	CCC	0.47	CAC	1.04	CGC	2.30	C
	CTA	0.35	CCA	0.81	CAA	1.43	CGA	0.32	A
	CTG	5.56	CCG	2.47	CAG	2.93	CGG	0.49	G
A	ATT	2.91	ACT	0.91	AAT	1.58	AGT	0.76	T
	ATC	2.64	ACC	2.42	AAC	2.28	AGC	1.59	C
	ATA	0.36	ACA	0.59	AAA	3.47	AGA	0.16	A
	ATG	2.80	ACG	1.37	AAG	1.07	AGG	0.11	G
G	GTT	1.88	GCT	1.57	GAT	3.18	GGT	2.60	T
	GTC	1.49	GCC	2.51	GAC	2.05	GGC	3.07	C
	GTA	1.11	GCA	1.98	GAA	4.12	GGA	0.67	A
	GTG	2.66	GCG	3.49	GAG	1.80	GGG	1.02	G

Tabelle 1.1: Gemittelte Codonhäufigkeitstabelle Escherichia Coli K-12

1.1.4 DNA

DNA steht für „desoxyribonucleic acid“ und besteht aus den vier Basen Adenin, Thymin, Guanin und Cytosin, welche ein Polymer bilden. Die DNA ist dabei der Baustein des Lebens. In der deutschen Sprache ist es nicht DNA sondern DNS, wobei das „A“ für acid dem Wort Säure weichen musste.

Gen

Das Gen ist die Grundeinheit der Vererbung, durch welche die Eigenschaften vom Elternteil zum Kind weitergegeben werden. Das Gen ist ein bestimmter Abschnitt auf der

[3] [5] entnommen S.8
[4] [3] Vgl. Seite 56

DNA, welche eine bestimmte Beeinflussung auf den Organismus hat. Jede lebende Zelle trägt dabei eine vollständige Ergänzung der Gene in linearer Reihenfolge auf den Chromosomen, angepasst auf die jeweilige Spezies, mit sich. [5]

Chromosomen

Chromosomen enthalten die Gene und somit auch die Erbinformationen. Unter einem Lichtmikroskop werden bei der Mitose und Meiose die Chromosomen als kleine stäbchenförmige Objekte(siehe Abb.1.1) sichtbar. [6]

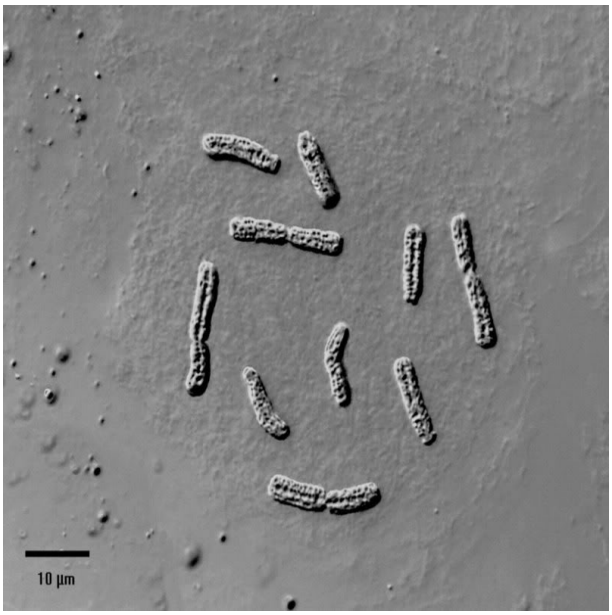

Abbildung 1.1: Reifeteilung der Spermatocyten beim Strudelwurm Mesostoma von F. Göltenboth

Intron

Ein Intron ist eine nicht kodierte Nukleotidsequenz. Eine grosse Anzahl von Genen hat eine stattliche Zahl von Intronsequenzen, welche fast das gesamte Gen umfassen. [7]

Exon

Das Exon ist, im Gegensatz zum Intron, der kodierte Teil der DNA-Sequenz. Die Exons werden im Gen durch jeweils ein Intron getrennt.[8]

[5] [4] Vgl. S. 253
[6] [4] Vgl. S. 119
[7] [4] Vgl. S. 333
[8] [4] Vgl. S. 220

1.1.5 RNA

RNA steht für „ribonucleic acid“ und besteht aus den vier Basen Adenin, Uracil, Guanin und Cytosin, welche ein Polymer bilden. Der Unterschied zwischen DNA und RNA besteht darin, das in der RNA das Thymin durch Uracil ausgetauscht wird. Die RNA wird auch noch in andere Bereiche, wie zum Beispiel tRNA, rRNA oder mRNA unterteilt.

1.1.6 Protein

Proteine sind der wichtigste Bestandteil von lebender Materie. In lebendigen Organismen treten sie in Enzymen, in Baugruppen von Zellen oder Geweben und bei der Kontrolle der Genexpression auf.
Die Genexpression ist die Umwandlung vom Erbbild/Muster zum Erscheinungsbild des Organismus. Sie wird in die Transkription und die Translation unterteilt. Die Transkription ist das umschreiben der DNA in die mRNA. Die Translation übersetzt die RNA in ein Protein.

1.1.7 gff Format

Im GFF (General Feature Format) müssen neun Felder vorhanden sein, welche durch einen Tabulator getrennt sind.

Inhalt der neun Felder [9] :

1. seqname - Name der Sequenz, muss eine Chromosom oder Gerüst dieser.
2. source - Das Programm welches es erstellt hat.
3. feature - Name des Merkmales, wie „CDS“, „start_codon“, „stop_codon“ oder „exon“.
4. start - Startposition in der Sequenz. Beginn bei 1.
5. end - Endposition(inklusive) in der Sequenz.
6. score - Ein Wert zwischen 0 und 1000 oder „.“, falls kein Wert vorhanden.
7. strand - ’+’, ’-’, oder ’.’ für nicht bekannt / ist egal.
8. frame - Ist das Merkmal ein „exon“, sollte es eine Nummer zwischen 0 und 2 bekommen oder „.“ wenn es kein „exon“ ist
9. group - Alle Linien einer Gruppe zusammengefasst.

Die GFF2 bzw. GFF3 Formate unterscheiden sich nur minimal von dem GFF und beinhalten noch mehr Informationen, welche in der „group“-Eigenschaft festgehalten werden.

1.1.8 gtf Format

Das GTF (Gene Transfer Format) ist die Verfeinerung des GFF, indem die Spezifikationen[10] gestrafft wurden. Das Feld „group“ wurde durch eine Liste von Eigenschaften, in der Form Typ/Wert, erweitert.

[9] Entnommen [2]
[10] Vgl. [2]

1.1.9 Sequenzalignment

Das Alignment erstellt einen Vergleich zwischen zwei oder mehreren Sequenzen und richtet diese dann passend aus. Bei den Vergleich von zwei Sequenzen kann man entweder das lokale oder globale Alignment anwenden. Diese beiden Verfahren unterscheiden sich darin, dass das lokal Alignment eine möglichst hohe Übereinstimmung in einen Teil der Sequenz erzielen will, wohin gegen das globale Alignment sich auf die gesamte Sequenz ausrichtet.

Isofunktionelle Gruppen

Aminosäuren werden in Gruppen zusammengefasst, wenn sie die gleichen physikalischen und/oder chemischen Eigenschaften besitzen. Diese Gruppierung wird meistens in einen Venndiagramm(siehe Abb. 1.2)[11] dargestellt. Eine alternative Darstellungsmethode ist die Eigenschaften in Tabellenform anzuordnen.

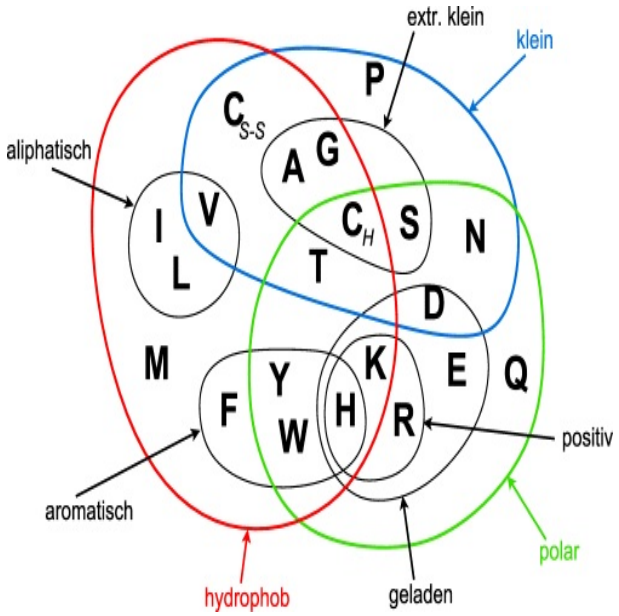

Abbildung 1.2: Venndiagramm isofunktionelle Gruppen

[11] [5] entnommen S.6

Darstellungsform

Das Ergebnis(siehe Abb. 1.3) wird dargestellt, in dem alle Sequenzen unter einander geschrieben und dann mittels „-“, für eine Lücke, angeordnet werden. Es kann auch vorkommen, dass zwei Buchstaben untereinander stehen und dabei jedoch nicht gleich sind. Dies kommt durch die isofunktionellen Gruppen zustande.

```
1PQS_A|PDBID|CHAIN|SEQUENCE      ---------------------------SEIFTLLVEKVWNFDDLIMAINS 23
1M7K_A|PDBID|CHAIN|SEQUENCE      -------------------------------NQDQSSSLPEECVPSDE 17
1R4T_A|PDBID|CHAIN|SEQUENCE      GSASSAVVFKQMVLQQALPMTLKGLDKASELATLTPEGLAREHSRLASGD 50
1SS6_A|PDBID|CHAIN|SEQUENCE      ---------------GSEKRQHSSQDVHVVLKLWKSGFSLDNGELRSYQ 34
                                                          .     .  :   .

1PQS_A|PDBID|CHAIN|SEQUENCE      KISNTHNNNISPI----TKIKYQDED-GDFVVL---GSDED-WN----VA 60
1M7K_A|PDBID|CHAIN|SEQUENCE      STPPSIKKIIHVL----EKVQYLEQEVEEFVGK---KTDKAYWLLEEMLT 60
1R4T_A|PDBID|CHAIN|SEQUENCE      GALRSLSTALAGIRAG-SQVEESRIQAGRLLERSIGGIALQQWGTTGGAA 99
1SS6_A|PDBID|CHAIN|SEQUENCE      DPSN--AQFLESIRRGEVPAELRRLAHGGQVNLDMEDHRDEDFVKPKGAF 82
                                       :  :       :         :          :

1PQS_A|PDBID|CHAIN|SEQUENCE      KEMLAEN--------------NEKFLNIRLY------------------- 77
1M7K_A|PDBID|CHAIN|SEQUENCE      KELLELDSVEIGGQDSVRQARKEAVCKIQAILEKLEKKG----------- 99
1R4T_A|PDBID|CHAIN|SEQUENCE      SQLVLDASPELR--REITDQLHQVMSEVALLRQAVESEVSRVSADYPYDV 147
1SS6_A|PDBID|CHAIN|SEQUENCE      KAFTGEG---------------QKLGSTAPQVLST-------------- 102
                                 . :                   : . .

1PQS_A|PDBID|CHAIN|SEQUENCE      ------
1M7K_A|PDBID|CHAIN|SEQUENCE      ------
1R4T_A|PDBID|CHAIN|SEQUENCE      PDYASL 153
1SS6_A|PDBID|CHAIN|SEQUENCE      ------
```

Abbildung 1.3: MSA - ClustalW über 1pqs, 1m7k,1r4t und 1ss6

Smith Waterman

Der Smith-Waterman-Algorithmus sagt das optimale lokale Alignment für zwei gegebene Sequenzen voraus. Der Algorithmus besitzt dabei die drei möglichen Werte: Match bzw. Missmatch, Deletion und Insertion. Aus den ermittelten Werten wird sich der größte Wert genommen und dieser diagonal zurück verfolgt, bis man auf eine Null oder den Matrixrand trifft. [12]

- a, b ... Zeichenketten über dem Alphabet der Proteine
- m ... Länge der Zeichenkette a
- n ... Länge der Zeichenkette b
- w(c,d) ... Alignment-Score, wobei $c,d \in$ Alphabet der Proteine $\cup$ -

$$A(i,0) = 0, 0 \leq i \leq m$$
$$A(0,j) = 0, 0 \leq j \leq n$$
$$A(i,j) = max \left\{ \begin{array}{cc} 0 & \text{leer} \\ A(i-1,j-1) + w(a_i, b_j) & \text{Match bzw. Missmatch} \\ A(i-1,j) + w(a_i, -) & \text{Deletion} \\ A(i,j-1) + w(-, b_j) & \text{Insertion} \end{array} \right\}, 1 \leq i \leq m, 1 \leq j \leq n$$

Needleman Wunsch

Der Needleman-Wunsch-Algorithmus sagt das optimale globale Alignment für zwei gegebene Sequenzen voraus. Auch dieser Algorithmus hat die drei möglichern Werte, nur das hier der günstigste Weg zwischen der linken oberen und rechten unteren Ecke gesucht wird. [13]

- a, b ... Zeichenketten über dem Alphabet der Proteine
- m ... Länge der Zeichenkette a
- n ... Länge der Zeichenkette b
- w(c,d) ... Alignment-Score, wobei $c,d \in$ Alphabet der Proteine
- f ... Lückenkostenfunktion

$$A(0,0) = 0$$
$$A(i,0) = f(i), 1 \leq i \leq m$$
$$A(0,j) = f(j), 2 \leq j \leq n$$
$$A(i,j) = max \left\{ \begin{array}{cc} A(i-1,j-1) + w(a_i, b_j) & \text{Match bzw. Missmatch} \\ max_{1 \leq k \leq i} A(i-k,j) + f(k) & \text{Deletion} \\ max_{1 \leq l \leq j} A(i,j-l) + f(l) & \text{Insertion} \end{array} \right\}, 1 \leq i \leq m, 1 \leq j \leq n$$

[12] [6] Vgl. S. 195ff.
[13] [7] Vgl. S. 443ff.

Multiple Sequence Alignment

Domainen oder Motife, welche eine Proteinfamilie charakterisieren, sind durch ein Multiple Sequence Alignment(MSA) definiert. Ein MSA besteht aus drei oder mehr Sequenzen, welche zu einander aligniert sind. Im Alignment stehen dabei die homologen Reste untereinander. Diese homologen Reste deuten auf einen gemeinsamen Vorfahren hin.[14]

1.1.10 PDB Datei

Die PDB Datei beinhaltet für eine Struktur die Daten für Atomkoordinaten, kristallographische Strukturfaktoren und experimentellen NMR Daten. Abgesehen von den Koordinaten enthält sie auch noch die Namen von Molekülen, primäre und sekundäre Struktur Informationen, Sequenz Referenzen, Liganden(Bindungen) und biologische Anordnungsinformationen. Details über die Datenerhebung und bibliographische Hinweise sind ebenfalls enthalten. Durch die PDB Datei lässt sich somit auch die 3D Struktur(siehe Abb. 1.4)[15] darstellen.

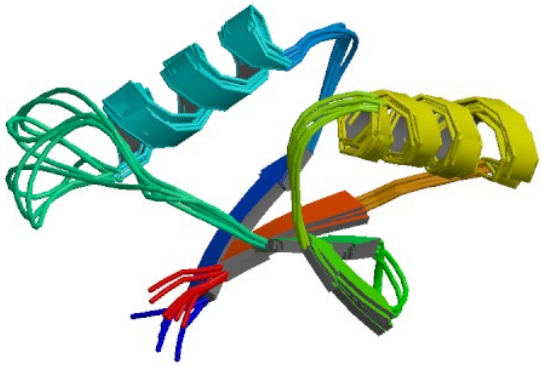

Abbildung 1.4: 3D Struktur des 1pqs

1.1.11 mmCIF Datei

Das macromolecular Crystallographic Information File (mmCIF) bietet eine Möglichkeit um im Detail die Eigenschaften zu einer Struktur und das Experiment, das zur Ableitung dieser Struktur führte, zu beschreiben. Die mmCIF kann für den Datenaustausch und Datenarchivierung verwendet werden.

[14] [8]Vgl. S. 180
[15] entnommen [30]

1.1.12 JMol

JMol ist ein Open Source 3D Betrachtungsprogramm für chemische Strukturen. Es beinhaltet auch Möglichkeiten für die Darstellung von Chemikalien, Kristallen, Materialien und Biomolekülen. Durch einige Funktionen wie Animationen, Vibrationen, Oberflächensimulation und Orbitaldarstellung wird es dem Anwender sehr erleichtert, sich die Struktur in 3D darzustellen. Mit der Unterstützung für Elementarzellen, Symmetrie-Operationen, schematische Formen für sekundäre Strukturen in Biomolekülen oder Messungen(Abstand, Winkel, Torsionswinkel, ...) sind auch wissenschaftliche Betrachtungen möglich. Der Umgang mit JMol erfordert für die Berechnungen etwas Erfahrung, für das reine Anzeigen und die Grundfunktionen braucht man keine Kenntnisse des Tools.

1.1.13 SCOP

Die Structural Classification of Proteins(SCOP) Datenbank beinhaltet eine detaillierte und umfassende Beschreibung der Struktur und evolutionären Beziehungen zwischen allen Proteinen, von denen die Struktur bekannt ist. Fast alle Proteine haben strukturelle Ähnlichkeiten mit anderen Proteinen und in einigen von diesen Fällen teilen sich einen gemeinsamen evolutionären Ursprung. Die Unterteilung wird unter anderen in Familie und Species gemacht. Diese Einteilungen sind der sogenannte taxonomische Rang. Dieser Rang stammt ursprünglich es der Zoology und wurde auf die Proteine angepasst. Durch die Einteilung in die Ränge ist die Darstellung des Inhaltes am besten über einen Baum zu realisieren. Die Suche nach Schlüsselwörtern und PDB Ids oder Namen.

1.1.14 Phylogenie

Phylogenie ist die Evolutionsgeschichte und Abstammungslinie einer Art oder einer höheren taxonomischen Gruppe. Das Ergebnis wird sehr oft in phylogenetischen Bäumen(siehe Abb. 1.5)[16] dargestellt.[17]

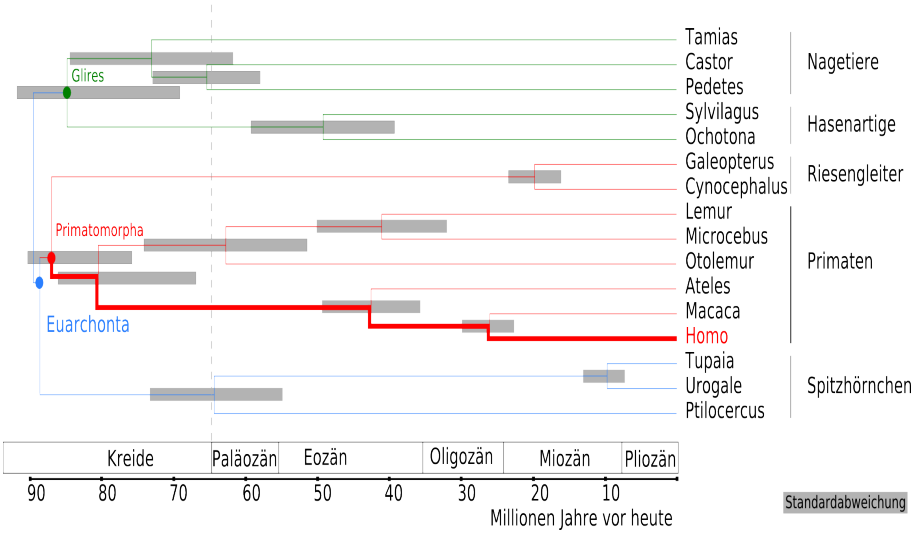

Abbildung 1.5: Phylogenetischer Baum[Janec]

Die Erzeugung von Bäumen findet über verschieden Algorithmen, wie zum Beispiel UPGMA/WPGMA oder Neighbor Joining statt. Das Ergebnis dieser Berechnung wird dann von Programmen in einen Baum transferiert. Je nach Programm lassen sich verschiedene Informationen zu den einzelnen Einträgen anzeigen.

[16] [9] entnommen S. 793
[17] [4] Vgl. S. 499

1.1.15 Proteinstruktur

Die Proteinstrukur wird in 4 Teile [18] eingeteilt: Primärstruktur, Sekundärstruktur, Tertiärstruktur und Quartärstruktur(siehe Abb.1.6)[19].
Die Primärstruktur umfasst dabei die Aminosäuresequenz der Peptidkette. In der Sekundärstruktur wird die räumliche Struktur eines lokalen Bereiches im Protein (z.B. α-Helix oder β-Faltblatt) dargestellt. Durch die Tertiärstruktur werden die räumlichen Strukturen der einzelnen Proteine bzw. Untereinheiten dargestellt und in der Quartärstruktur die räumliche Struktur des gesamten Protein mit allen Untereinheiten angezeigt.

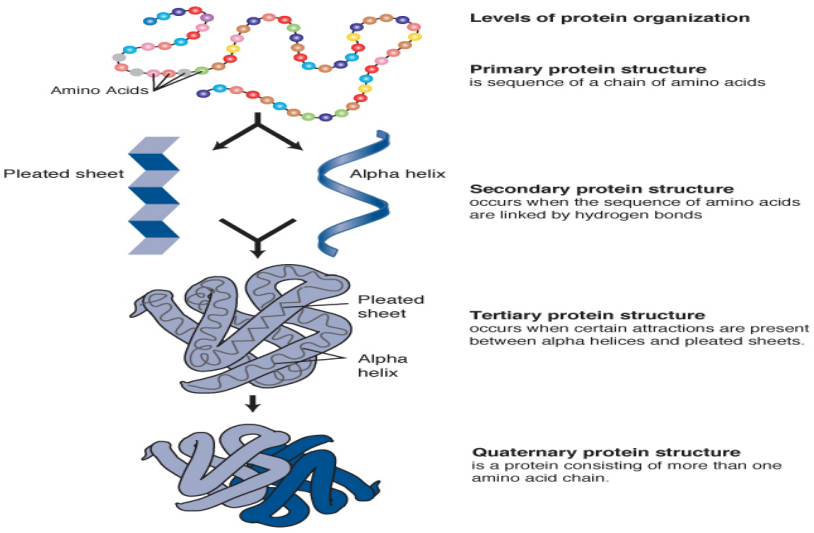

Abbildung 1.6: 4 Ebenenen der Proteinstruktur

1.1.16 Protein Modifikation

Die Protein Modifikation umfasst die pre-, co- und post-translationale Modifikation. Viele Arten der Modifikation werden mittels der RESID, PSI-MOD und der Protein Data Bank Chemical Component Dictrionary bereit gestellt. Modifikationen können zum Beispiel Glykosylierung oder Phosphorylierung sein.

[18] [10] Vgl. S.1ff
[19] entnommen [11]

1.1.17 GC Gehalt

Der GC Gehalt ist eine massgebliche Größe im Genom und schwankt zwischen 25% und 75% im Normalfall. In GC Paaren werden drei Wasserstoffbrückenbindungen eingegangen, wohin gegen es in AT Paaren nur zwei sind. Somit kann man schlussfolgern, dass ein hoher GC Gehalt ein stabilere Verbindung besitzt, als eine mit niedrigen Gehalt.[20]

$$GC[\%] = \frac{\sum G+C}{\sum G+C+A+T} * 100$$

Zur Bestimmung des GC-Gehaltes gibt es mehrere Möglichkeiten. Die Erste ist dabei das Auszählen der gesamten Sequenz und die Werte in die oben genannte Formel einsetzen. Der Vorteil dieser Methode ist, dass sie sehr schnell ist, jedoch auch ungenau. Variante zwei unterteilt sich in zwei Teilvarianten. Allgemein wird in dieser Methode mit einen Fenster über die Sequenz gegangen und die Werte werden sich gemerkt. Die normale Fensterlänge ist dabei 20. Die beiden Teilmethoden unterscheiden sich in der Schrittweise mit dem das Fenster über die Sequenz läuft. Bei der ersten Variante ist die Schrittweite gleich der Fensterlänge und bei der Zweiten gleich eins. Diese Methoden erfordern jedoch mehr Aufwand und geben in der ersten Variante auch ungenaue Werte bzw. extreme Peaks. Im nachfolgenden Beispiel wird dies mit allen drei Varianten dargestellt.

Sequenz:
ATGGTCCATGGCATGATCAGTAGCTGACTAATGGTCCATGGCATGATCAGTAGCTGACTA

Variante 1(ohne Fenster): 46.666666666666664 %
Variante 2a(mit Fenster, Schrittweite=Fensterlänge): 47.5%
Variante 2b(mit Fenster, Schrittweite=1): 47.375 %

[20] [12] Vgl. S. 8

1.1.18 Aminosäuren

Aminosäuren sind die Baugruppen der Proteine, dabei kommen nur 20 in den Proteinen regelmässig vor.[21] Es sind jedoch auch noch andere Aminosäuren(siehe Tabelle 1.2)[22] vorhanden.

Name/Bedeutung	Einbuchstabencode	Dreibuchstabencode
Alanin	Ala	A
Arginin	Arg	R
Asparagin	Asn	N
Asparaginsäure	Asp	D
Cystein	Cys	C
Glutamin	Gln	Q
Glutaminsäure	Glu	E
Glycin	Gly	G
Histidin	His	H
Isoleucin	Ile	I
Leucin	Leu	L
Lysin	Lys	K
Methionin	Met	M
Phenylalanin	Phe	F
Prolin	Pro	P
Serin	Ser	S
Threonin	Thr	T
Tryptophan	Trp	W
Tyrosin	Tyr	Y
Valin	Val	V
Selenocysteine	Sec	U
Pyrrolysine	Pyl	O
Asparagin oder Asparaginsäure	Asx	B
Glutamin oder Glutaminsäure	Glx	Z
Leucin oder Isoleucin	Xle	J
Nichtdefiniert oder unbekannte Aminosäure	Xaa	X

Tabelle 1.2: Aminosäuren

[21] [13] Vgl. Seite 56
[22] entnommen [14]

1.1.19 Newick Format

Das Newick Format ist eine Möglichkeit Graphen mittels Klammern, Kommas sowie Doppelpunkten darzustellen. Es wird die minimal Definition für einen phylogenetischen Baum erreicht. Interneknoten sind durch ein Paar von Klammern dargestellt. Knoten welche direkt von diesem Knoten abstammen werden durch Kommas getrennt. Der Name eines Knoten kann eine beliebige Zeichenfolge von Zeichen, ohne Leerzeichen, Doppelpunkte, Semikolons und Klammern sein. Jeder Name kann auch leer sein. Knoten können dabei auch mehr als zwei Kinder besitzen. Durch Doppelpunkte werden die Längen der jeweiligen Zweige dargestellt.[23]

Für die Transmission sollte das phyloXML Format[24] benutzt werden. Dieses Format basiert auf XML und beinhaltet Elemente unter anderen für taxonomische Werte oder Gennamen. Das phyloXML Format ist außerdem robuster.

1.1.20 Distanzmatrix mittels PID

Der Prozentsatz der Identität zwischen den beiden Sequenzen an jeder ausgerichteten Position, wird als PID bezeichnet. In diesen Verfahren werden jeweils die Werte über die unten stehende Formel berechnet.

$$PID = \frac{\text{Anzahl äquivalent ausgerichteten Symbole(ohne Lücken)} * 100}{\text{Anzahl der ausgerichteten Symbole(ohne Lücken)}}$$

[23] Vgl. [29] S. 126
[24] Vgl. [28]

1.1.21 Distanzmatrix mittels BLOSUM62

Die BLOSUM62 ist eine Substitutionsmatrix, welche eine 62%ige identische Übereinstimmung der Sequenzen hat. Durch die bereits ermittelten Werte(siehe Tabelle[25] 1.3) aus der BLOSUM62 wird die Distanzmatrix an den entsprechenden Stellen gefüllt.

C	9																			
S	-1	4																		
T	-1	1	4																	
P	-3	-1	1	7																
A	0	1	-1	-1	4															
G	-3	0	1	-2	0	6														
N	-3	1	0	-2	-2	0	6													
D	-3	0	1	-1	-2	-1	1	6												
E	-4	0	0	-1	-1	-2	0	2	5											
Q	-3	0	0	-1	-1	-2	0	0	2	5										
H	-3	-1	0	-2	-2	-2	1	1	0	0	8									
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5								
K	-3	0	0	-1	-1	-2	0	-1	1	1	-1	2	5							
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5						
I	-1	-2	-2	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4					
L	-1	-2	-2	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4				
V	-1	-2	-2	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4			
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6		
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7	
W	-2	-3	-3	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

Tabelle 1.3: BLOSUM62

[25] [15] entnommen S. 10915ff

1.1.22 Baumkonstruktionsalgorithmus mittels UPGMA

Die Unweighted Pair Group Method with Arithmetic mean(UPGMA) ist ein Clusterverfahren zur Ermittlung von phylogenetischen Bäumen. Es werden die beiden Sequenzen zusammengefasst, welche den kleinsten Abstand haben. Danach wird die Distanzmatrix neuberechnet und der weighted pair group method with averaging(WPGMA) bzw. UPGMA angewendet. Dieses Verfahren wird solange durchgeführt, bis nur noch zwei Sequenzen übrig sind.

Annahmen:
S_a, S_b, S_i sind Sequenzen a und b, sowie Sequenzen i
S_n ist neue Sequenz aus $S_a \cup S_b$
d_{S_i,S_i} Distanz

UPGMA:[26]

$$d_{S_n,S_i} = \frac{d_{S_a,S_i}+d_{S_b,S_i}}{2}$$

WPGMA:

$$d_{S_n,S_i} = \frac{|S_a|*d_{S_a,S_i}+|S_b|*d_{S_b,S_i}}{|S_a|+|S_b|}$$

1.1.23 Baumkonstruktionsalgorithmus mittels Neighbor Joining

Neighbour Joining ist ein Verfahren um zwei Gruppen mit einander zu vergleichen und diese anzuordnen. Dieser Algorithmus findet jedoch nicht immer den besten Baum, dies ist durch den Greedyalgorithmus bestimmt.

[26] [16] entnommen S. 1409ff

1.1.24 FATCAT Algorithmus

Der FATCAT Algorithmus ist ein flexibles Struktur Alignment, welches Aligned Fragment Pairs mit Twists verbindet. Er minimiert dabei die Drehungen um die Drehpunkte. Der Algorithmus erzeugt genauere Ausrichtungen als alle anderen Algorithmen.

1.1.25 CE Algorithmus

Der CE Algorithmus ist eine Methode zur Berechnung von paarweisen Ausrichtungen. Der Algorithmus richtet zwei Polypeptidketten über Merkmale ihrer lokalen Geometrie aus.

1.1.26 Strukturelles Alignment

Die Strukturelle Ausrichtung versucht, Äquivalenzen zwischen zwei oder mehr polymeren Strukturen zu finden, welche auf der Basis ihrer Form und dreidimensionale Konformation beruht. Im Gegensatz zu einfachen konstruktiven Überlagerung , wo zumindest einige äquivalenten Resten der beiden Strukturen bekannt sind, erfordert die strukturelle Ausrichtung keine Kenntnis von den entsprechenden Positionen. Strukturelle Ausrichtung ist ein wertvolles Werkzeug für den Vergleich von Proteinen mit geringer Sequenzähnlichkeit, wo evolutionären Beziehungen zwischen Proteinen nicht leicht durch Sequenz-Alignment Techniken nachgewiesen werden können. Daher kann es zur Vorhersage evolutionären Beziehungen zwischen Proteinen verwendet werden. Allerdings ist Vorsicht bei der Verwendung der Ergebnisse, als Beweis für einer gemeinsamen evolutionären Abstammung, geboten.

1.2 Informatik

1.2.1 SwingX

SwingX ist eine freie Bibliothek für die Erweiterung der grafischen Oberfläche unter Java. Sie beinhaltet unter anderen eine Erweiterung für Tabellen, Listen, TreeTables oder eine „Tip of the Day“- Komponente. Zur Darstellung von erweiterten Informationen wird diese Bibliothek im Tool verwendet.

1.2.2 XML

Die Extensible Markup Language(XML) ist eine Standard des W3C, sowie eine Metasprache. XML ist eine universelle Sprache zur Transmission und Darstellung von Inhalten. Der Einsatz ist in vielen Komponenten notwendig, um die Möglichkeit von unteranderen Multisprachunterstützung zu bieten.

1.2.3 mxGraph

Die freie Bibliothek „mxGraph“ ist eine Sammlung von Möglichkeiten zur Darstellung von Graphen. Es können unterschiedliche Arten von Graphen dargestellt werden. Die Darstellung phylogenetischer Bäume wird hier mit ermöglicht.

2 BioJava

2.1 Bestandteile

BioJava hat die sieben großen Gebiete Core, Genome, Phylogenie, Alignment, Protein Struktur, Protein Modifikation, Webservices. Jedes der einzelnen Gebiete ist für bestimmte Bereiche zuständig, so ist der Core für die Verarbeitung von u.a. RNA-, DNA- oder Proteinsequenzen zuständig.

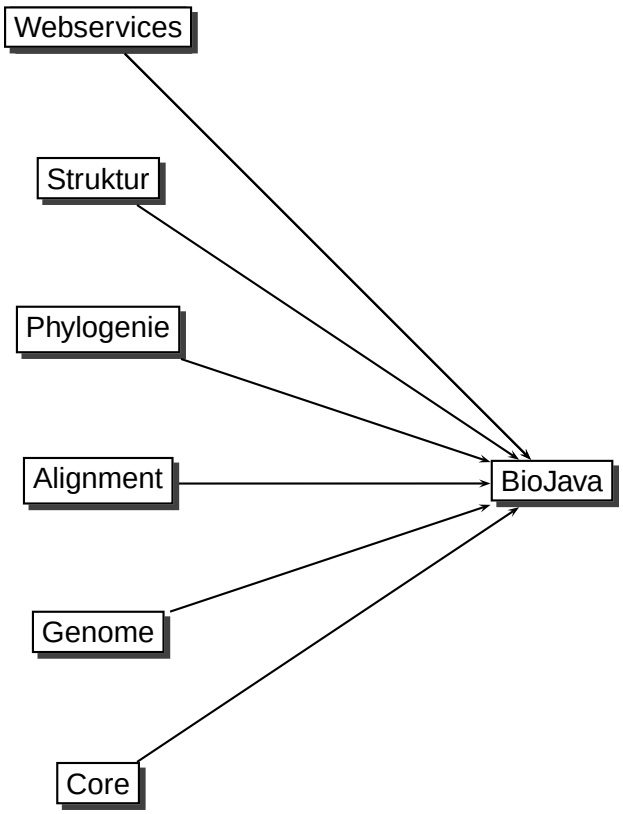

Abbildung 2.1: Bestandteile von BioJava 3.0.1

2.2 BioJava Core

Der Core ist das zentrale Element der BioJava-Bibliothek, denn die gemeinsamen Elemente für alle Module sind das Lesen, das Schreiben und die Darstellung der Sequenzdaten. Die Erstellung der Sequenzen soll für den Benutzer einfach und doch sicher sein. Die Bibliothek hat dabei den folgenden Aufbau für Sequenzen:[27]

- AbstractSequence
  - DNASequence
    - ChromosomeSequence
    - GeneSequence
    - IntronSequence
    - ExonSequence
    - TranscriptSequence
  - RNASequence
  - ProteinSequence

2.2.1 DNASequence

Die DNA Sequenz wurden von den Entwicklern der BioJava Bibliothek als eine Folge der Buchstaben „A“ , „T“ , „G“ , „C“ definiert. Sollten andere Buchstaben eingegeben werden, führt dies zu einen Fehler und mögliche Berechnungen sind nicht mehr möglich. Mit der DNASequence ist es machbar sich das Komplement, den GC Gehalt, das Reverse der Sequenz oder die RNA Sequenz berechnen und anzeigen zulassen.

ChromosomeSequence

Da die ChromosomeSequence von der DNASequence abgeleitet wird, sind die vier Buchstaben auch hier zugelassen. Bei anderen Buchstaben wird ebenfalls ein Fehler erzeugt. Zusätzlich zu den Eigenschaften und Berechnungen der DNASequence, verfügt die ChromosomeSequence noch über eine Chromosomnummer, die einzelnen Gene und Gensequenzen.

GeneSequence

Die GeneSequence ist ihrerseits von der ChromosomSequence abgeleitet und braucht neben der Sequenz auch noch den Startpunkt und den Endpunkt, sowie den Strand. Der Strand kann dabei undefiniert, negativ oder positiv sein. Neben den Möglichkeiten welche die GeneSequence von ihren Eltern bekommt, besitzt sie auch noch die Anzeige von Transcript-, Intron- und Exonsequenzen, sowie die Umwandlung auf 5’ $\rightarrow$ 3’.

[27] Vgl. [17]

IntronSequence

Als Kindsequenz der GeneSequence benötigt die Sequenz neben den Informationen der Elternsequenz auch noch den Start- und Endwert des Intronbereiches, bezogen auf den Bereich der GeneSequence. Die IntronSequence hat alle Eigenschaften und Methoden der Elternsequence geerbt.

ExonSequence

Ebenso, wie die IntronSequence, benötigt sie den Start- und Endwert des Exonbereiches. Die ExonSequence hat auch alle Eigenschaften und Methoden vererbt bekommen.

TranscriptSequence

Die TranscriptSequence wird genau wie die Intron- oder die ExonSequence erstellt. Diese Sequenz hat neben den Methoden und Eigenschaften der Elternsequenzen auch noch die Möglichkeiten der Erzeugung der Start-Codon und Stop-Codon Sequenzen.

2.2.2 RNASequence

In der RNA Sequenz sind die Buchstaben „A“ , „U“ , „G“ , „C“ für eine gültige Folge definiert, sollten andere Buchstaben eingegeben werden, führt dies auch hier zu einen Fehler und mögliche Berechnungen sind nicht mehr möglich. Sie besitzt die selben Möglichkeiten wie die DNASequence.

2.2.3 ProteinSequence

Als Proteinsequenz ist die eine Reihenfolge aus den Buchstaben „A“ , „R“ , „N“ , „D“ , „C“ , „E“ , „Q“ , „G“ , „H“ , „I“ , „L“ , „K“ , „M“ , „F“ , „P“ , „S“ , „T“ , „W“ , „Y“ , „V“ , „B“ , „Z“ , „J“ , „X“ , „U“ , „O“ ,“*“ und „-“ erlaubt. Sollten andere Zeichen vorkommen, wird ein Fehler angezeigt. Die ProteinSequence kann auch über die Accession ID oder die PDB ID aufgerufen werden. Die Eigenschaften wie Länge oder Taxonomy können abgerufen werden, falls diese eingegeben wurden sind.

2.3 BioJava Genome

Der Bereich „Genomme“ klassifiziert das Gebiet der Datei Ein- und Ausgabe, im Bezug auf die gtf, gff2/3 Formate.

2.3.1 gtf Dateien lesen

Es muss eine gültige gtf Datei vorhanden sein, welche dann eingelesen werden kann. Diese Datei wird dann umgewandelt in eine „FeatureList“, welche alle einzelnen Einträge der gtf Datei enthält. Diese List kann dann in ein Format der eigenen Wahl ohne grossen Aufwand transformiert werden.

2.3.2 gff 2 oder gff3 Dateien lesen

Das gff2 Format kann mit dem gff3 Reader gelesen werden und sollte auch damit behandelt werden. Als Ergebnis des Einlesens wird genau wie bei gtf eine „FeatureList“ erzeugt und kann in alle anderen Formate transformiert werden.

2.3.3 gff3 Dateien schreiben

Zum schreiben von gff3 Dateien ist eine Chromosomsequenzliste erforderlich. Diese Liste kann man sich aus einen Fasta File erzeugen lassen, welches die Chromosomdaten enthält.

2.4 BioJava Phylogenie

Die Konstruktion eines phylogenetischen Baumes erfordert die Eingabe von mindestens 2 Sequenzen. In der Berechnung wird zunächst ein Multisequenzalignment mittels PID oder BLOSUM62 berechnet und dies an den Baumkonstruktionsalgorithmus übergeben. Als Konstruktionsalgorithmen stehen Neighbour Joining oder UPGMA zur Verfügung. Ist die Berechnung abgeschlossen, wird ein Newick Format String zurück gegeben. Mit diesem Ergebnis kann der Benutzer sich einen Baum zum Beispiel zeichnen lassen.

2.5 BioJava Alignment

Das Alignment kann mittels BioJava in 2 Gruppen unterteilt werden. Das Paarweise Alignment, mit Smith Waterman und Neddleman Wunsch, ist für den Vergleich von 2 Sequenzen vorgesehen. Sollte man mehr als 2 Sequenzen mit einander alignieren wollen, so muss man das Multiple Sequence Alignment benutzen.

2.5.1 Paarweises Alignment

Das Paarweise Alignment unterteilt sich in das Globale und Lokale Alignment. Das globale Alignment wird dabei über den Smith Waterman Algorithmus bestimmt, wohingegen das lokale Alignment über den Needleman Wunsch Algorithmus ermittelt wird.

Smith Waterman

Das globale paarweise Alignment benötigt ein Array über die Beiden zu vergleichenden Sequenzen. Diese Sequenzen müssen vom selben Typ sein. Neben den Sequenzen muss eine Substitutionsmatrix vorhanden sein. Diese Matrix wird benötigt, um alle Werte darin einzutragen. Als Ergebnis erhält man ein „SequencePair“, welches die beiden alignierten Sequenzen und andere Informationen, wie die Anzahl der ähnlichen oder identischen Komponenten.

Needleman Wunsch

Im lokalen paarweisen Alignment ist ebenfalls ein Array der Sequenzen erforderlich, eben so wie die Substitutionsmatrix. Auch hier wird ein „SequencePair“ zurück gegeben, mit den selben Eigenschaften wie bei Smith Waterman.

2.5.2 Multiples Sequence Alignment

Die Analyse von mehr als zwei Sequenzen erfordert ein Multiples Sequence Alignment(MSA). Um dies mittels BioJava zu realisieren ist als erstes eine Liste der zu alignierenden Sequenzen erforderlich. Als Ergebnis bekommt der Benutzer ein „Profile“. Mittels dem „Profile“ kann sich der Benutzer die Sequenzen in Rohform(ohne Alignierung) oder aligniert ansehen, zu jeder einzelnen Sequenz können auch noch andere Information, wie Anzahl der Lücken oder Überlappungen, abgerufen werden.

2.6 BioJava Protein Struktur

Das Paket Protein Struktur ist für die Verarbeitung von mmCIF und PDB Dateien zuständig. Die Verarbeitung der Dateien ist dabei sehr einfach gehalten und erfordert vom Benutzer kaum ein Eingreifen.

2.6.1 PDB Datei verarbeiten

Das Einlesen der PDB Datei erfolgt mittels dem „PDBFileReader“. In diesen Reader muss zwingend der Parameter „setAlignSeqRes“ über die „setFileParsingParameters“ gesetzt werden, denn sonst kann es sein, dass nicht alle PDB Dateien eingelesen werden können. Nach dem erfolgreichen Lesen der Datei erhält der Benutzer eine „Structure“, in welchen Informationen u.a. zu den einzelnen Ketten, Datenbankreferenzen oder SSBonds stehen. Diese können vom Benutzer dann verarbeitet werden.

2.6.2 mmcif Datei verarbeiten

Der Umgang mit der mmCIF Datei ist etwas einfacher, als der Umgang mit der PDB Datei. Im Gegensatz zum PDB File ist beim mmCIF kein setzen der Parameter nötig.

2.6.3 Zugriff auf Atome in einer Struktur

Der Zugriff auf die einzelnen Atome der Struktur erfolgt entweder über die „StructureTools“ oder die direkte Abfrage der Atome über Structure→Model→AtomGroup→Atom.

2.6.4 Berechnungen mit Atomen

Mit der „Calc“-Klasse lassen sich Berechnungen, wie die Bestimmung der Torsionswinkel ϕ und ψ bestimmen oder auch das Zentroid der Atomgruppe.

2.6.5 Arbeiten mit Gruppen

Die Bibliothek unterteilt in die 3 Gruppen AminoAcid, Nucleotide und Hetatom. Diese Unterteilung erfolgte in Anlehnung an die entsprechenden Formate zur Darstellung.

2.6.6 Zugriff auf die Headerinformationen in der PDB Datei

In der PDB Datei können ebenfalls die Headerinformationen abgefragt werden. Inhalte des Header können u.a. die biologische Einheit, Zelllinie, zellulare Lokalisierung oder die Taxonomie ID sein.

2.6.7 Umgang mit SEQRES und ATOM Gruppen

Die SEQRES Datensätze in einer PDB-Datei enthalten die Aminosäure- oder Nukleinsäuresequenz von Rückständen in jeder Kette des Moleküls, welches untersucht wurde. Die ATOM Gruppen liefern die Koordinaten für die Rückstände.

2.6.8 Residue mutieren

Es können auch Mutationen simuliert werden, in dem man sich eine neue Struktur aus einer alten schafft. Es können verschiedene Dinge mutiert werden, wie zum Beispiel verschiedene Stellen in der Sequenz die getauscht werden.

2.6.9 Berechnung eines strukturierten Alignments

Zur Berechnung des strukturierten Alignments stehen der FATCAT und der CE Algorithmus zur Verfügung. Die beiden Algorithmen werden jedoch über das „CE“ Tool betrachtet.

2.6.10 SCOP Klassifikation laden

Der SCOP Parser kann die SCOP Dateien downloaden und analysieren. Es werden nur die entsprechenden Information für eine Struktur herunter geladen, falls die Informationen nicht schon lokal verfügbar sind.

2.7 BioJava Protein Modifikation

Das Finden von Modifikationen in Proteinen ist durch die BioJava-Bibliothek ermöglicht. Das Scannen nach diesen Modifikationen erfolgt über ein Structure-Objekt. In dieses Objekt kann nach der ID, RESID ID, PSI-MOD ID, PDBCC ID, Kategorie, involvierte Komponenten, Schlüsselwörtern oder der Art des Auftretens gesucht werden.

2.7.1 Unterstützte Proteinmodifikationen

In der aktuellen Version der Bibliothek können über die ID, RESID ID und PSI-MOD ID ca. 250 Datensätze angesprochen werden. Die PDBCC ID gibt in etwas das Doppelte an Datensätzen zurück. Die Kategorien sind dabei „attachment“, „modified residue“, „crosslink1“, „crosslink2“, ..., „crosslink7“. Unter den Schlüsselwörter sind u.a. glycoprotein, phosphoprotein, sulfoprotein, ... eingetragen. Die Art des Auftretens kann man in „natural“ oder „hypothetical“ unterteilen. Eine genau Auflistung aller Modifikationen ist unter A nach zu lesen.

2.7.2 Definieren und Registrierung von neuen Proteinmodifikationen

Durch die eigene Definition und Registrierung von Modifikationen ist es möglich sich eigene Filter zu definieren, falls durch die Grundeinstellungen nicht die gewünschte Modifikation mitgeladen wurde. Die Definition einer neuen Modifikation erfordert vom Benutzer gute Kenntnisse im Zusammenhang mit der RESID ID, PSI-MOD und der PSI-MOD ID, da diese IDs angegeben werden sollten.

2.8 BioJava Webservices

Die Webservices von BioJava unterscheiden sich nur in den 5 Blastarten: blastn, blastp, blastx, tblastn und tblastx. Da dies nicht genug ist, wurde vom Entwickler die Möglichkeit geschaffen auch die PDB mittels eines Webservices abzufragen.

2.8.1 NCBI Blast Services

Wie schon erwähnt unterscheidet BioJava nur die fünf Blastarten. Das Anbinden des Webservices erfordert jedoch sehr viel Nacharbeit, dies liegt zum einen an der Bibliothek selber als auch an der NCBI, an der aktuell gearbeitet wird. Durch die Bibliothek kann man nicht zu 100% sicher gehen, dass die Anfrage erfolgreich gesendet wird.

2.8.2 PDB Services

Der PDB Services stammt nicht aus der BioJava Bibliothek selbst, sondern von der PDB direkt und funktioniert dem entsprechend auch ohne BioJava. Die Einbindung der Bibliothek in ein System ist sehr einfach und erfordert keine großen Kenntnisse. Dieser Webservice ist im Gegenzug zum NCBI Blast Service sehr fehlerresistent.

3 BioJava Bestandteile Java

3.1 BioJava Core

3.1.1 DNASequence

Es muss einfach eine neues Objekt vom Typ DNASequence erzeugt werden, diese Erzeugung ist über fünf Möglichkeiten machbar, von denen aber nur vier benutzt werden sollten. Die nachfolgenden Listings(3.2 - 3.8) zeigen die fünf Konstruktoren und die Abfrage der Eigenschaften.

```
1 DNASequence seq_Default = new DNASequence();
2 DNASequence seq_String = new DNASequence(String seqString);
3 DNASequence seq_Proxy = new DNASequence(SequenceReader<
      NucleotideCompound> proxyLoader);
4 DNASequence seq_String_CompoundSet = new DNASequence(String
      seqString, CompoundSet<NucleotideCompound> compoundSet);
5 DNASequence seq_Proxy_CompoundSet = new DNASequence(SequenceReader<
      NucleotideCompound> proxyLoader,        CompoundSet<
      NucleotideCompound> compoundSet)
```

Listing 3.1: DNA Sequenz erzeugen

Default

Diese Variante der Erzeugung einer DNA Sequenz sollte nicht benutzt werden, da sie, ohne die richtige Einstellung der Parameter, Fehlermeldungen wirft. Das Erstellen erfolgt dabei mit dem Standardkonstruktor(Listing 3.2 Nr. 1).

```
1 DNASequence seq_Default = new DNASequence();
```

Listing 3.2: DNA Sequenz erzeugen - Default Variante

String

Die Methode welche am schnellsten und sichersten zum Erfolg führt. Die Erzeugung der Sequenz erfolgt dabei über einen Kontruktor(Listing 3.3 Nr. 2), welcher eine Zeichenkette, welche äquivalent zur biologischen Sequenz ist, übergeben bekommt.

```
1 String seq = "ATGC";
2 DNASequence seq_String = new DNASequence(seq);
```

Listing 3.3: DNA Sequenz erzeugen - String Variante

Proxy

Diese Variante wird benutzt um eine Sequenz zu erstellen, deren Daten anderswo gespeichert sind. Um die Sequenz erzeugen zulassen muss als erstes ein ProxyLoader definiert werden(Listing 3.4 Nr. 1) und dieser dann dem entsprechenden Konstruktor(Listing 3.4 Nr. 2) übergeben werden.

```
1  StringProxySequenceReader<NucleotideCompound>
       sequenceStringProxyLoader = new StringProxySequenceReader<
       NucleotideCompound>("ATGC", DNACompoundSet.getDNACompoundSet());
2          DNASequence seq_Proxy = new DNASequence(
               sequenceStringProxyLoader);
```

Listing 3.4: DNA Sequenz erzeugen - Proxy Variante

String und CompoundSet

Diese Variante, dient zur Erstellung einer Sequenz mittels einen benutzerdefinierten CompoundSet(Listing 3.5). Im CompoundSet müssen alle abstrakten Methoden implementiert sein. Dem Konstruktor(Listing 3.6 Nr. 2) muss deshalb eine Zeichenkette und das CompoundSet übergeben werden.

```
1  import org.biojava3.core.sequence.template.
       AbstractNucleotideCompoundSet;
2  public class UserCompoundSet extends AbstractNucleotideCompoundSet<
       NucleotideCompound> {
3    private static class InitaliseOnDemand {
4      public static final UserCompoundSet INSTANCE = new
           UserCompoundSet();
5    }
6    public static UserCompoundSet getUserCompoundSet() {
7      return InitaliseOnDemand.INSTANCE;
8    }
9    public UserCompoundSet() {
10     addNucleotideCompound("A", "T");
11     addNucleotideCompound("T", "A");
12     addNucleotideCompound("G", "C");
13     addNucleotideCompound("C", "G");
14     addNucleotideCompound("N", "N");
15     addNucleotideCompound("-", "-");
16   }
17
18   public NucleotideCompound newNucleotideCompound(String base,
         String complement, String... equivalents) {
19     if(equivalents.length == 0) {
20       return new NucleotideCompound(base, this, complement);
21     }
22     else {
23       NucleotideCompound[] compounds = new NucleotideCompound[
             equivalents.length];
24       for(int i=0; i<compounds.length; i++) {
25         compounds[i] = getCompoundForString(equivalents[i]);
26       }
27       return new NucleotideCompound(base, this, complement,
             compounds);
28     }
29   }
30 }
```

Listing 3.5: CompoundSet erzeugen

```
1 String seq ="ATGC";
2 DNASequence seq_String_CompoundSet = new DNASequence(seq,
      UserCompoundSet.getUserCompoundSet());
```

Listing 3.6: DNA Sequenz erzeugen - String + CompoundSet Variante

Proxy und CompoundSet

Diese Variante, dient zur Erstellung einer Sequenz mittels einen benutzerdefinierten CompoundSet(Listing 3.5). Dem Konstruktor(Listing 3.7 Nr. 2) muss deshalb ein ProxyLoader und das CompoundSet übergeben werden.

```
1 StringProxySequenceReader<NucleotideCompound>
      sequenceStringProxyLoader = new StringProxySequenceReader<
      NucleotideCompound>("GCTA", DNACompoundSet.getDNACompoundSet())
      ;
2 DNASequence dnaSequenceFromProxy = new DNASequence(
      sequenceStringProxyLoader, UserCompoundSet.getUserCompoundSet());
```

Listing 3.7: DNA Sequenz erzeugen - Proxy + CompoundSet Variante

Eigenschaften und Methoden

Die DNA Sequenz verfügt dabei über neun wichtige Methoden(Listing 3.8 Nr. 3 - 11). Die Transkription der DNA in die RNA spielt dabei eine sehr grosse Rolle. Die erste Transkriptionsmethode gibt die RNA Sequenz äquivalent zur DNA, mit Hilfe der Defaulteinstellungen, zurück. Man sollte beachten, dass diese Methode nicht für alle Spezies gültig ist. In der zweiten Methode kann man die TranscriptionsEngine einstellen. Die nächste Methode erlaubt es einen von sechs Frames(ONE,TWO,THREE, REVERSED_ONE,REVERSED_TWO ,REVERSED_THREE) zu benutzen. Die letzte Möglichkeit ist die Kombination aus der zweiten und dritten Methode. Die Bestimmung der Anzahl von GC in der Sequenz ist mit der Methode „getGCCount“ möglich. Die Methoden(Listing 3.8 Nr. 8,9,10) erlauben es die Umkehrung, das Komplement und das umgekehrte Komplement zu bestimmen. Der DNAType kann CHROMOSOME, MITOCHONDRIAL, PLASMID, PLASTID oder UNKNOWN sein und kann gelesen als auch gesetzt werden.

```
1  String seq = "ATGC";
2  DNASequence seq_String = new DNASequence(seq);
3  seq_String.getRNASequence();
4  seq_String.getRNASequence(TranscriptionEngine engine);
5  seq_String.getRNASequence(Frame frame);
6  seq_String.getRNASequence(TranscriptionEngine engine, Frame frame);
7  seq_String.getGCCount();
8  seq_String.getReverse();
9  seq_String.getComplement();
10 seq_String.getReverseComplement();
11 seq_String.getDNAType();
12 seq_String.setDNAType(DNASequence.DNAType dnaType);
```

Listing 3.8: DNA Sequenz Eigenschaften

ChromosomeSequence

Es muss einfach eine neues Objekt vom Typ ChromosomeSequence erzeugt werden, diese Erzeugung ist ebenfalls über fünf Möglichkeiten machbar, von denen aber auch nur vier benutzt werden sollten, genau wie in der DNA Sequenz. Die Erzeugung der Sequenzen ist äquivalent zu den Listings(3.2 - 3.8) möglich, der Unterschied besteht nur dadurch, dass DNASequence durch ChromosomeSequence getauscht wird.

```
1 ChromosomeSequence seq_Default = new ChromosomeSequence();
2 ChromosomeSequence seq_String = new ChromosomeSequence(
      String seqString);
3 ChromosomeSequence seq_Proxy = new ChromosomeSequence(
      SequenceReader<NucleotideCompound> proxyLoader);
4 ChromosomeSequence seq_String_CompoundSet = new
      ChromosomeSequence(String seqString, CompoundSet<
      NucleotideCompound> compoundSet);
5 ChromosomeSequence seq_Proxy_CompoundSet = new
      ChromosomeSequence(SequenceReader<NucleotideCompound>
      proxyLoader, CompoundSet<NucleotideCompound>
      compoundSet)
```

Listing 3.9: Chromosom Sequenz erzeugen

Die Eigenschaften der ChromosomSequence sind im Listing3.10 dargestellt. Sie umfassen das Hinzufügen von Genen, das Auslesen der Chromosomnummer, der Gene und der Gensequenzen, das Löschen von Gensequenzen und das Setzen der Chromosomnummer.

```
1 String seq = "ATGC";
2 ChromosomeSequence seq_String = new ChromosomeSequence(seq);
3 seq_String.addGene(AccessionID accession, int bioBegin, int bioEnd,
      Strand strand);
4 seq_String.getChromosomeNumber();
5 seq_String.getGene(String accession);
6 seq_String.getGeneSequences();
7 seq_String.removeGeneSequence(String accession);
8 seq_String.setChromosomeNumber(int chromosomeNumber);
```

Listing 3.10: Chromosom Sequenz Eigenschaften

GeneSequence

Die Konstruktion einer GeneSequence erfordert eine Elternsequenz vom Typ ChromosomeSequence, den Beginn, das Ende und den Strand. Der Strand unterteilt sich in die Möglichkeiten NEGATIVE, POSITIVE oder UNDEFINED.

```
1 GeneSequence seq = new GeneSequence(ChromosomeSequence
      parentSequence, int begin, int end, Strand strand);
```

Listing 3.11: Gen Sequenz erzeugen

Die nachfolgenden Methoden(Listing 3.12) ermöglichen es den Benutzer Exons, Introns oder Transcript Elemente der GeneSequence hinzuzufügen, auszulesen oder zu löschen. Die Länge, der Strand und die Elternchromosomsequenz können einfach ausgelesen werden.

```
1  addExon(AccessionID accession, int begin, int end)
2  addIntron(AccessionID accession, int begin, int end)
3  addIntronsUsingExons()
4  addTranscript(AccessionID accession, int begin, int end)
5  getExonSequences()
6  getIntronSequences()
7  getLength()
8  getParentChromosomeSequence()
9  getSequence5PrimeTo3Prime()
10 getStrand()
11 getTranscript(String accession)
12 getTranscripts()
13 removeExon(String accession)
14 removeIntron(String accession)
15 removeTranscript(String accession)
16 setStrand(Strand strand)
```

Listing 3.12: Gen Sequenz Eigenschaften

IntronSequence

Zur Erstellung einer IntronSequence wird eine GeneSequence benötigt, sowie das Ende und der Start dieser.

```
1 IntronSequence(GeneSequence parentGeneSequence, int begin, int end)
```

Listing 3.13: Intron Sequenz erzeugen

Die einzige wichtige Eigenschaft der IntronSequence ist die Länge.

```
1 getLength()
```

Listing 3.14: Intron Sequenz Eigenschaften

ExonSequence

Da im biologischen die Intron und Exon Sequenzen sehr eng mit einander interagieren, wird dies auch hier gemacht. Der Konstruktor benötigt, genau wie der bei der IntronSequence nur die als Elternsequenz übergebene GeneSequence, den Start und das Ende für den Exonbereich.

```
1 ExonSequence(GeneSequence parentGeneSequence, int bioBegin, int
      bioEnd)
```

Listing 3.15: Exon Sequenz erzeugen

Auch hier ist die wichtigste Methode, die Ermittlung der Länge.

```
1 getLength()
```

Listing 3.16: Exon Sequenz Eigenschaften

TranscriptSequence

Die TranscriptSequence ist der dritte Teil um die GeneSequence, auch dieser Konstruktor benötigt die GeneSequence und den Start, sowie das Ende.

```
1 TranscriptSequence(GeneSequence parentDNASequence, int begin, int
      end)
```

Listing 3.17: Transcript Sequenz erzeugen

Im Gegensatz zur IntronSequence bzw. ExonSequence besitzt die TranscriptSequence mehr Eigenschaften, wie das Hinzufügen und Auslesen von kodierten Sequenz Regionen(CDS), die Bestimmung der Länge oder das Auslesen der Proteinsequenz, Startcodonsequenz und Stopcodonsequenz.

```
1  addCDS(AccessionID accession, int begin, int end, int phase)
2  addStartCodonSequence(AccessionID accession, int begin, int end)
3  addStopCodonSequence(AccessionID accession, int begin, int end)
4  getCDSSequences()
5  getDNACodingSequence()
6  getLength()
7  getProteinCDSSequences()
8  getProteinSequence()
9  getProteinSequence(TranscriptionEngine engine)
10 getStartCodonSequence()
11 getStopCodonSequence()
12 getStrand()
13 removeCDS(String accession)
```

Listing 3.18: Transcript Sequenz Eigenschaften

3.1.2 RNASequence

Die RNA Sequenz lässt sich über 4 Konstruktoren erzeugen, wobei der Konstruktor über eine Zeichenkette die einfachste Variante darstellt. Das Erstellen der Sequenz über den ProxyLoader und das CompoundSet ist ebenfalls möglich.

```
1 RNASequence(ProxySequenceReader<NucleotideCompound> proxyLoader)
2 RNASequence(ProxySequenceReader<NucleotideCompound> proxyLoader,
      CompoundSet<NucleotideCompound> compoundSet)
3 RNASequence(String seqString)
4 RNASequence(String seqString, CompoundSet<NucleotideCompound>
      compoundSet)
```

Listing 3.19: RNA Sequenz erzeugen

Ein wichtige Eigenschaft, ist das Auslesen der Anzahl von GC, welche benutzt werden kann um den GC-Gehalt zu bestimmen. Die Ermittlung des Komplements, der Inversen, sowie der Proteinsequenz ist ebenfalls möglich.

```
1 getComplement()
2 getGC()
3 getInverse()
4 getProteinSequence()
5 getProteinSequence(TranscriptionEngine engine)
6 getReverseComplement()
```

Listing 3.20: RNA Sequenz Eigenschaften

3.1.3 ProteinSequence

Auch bei der Proteinsequenz gibt es die vier bekannten Konstruktoren über eine Zeichenkette, CompoundSet oder ProxyLoader.

```
1 ProteinSequence(ProxySequenceReader<AminoAcidCompound> proxyLoader)
2 ProteinSequence(ProxySequenceReader<AminoAcidCompound> proxyLoader,
      CompoundSet<AminoAcidCompound> compoundSet)
3 ProteinSequence(String seqString)
4 ProteinSequence(String seqString, CompoundSet<AminoAcidCompound>
      compoundSet)
```

Listing 3.21: Protein Sequenz erzeugen

Ein wichtige Methode ist das Setzen der Elternsequenz für die Proteinsequenz

```
1 setParentDNASequence(AbstractSequence parentDNASequence, Integer
      begin, Integer end)
```

Listing 3.22: Protein Sequenz Eigenschaften

3.2 BioJava Genome

3.2.1 gtf Dateien lesen

Das Lesen von gtf Dateien funktioniert am besten über den GeneMarkGTFReader. Das Einlesen der gtf Datei erzeugt dabei eine FeatureList(siehe Listing 3.24) mit den Einträgen aus dem gtf.

```
1  GeneMarkGTFReader gtfReader = new GeneMarkGTFReader();
2  FeatureList featureList = gtfReader.read(file.getAbsolutePath());
```

Listing 3.23: gtf Dateien lesen

```
1  add(Collection<FeatureI> list)
2  add(FeatureI feature)
3  attributeValues(String key)
4  bounds()
5  groupValues()
6  hasAttribute(String key)
7  hasAttribute(String key, String value)
8  hasGaps(int gapLength)
9  omitOverlapping(String seqname, Location location, boolean
       useBothStrands)
10 selectByAttribute(String key)
11 selectByAttribute(String key, String value)
12 selectByGroup(String groupid)
13 selectByType(String type)
14 selectByUserData(String key)
15 selectByUserData(String key, Object value)
16 selectOverlapping(String seqname, Location location, boolean
       useBothStrands)
17 sortByStart()
18 splice(DNASequence sequence)
19 toString()
```

Listing 3.24: FeatureList Eigenschaften

3.2.2 gff2 oder gff3 Dateien lesen

Der Umgang mit gff2 oder gff3 Dateien erfolgt über den GFF3Reader. Auch hier wird eine FeatureList(siehe Listing 3.24) zurückgegeben.

```
1  GFF3Reader gff3Reader = new GFF3Reader();
2  FeatureList featureList = gff3Reader.read(file.getAbsolutePath());
```

Listing 3.25: gff2 oder Dateien lesen

3.2.3 gff3 Dateien schreiben

Das Erstellen der gff3 Dateien ist etwas schwieriger, als das Lesen. Man benötigt zum Schreiben eine ChromosomeSequenceList. Diese Liste wird dann an den GFF3Writer übertragen und in das gewählte File geschrieben.

```
1 LinkedHashMap<String, ChromosomeSequence> chromosomSequenceList =
      GeneFeatureHelper.getChromosomeSequenceFromDNASequence(
      FastaReaderHelper.readFastaDNASequence(new File("sequence.fasta"
      )));
2 FileOutputStream fo = new FileOutputStream(file);
3 GFF3Writer gff3Writer = new GFF3Writer();
4 gff3Writer.write(fo, chromosomSequenceList);
5 fo.close();
```

Listing 3.26: gff3 Dateien schreiben

3.3 BioJava Alignment

3.3.1 Paarweises Alignment

Smith Waterman

Durch die Übergabe eines Arrays von Proteinsequenzen kann ein globales Alignment berechnet werden. Eine einfache Substitutionsmatrix kann einfach erstellt werden. Als Rückgabe erhält man ein SequencePair, welches die beiden Sequenzen in der Rohform und in der alignierten Form enthält.

```
1 public SequencePair<ProteinSequence, AminoAcidCompound>
      calculateGlobalAlignment(ProteinSequence[] proteinSequence){
2         SubstitutionMatrix<AminoAcidCompound> matrix = new
              SimpleSubstitutionMatrix<AminoAcidCompound>();
3         SequencePair<ProteinSequence, AminoAcidCompound> pair =
              Alignments.getPairwiseAlignment(proteinSequence[0],
              proteinSequence[1], PairwiseSequenceAlignerType.GLOBAL,
              new SimpleGapPenalty(), matrix);
4         return pair;
5 }
```

Listing 3.27: Paarweises Alignment Smith Waterman

Needleman Wunsch

Das lokale Alignment ist genau so aufgebaut wie das Globale, nur mit den Unterschied das der PairwiseSequenceAlignerType auf LOCAL gestellt wird.

```
1 public SequencePair<ProteinSequence, AminoAcidCompound>
      calculateLocalAlignment(ProteinSequence[] proteinSequence){
2         SubstitutionMatrix<AminoAcidCompound> matrix = new
              SimpleSubstitutionMatrix<AminoAcidCompound>();
3         SequencePair<ProteinSequence, AminoAcidCompound> pair =
              Alignments.getPairwiseAlignment(proteinSequence[0],
              proteinSequence[1], PairwiseSequenceAlignerType.LOCAL,
              new SimpleGapPenalty(), matrix);
4         return pair;
5     }
```

Listing 3.28: Paarweises Alignment Needleman Wunsch

3.3.2 Multiples Sequence Alignment

Für das MSA ist eine ArrayList mit allen zu alignierenden Sequenzen erforderlich. Um das System sauber zu verlassen, sollte der Befehl „ConcurrencyTools.shutdown();“ mit angegeben werden, er bewirkt, dass die gegeben falls geöffneten Prozesse geschlossen werden.

```
1 public Profile<ProteinSequence, AminoAcidCompound> calculateMSA(
      ArrayList<ProteinSequence> lst){
2         Profile<ProteinSequence, AminoAcidCompound> alignment =
              Alignments.getMultipleSequenceAlignment(lst);
3         ConcurrencyTools.shutdown();
4         return alignment;
5     }
```

Listing 3.29: Multiples Sequence Alignment

3.4 BioJava Phylogenie

Die Phylogenie besteht darin, dass eine Newickzeichenkette erstellt wird. Mit dieser Zeichenkette kann dann ein Baum gezeichnet werden. Zur Kalkulation der Distanzmatrix, wird eine modifizierte Variante(Listing 3.31) der eigentlichen Methode genommen, da es zu Fehlern(siehe Kapitel 4.2) gekommen ist.

```
1  public String calculateNewickString(ArrayList<String> sequences) {
2          try {
3              ArrayList<ProteinSequence> proteinSequences = new
                   ArrayList<ProteinSequence>();
4              for (String sequence : sequences) {
5                  proteinSequences.add(core.createProteinSequence(
                       sequence));
6              }
7              Profile<ProteinSequence, AminoAcidCompound> msa = align
                   .calculateMSA(proteinSequences);
8
9              MultipleSequenceAlignment<ProteinSequence,
                   AminoAcidCompound> multipleSequenceAlignment = new
                   MultipleSequenceAlignment<ProteinSequence,
                   AminoAcidCompound>();
10             List<AlignedSequence<ProteinSequence, AminoAcidCompound
                   >> alSeq = msa.getAlignedSequences();
11             Sequence<AminoAcidCompound> seq;
12             ProteinSequence pSeq;
13             for (int i = 0; i < alSeq.size(); i++) {
14
15                 seq = alSeq.get(i);
16                 pSeq = new ProteinSequence(seq.getSequenceAsString
                       (), seq.getCompoundSet());
17                 pSeq.setAccession(seq.getAccession());
18                 multipleSequenceAlignment.addAlignedSequence(pSeq);
19
20             }
21
22
23             DistanceMatrix distmatrix = calc.start(
                   multipleSequenceAlignment, TreeType.AV,
                   TreeConstructionAlgorithm.PID, null);
24             TreeConstructor<ProteinSequence, AminoAcidCompound>
                   treeConstructor = new TreeConstructor<
                   ProteinSequence, AminoAcidCompound>(distmatrix,
                   TreeType.NJ, TreeConstructionAlgorithm.PID, new
                   ProgessListenerStub());
25             treeConstructor.process();
```

```
26              String newick = treeConstructor.getNewickString(true,
                    true);
27              return newick;
28          } catch (Exception ex) {
29              Logger.getLogger(Phylogenomics.class.getName()).log(
                    Level.SEVERE, null, ex);
30              return null;
31          }
32      }
```

Listing 3.30: Phylogenie

Die modifizierte Variante der Distanzmatrix behebt einige Fehler, welche zur Nichtberechnung führten. Die Berechnung der Matrix erfolgt über der Methode „calculateDinstanceMatrix“

```
1  public class calculate {
2
3      boolean verbose = false;
4      Phylogeny p = null;
5      DistanceMatrix matrix = null;
6      DistanceMatrix copyDistanceMatrix = null;
7      TreeType treeType;
8      TreeConstructionAlgorithm treeConstructionAlgorithm;
9      NJTreeProgressListener treeProgessListener;
10     MultipleSequenceAlignment<ProteinSequence, AminoAcidCompound>
           multipleSequenceAlignment = new MultipleSequenceAlignment<
           ProteinSequence, AminoAcidCompound>();
11
12     private double[][] calculateDistanceMatrix(
           MultipleSequenceAlignment<ProteinSequence, AminoAcidCompound
           > multipleSequenceAlignment, TreeConstructionAlgorithm tca)
           {
13          //updateProgress("Determing Distances", 0);
14          int numberOfSequences = multipleSequenceAlignment.getSize()
                ;
15          String[] sequenceString = new String[numberOfSequences];
16          for (int i = 0; i < multipleSequenceAlignment.getSize(); i
                ++) {
17              sequenceString[i] = multipleSequenceAlignment.
                    getAlignedSequence(i + 1).getSequenceAsString();
18
19          }
20
21
22          double[][] distance = new double[numberOfSequences][
                numberOfSequences];
23
```

```
24          int totalloopcount = (numberOfSequences / 2) * (
                numberOfSequences + 1);
25
26          if (tca == TreeConstructionAlgorithm.PID) {
27              int loopcount = 0;
28              for (int i = 0; i < (numberOfSequences - 1); i++) {
29                  //updateProgress("Determining Distances", (
                        loopcount * 100) / totalloopcount);
30                  for (int j = i; j < numberOfSequences; j++) {
31                      loopcount++;
32                      if (j == i) {
33                          distance[i][i] = 0;
34                      } else {
35                          distance[i][j] = 100 - Comparison.PID(
                                sequenceString[i], sequenceString[j]);
36
37                          distance[j][i] = distance[i][j];
38                      }
39                  }
40              }
41          } else {
42              // Pairwise substitution score (with no gap penalties)
43              ScoreMatrix pwmatrix = ResidueProperties.getScoreMatrix
                    (treeConstructionAlgorithm.name());
44              if (pwmatrix == null) {
45                  pwmatrix = ResidueProperties.getScoreMatrix(
                        treeConstructionAlgorithm.BLOSUM62.name());
46              }
47              int maxscore = 0;
48              int end = sequenceString[0].length();
49              int loopcount = 0;
50              for (int i = 0; i < (numberOfSequences - 1); i++) {
51                  // updateProgress("Determining Distances", (
                        loopcount * 100) / totalloopcount);
52                  for (int j = i; j < numberOfSequences; j++) {
53                      int score = 0;
54                      loopcount++;
55                      for (int k = 0; k < end; k++) {
56                          try {
57                              score += pwmatrix.getPairwiseScore(
                                    sequenceString[i].charAt(k),
                                    sequenceString[j].charAt(k));
58                          } catch (Exception ex) {
59                              System.err.println("err creating
                                    BLOSUM62 tree");
60                              ex.printStackTrace();
```

```
61                          }
62                      }
63
64                      distance[i][j] = (float) score;
65
66                      if (score > maxscore) {
67                          maxscore = score;
68                      }
69                  }
70              }
71
72              for (int i = 0; i < (numberOfSequences - 1); i++) {
73                  for (int j = i; j < numberOfSequences; j++) {
74                      distance[i][j] = (float) maxscore - distance[i
                            ][j];
75                      distance[j][i] = distance[i][j];
76                  }
77              }
78
79          }
80          //updateProgress("Determining Distances", 100);
81
82          return distance;
83      }
```

Listing 3.31: Distanzmatrix

Die „Start“ Methode setzt die Werte für die Distanzmatrixberechnung und wandelt das Ergebnis der Berechnung in die entsprechende Matrix um.

```
1       public DistanceMatrix start(MultipleSequenceAlignment<
            ProteinSequence, AminoAcidCompound>
            _multipleSequenceAlignment, TreeType _treeType,
            TreeConstructionAlgorithm _treeConstructionAlgorithm,
            NJTreeProgressListener _treeProgessListener) throws
            Exception {
2           this.treeType = _treeType;
3           this.treeConstructionAlgorithm = _treeConstructionAlgorithm
                ;
4           this.treeProgessListener = _treeProgessListener;
5           this.multipleSequenceAlignment = _multipleSequenceAlignment
                ;
6
7           matrix = null;
8           if (matrix == null) {
9               double[][] distances = calculateDistanceMatrix(
                    multipleSequenceAlignment, treeConstructionAlgorithm
```

```
                );
10              System.out.println("Dis:\n");
11
12              matrix = new BasicSymmetricalDistanceMatrix(
                    multipleSequenceAlignment.getSize());
13
14              for (int i = 0; i < matrix.getSize(); i++) {
15                  if (multipleSequenceAlignment.getAlignedSequence(i
                        + 1).getAccession() != null) {
16                      String id = multipleSequenceAlignment.
                            getAlignedSequence(i + 1).getAccession().
                            getID();
17                      System.out.println(id);
18                      matrix.setIdentifier(i, id);
19                  } else {
20                      matrix.setIdentifier(i, "" + (i + 1));
21                  }
22              }
23              System.out.println("Matrix:␣␣␣␣" + matrix.getSize());
24              for (int col = 0; col < matrix.getSize(); col++) {
25                  for (int row = 0; row < matrix.getSize(); row++) {
26                      matrix.setValue(col, row, distances[col][row]);
27
28                  }
29              }
30              copyDistanceMatrix = CheckTreeAccuracy.copyMatrix(
                    matrix);
31          }
32          System.out.println(matrix);
33          final List<Phylogeny> ps = new ArrayList<Phylogeny>();
34          final NeighborJoining nj = NeighborJoining.createInstance()
                ;
35          nj.setVerbose(verbose);
36
37          ps.add(nj.execute(matrix));
38          p = ps.get(0);
39          System.out.println(matrix);
40          return matrix;
41      }
42  }
```

Listing 3.32: Distanzmatrix Start Methode

3.5 BioJava Protein Struktur

3.5.1 PDB Datei verarbeiten

Das Auslesen der PDB Datei ist über den PDBFileReader möglich. Bei diesem Reader müssen zwingend die FileParsingParameter gesetzt werden, denn sonst können nicht alle PDB Dateien gelesen werden. Bei den FileParsingParametern muss der AlignSeqRes auf true gesetzt werden. Nach dem erfolgreichen Einlesen der Datei wird ein Structure Element zurück gegeben.

```
1  public Structure readPDB(File pdbFile) {
2          try {
3              PDBFileReader pdbfileReader = new PDBFileReader();
4              FileParsingParameters fpp = new FileParsingParameters()
                   ;
5              fpp.setAlignSeqRes(true);
6              pdbfileReader.setFileParsingParameters(fpp);
7              Structure structure = pdbfileReader.getStructure(
                   pdbFile);
8              return structure;
9          } catch (IOException ex) {
10             Logger.getLogger(ProteinStructure.class.getName()).log(
                   Level.SEVERE, null, ex);
11             return null;
12         }
13     }
```

Listing 3.33: PDB Datei verarbeiten

3.5.2 mmcif Datei verarbeiten

Das Verarbeiten der mmcif Datei erfordert nicht das Setzen von Parametern. Auch hier gibt es als Rückgabe ein Structure Element.

```
1  public Structure readMMCIF(File mmcifFile) {
2          try {
3              StructureIOFile mmcifReader = new MMCIFFileReader();
4              Structure structure = mmcifReader.getStructure(
                   mmcifFile);
5              return structure;
6          } catch (IOException ex) {
7              Logger.getLogger(ProteinStructure.class.getName()).log(
                   Level.SEVERE, null, ex);
8              return null;
9          }
10     }
```

Listing 3.34: mmcif Datei verarbeiten

3.5.3 Zugriff auf Atome in einer Struktur

Der Zugriff auf Atome kann auf zwei einfache Arten vorgenommen werden. Variante eins ist dabei die Ausnutzung von vordefinierten Atomen. Im Listing 3.35 Nr. 1 ist das Lesen der vordefinierten C-α-Atome dargestellt. In der zweiten Variante im Listing ist das Laden von C-α-Atomen in Verbindung mit Stickstoff(„N“) dargestellt. Die zweite Variante erlaubt es dem Benutzer spezielle Atome, welche für ihn wichtig sind auszulesen.

```
1  Atom[] caAtom = StructureTools.getAtomCAArray(struct);
2  Atom[] atoms = StructureTools.getAtomArray(struct, new String[]{"
       CA"});
```

Listing 3.35: Zugriff auf Atome in der Struktur

3.5.4 Berechnungen mit Atomen

Die Berechnungen können mittels der Klasse „Calc“ durchgeführt werden. Im nachfolgenden Listing ist die Berechnung des Torsionswinkel ϕ exemplarisch dargestellt.

```
1  if (!Calc.isConnected((AminoAcid)groups.get(0),(AminoAcid)groups
       .get(1))){
2          throw new StructureException("can not calc Phi - 
               AminoAcids are not connected!");
3    }
4        Atom a_C  = ((AminoAcid)groups.get(0)).getAtom("C");
5        Atom b_N  = ((AminoAcid)groups.get(1)).getAtom("N");
6        Atom b_CA = ((AminoAcid)groups.get(1)).getAtom("CA");
7        Atom b_C  = ((AminoAcid)groups.get(1)).getAtom("C");
8        double phi = Calc.torsionAngle(a_C,b_N,b_CA,b_C);
9        System.out.println("phi  "+ phi);
```

Listing 3.36: Berechnungen mit Atomen

3.5.5 Arbeiten mit Gruppen

Das Auslesen der Gruppen kann über die Methode „getAtomGroups(AtomGruppenname)“ erfolgen, als Atomgruppennamen haben sich „amino“, „nucleotide“ und „hetatm“ als die Wichtigsten herausgestellt. Ist man sich nicht sicher, welche Atomgruppen enthalten sind, sollte man die Möglichkeit zum ermitteln aller Gruppen über die Methode „getAtomGroups()“ bevorzugen.

```
1  List<Group> groups = chain.getAtomGroups("amino");
2  chain.getAtomGroups("nucleotide");
3  chain.getAtomGroups("hetatm");
4  List<Group> allgroups = chain.getAtomGroups();
```

Listing 3.37: Arbeiten mit Gruppen in der Struktur

3.5.6 Zugriff auf die Headerinformationen in der PDB Datei

Der Zugriff auf die Headerinformationen erfolgt über zwei Schritte. Schritt eins ist das Lesen der Headerwerte. Der zweite Schritt ist der des Auslesens der Compoundinformationen.

```
1  //Header
2  Map<String, Object> m = struct.getHeader();
3  Set<String> keys = m.keySet();
4  for (String key : keys) {
5      System.out.println(key + ": " + m.get(key));
6  }
7  System.out.println("available compounds:");
8  List<Compound> compounds = struct.getCompounds();
9  for (Compound compound : compounds) {
10     System.out.println(compound);
11 }
```

Listing 3.38: Headerinformationen

3.5.7 Umgang mit SEQRES und ATOM Gruppen

Der Zugriff erfolgt über die jeweilige Kette in der Sturktur. Über getSeqResLength bzw. getAtomLength kann die Länge beider Gruppen ermittelt werden.

```
1  System.out.println("The SEQRES and ATOM information is available
       via the chains:");
2  int modelnr = 0; // also is 0 if structure is an XRAY structure.
3  List<Chain> chains = struct.getChains(modelnr);
4  for (Chain cha : chains) {
5  List<Group> agr = cha.getAtomGroups("amino");
6  List<Group> hgr = cha.getAtomGroups("hetatm");
7  List<Group> ngr = cha.getAtomGroups("nucleotide");
8  System.out.print("chain: >" + cha.getChainID() + "<");
9  System.out.print(" length SEQRES: " + cha.getSeqResLength());
10 System.out.print(" length ATOM: " + cha.getAtomLength());
11 System.out.print(" aminos: " + agr.size());
12 System.out.print(" hetatms: " + hgr.size());
13 System.out.println(" nucleotides: " + ngr.size());
14 }
```

Listing 3.39: Umgang mit SEQRES und ATOM Gruppen

3.5.8 Residue mutieren

Über die Klasse Mutator kann eine vorhandene Structure mutiert werden. Bei der Mutation ist die zu mutierende Structure, die KettenID, die Residuenummer und der neue Typ erforderlich.

```
1 String chainId = "␣";
2 String pdbResnum = "3";
3 String newType = "ARG";
4 Mutator mut = new Mutator();
5 Structure newstruc = mut.mutate(struct, chainId, pdbResnum, newType
      );
```

Listing 3.40: Residue mutieren

3.5.9 SCOP Klassifikation laden

Um mit der SCOP Klassifikation zuarbeiten muss zwingend ein temporäres Verzeichnis angelegt werden, sollte dies nicht angelegt werden, wird es in den Standard tmp Ordner des Betriebssystems geladen, was dazu führen kann, das nach jeden Neustart des Betriebssystems die SCOP Klassifikation neu aus dem Netz geladen werden muss. Es werden dabei nur die notwendigen Daten für die entsprechende PDB ID geladen.

```
1  String cacheLocation = "./tmp/";
2  String pdbId = "1PQS";
3  ScopInstallation scop = new ScopInstallation(cacheLocation);
4  List<ScopDomain> domains = scop.getDomainsForPDB(pdbId);
5  System.out.println(domains);
6  ScopNode node = scop.getScopNode(domains.get(0).getSunid());
7  while (node != null) {
8          System.out.println("This␣node:␣sunid:" + node.getSunid());
9          System.out.println(scop.getScopDescriptionBySunid(node.
               getSunid()));
10         node = scop.getScopNode(node.getParentSunid());
11 }
```

Listing 3.41: SCOP Klassifikation laden

3.6 BioJava Protein Modifikation

3.6.1 Identifikation von Proteinmodifikationen in einer 3D Struktur

Die einfachste Variante zur Identifikation von Proteinmodifikationen ist sich alle Modifikation auf einmal ermitteln zulassen, da die BioJava Bibliothek über 250 Modifikationen(siehe Anhang A) von Haus aus unterstützt.

```
1 public Set<ModifiedCompound> identifyAll(Structure struct) {
2         ProteinModificationIdentifier parser = new
              ProteinModificationIdentifier();
3         parser.identify(struct, ProteinModificationRegistry.
              allModifications());
4         Set<ModifiedCompound> mcs = parser.
              getIdentifiedModifiedCompound();
5         return mcs;
6     }
```

Listing 3.42: Proteinmodifikationen laden

3.7 BioJava Webservices

3.7.1 NCBI QBlast Services

Die Abfrage einer Blastanfrage ist normalerweise über den NCBIQBlastService möglich, da jedoch zum Zeitpunkt der Verfassung der Arbeit an der NCBI in vielen Bereichen, darunter auch die Webschnittstellen gearbeitet wurde, musste der NCBIQBlastService umgeschrieben werden, die neue Version ist im Listing 3.53 dargestellt. Das Setzen der Art des Blast und die Datenbank ist abhängig vom Blast. Als Defaultdatenbankwert sollte „nr“ genommen werden. Eine genaue Übersicht der einzelnen Datenbanken ist im Anhang B und C enthalten. Da es zu grossen Problemen beim Test des Blastservices kam, hat der Benutzer in dieser Version nicht die Möglichkeit weitere Einstellung am Blast vorzunehmen.

```
1  public String blast(String kind, String sequence) {
2          try {
3              NCBIQBlastServiceNew rbw;
4              NCBIQBlastAlignmentProperties rqb;
5              NCBIQBlastOutputProperties rof;
6              InputStream is;
7              ArrayList<String> rid = new ArrayList<String>();
8              String request = "";
9              rbw = new NCBIQBlastServiceNew();
10             rqb = new NCBIQBlastAlignmentProperties();
11             rqb.setBlastProgram(kind);
12             rqb.setBlastDatabase("nr");
13             rqb.setAdvancedOptions("not_used");
14             request = rbw.sendAlignmentRequest(sequence, rqb);
15             rid.add(request);
16                   StringBuffer sb = new StringBuffer();
17             for (String aRid : rid) {
18                 System.out.println("trying␣to␣get␣BLAST␣results␣for
                       ␣RID␣"+ aRid);
19                 boolean wasBlasted = false;
20                 while (!wasBlasted) {
21                     wasBlasted = rbw.isReady(aRid, System.
                           currentTimeMillis());
22                 }
23                 rof = new NCBIQBlastOutputProperties();
24                 rof.setOutputFormat(NCBIQBlastOutputFormat.TEXT);
25                 rof.setAlignmentOutputFormat(NCBIQBlastOutputFormat
                       .PAIRWISE);
26                 rof.setDescriptionNumber(10);
27                 rof.setAlignmentNumber(10);
28                 is = rbw.getAlignmentResults(request, rof);
29                 BufferedReader br = new BufferedReader(new
                       InputStreamReader(is));
```

```
30                  String line = null;
31                  while ((line = br.readLine()) != null) {
32                      sb.append(line + "\n");
33                  }
34              }
35              return sb.toString();
36          } catch (Exception ex) {
37              Logger.getLogger(Webservices.class.getName()).log(Level
                    .SEVERE, null, ex);
38              return null;
39          }
40      }
```

Listing 3.43: NCBI QBlast Services

Die vordefinierten Eingaben sollten nur im Notfall geändert werden. Es konnte nicht ermittelt werden, ob die Mailadresse wirklich gebraucht wird. Alle anderen Einstellungen werden im Zuge der Bearbeitung der Anfrage gesetzt.

```
1  public class NCBIQBlastServiceNew implements
       RemotePairwiseAlignmentService {
2
3      private static String baseurl = "http://www.ncbi.nlm.nih.gov/
           blast/Blast.cgi";
4      private URL aUrl;
5      private URLConnection uConn;
6      private OutputStreamWriter fromQBlast;
7      private BufferedReader rd;
8      private String email = "anonymous@biojava.org";
9      private String tool = "biojava3";
10     private String seq = null;
11     private String tmp = null;
12     private String prog = null;
13     private String db = null;
14 //      private String advanced = null;
15     private String rid;
16     private long step;
17     private long start;
18     private HashMap<String, Long> holder;
```

Listing 3.44: NCBI QBlast Services Neu

Der Defaultkonstruktor für den NCBIBlastservice erzeugt ein neues Objekt mit den Defaulteinstellungen.

```
1       public NCBIQBlastServiceNew() throws Exception {
2           try {
3               this.aUrl = new URL(baseurl);
4               this.uConn = setQBlastServiceProperties(aUrl.
                    openConnection());
5               this.holder = new HashMap<String, Long>();
6           } catch (IOException e) {
7               throw new Exception(
8                       "Impossible␣to␣connect␣to␣QBlast␣service␣at␣
                            this␣time.␣Check␣your␣network␣connection.\n"
                            );
9           }
10      }
```

Listing 3.45: NCBI QBlast Services Konstruktur Neu

Diese Methode macht die gesamte Arbeit, in dem sie die Anfrage zusammenbaut und an den Blastservice sendet. Die Anpassung muss auch hier stattfinden, wenn noch weiter an der NCBI gebaut wird.

```
1           private String sendActualAlignementRequest(String str,
2               RemotePairwiseAlignmentProperties rpa) throws Exception
                    {
3
4           seq = "QUERY=" + str;
5           prog = "PROGRAM=" + rpa.getAlignmentOption("PROGRAM");
6           db = "DATABASE=" + rpa.getAlignmentOption("DATABASE");
7
8           if (prog == null || db == null || str == null || str.length
                () == 0) {
9               throw new Exception(
10                      "Impossible␣to␣execute␣QBlast␣request.␣One␣or␣
                            more␣of␣sequence|database|program␣has␣not␣
                            been␣set␣correctly.\n");
11          }
12
13          String cmd = "CMD=Put&SERVICE=plain&" + seq + "&" + prog +
                "&"
14                  + db + "&" + "FORMAT_TYPE=HTML" + "&TOOL=" +
                        getTool() + "&EMAIL=" + getEmail();
15
16
17          System.out.println(baseurl+"?"+cmd);
18          try {
19
```

```
20              uConn = setQBlastServiceProperties(aUrl.openConnection
                    ());
21
22              fromQBlast = new OutputStreamWriter(uConn.
                    getOutputStream());
23
24              fromQBlast.write(cmd);
25
26              fromQBlast.flush();
27
28              // Get the response
29              rd = new BufferedReader(new InputStreamReader(uConn.
                    getInputStream()));
30
31              String line = "";
32
33              while ((line = rd.readLine()) != null) {
34                  //System.out.println(line);
35                  if (line.contains("RID")) {
36
37                      String[] arr = line.split("=");
38                      rid = arr[1].trim();
39                  } else if (line.contains("RTOE")) {
40                      //System.out.println(line);
41                      String[] arr = line.split("=");
42                      if(arr[1].trim().equals("")){
43                          arr[1]=arr[1].trim()+"1";
44                      }
45                      step = Long.parseLong(arr[1].trim()) * 1000;
46                      start = System.currentTimeMillis() + step;
47                  }
48                  holder.put(rid, start);
49              }
50             // System.exit(0);
51          } catch (IOException e) {
52              throw new Exception(
53                      "Can't submit sequence to BLAST server at this
                            time.\n");
54          }
55
56          return rid;
57      }
```

Listing 3.46: NCBI QBlast Services Anfrage senden

Die nachfolgenden Methoden sind Wrapper, die die Blast Anfrage über das Put Kommando an die CGI-BIN-Schnittstelle mit den angegebenen Parametern und einen String für die Sequenz senden. Die Fertigstellung der Anfrage wird über die Erfassung der RTOE Variable ermittelt.

```
1       public String sendAlignmentRequest(String str,
2               RemotePairwiseAlignmentProperties rpa) throws Exception
                    {
3           return rid = sendActualAlignementRequest(str, rpa);
4       }
```

Listing 3.47: NCBI QBlast Services Anfrage senden Wrapper

```
1       public String sendAlignmentRequest(Sequence rs,
2               RemotePairwiseAlignmentProperties rpa) throws Exception
                    {
3
4           tmp = rs.getSequenceAsString();
5
6           return rid = sendActualAlignementRequest(tmp, rpa);
7       }
```

Listing 3.48: NCBI QBlast Services Anfrage senden Wrapper

```
1       public String sendAlignmentRequest(int gid,
2               RemotePairwiseAlignmentProperties rpa) throws Exception
                    {
3
4           tmp = Integer.toString(gid);
5           return rid = sendActualAlignementRequest(tmp, rpa);
6       }
```

Listing 3.49: NCBI QBlast Services Anfrage senden Wrapper

Diese Methode dient zum Abfragen, ob die Anfrage bereits durch den Blastservice der NCBI bearbeitet wurde. Der Nachteil dieser Variante ist, dass sie über Polling gelöst wurde. Die Abfrage der Werte „READY“ und „WAITING“ muss immer weiter überprüft werden, da sich ändern können.

```
1
2       public boolean isReady(String id, long present) throws
            Exception {
3           boolean isReady = false;
4           String check = "CMD=Get&RID=" + id;
5
6           if (holder.containsKey(id)) {
7
8               if (present < start) {
9                   isReady = false;
10              } else {
```

```
11                  try {
12                      uConn = setQBlastServiceProperties(aUrl.
                            openConnection());
13
14                      fromQBlast = new OutputStreamWriter(uConn.
                            getOutputStream());
15                      fromQBlast.write(check);
16                      fromQBlast.flush();
17
18                      rd = new BufferedReader(new InputStreamReader(
                            uConn.getInputStream()));
19
20                      String line = "";
21
22                      while ((line = rd.readLine()) != null) {
23                          // System.out.println(line);
24                           if(line.contains("READY") |line.contains("
                                WAITING") ){
25                               System.out.println(line);
26                           }
27                           if (line.contains("READY")) {
28                               isReady = true;
29                           } else if (line.contains("WAITING")) {
30                               start = present + step;
31                               holder.put(id, start);
32                           }
33                      }
34                    //   System.exit(0);
35                  } catch (IOException e) {
36                      e.printStackTrace();
37                  }
38              }
39          } else {
40              throw new Exception("Impossible␣to␣check␣for␣request␣ID
                    ␣named␣"
41                      + id + "␣because␣it␣does␣not␣exists!\n");
42          }
43          return isReady;
44      }
```

Listing 3.50: NCBI QBlast Services Anfrage fertig berechnet

Die Extraktion der Daten aus dem Ergebnis vom NCBI wird hier erstellt. Es wird dabei einfach nur der Inhalt der Webseite abgefragt und weiter geleitet.

```
1       public InputStream getAlignmentResults(String id,
```

```
2             RemotePairwiseAlignmentOutputProperties rb) throws
                  Exception {
3         if (holder.containsKey(id)) {
4             String srid = "CMD=Get&RID=" + id + "&"
5                     + rb.getOutputOption("FORMAT_TYPE") + "&"
6                     + rb.getOutputOption("ALIGNMENT_VIEW") + "&"
7                     + rb.getOutputOption("DESCRIPTIONS") + "&"
8                     + rb.getOutputOption("ALIGNMENTS")
9                     + "&TOOL=" + getTool() + "&EMAIL=" + getEmail()
                          ;
10
11            try {
12                uConn = setQBlastServiceProperties(aUrl.
                      openConnection());
13
14                fromQBlast = new OutputStreamWriter(uConn.
                      getOutputStream());
15                fromQBlast.write(srid);
16                fromQBlast.flush();
17
18                return uConn.getInputStream();
19
20            } catch (IOException ioe) {
21                throw new Exception(
22                        "It␣is␣not␣possible␣to␣fetch␣Blast␣report␣
                              from␣NCBI␣at␣this␣time.\n");
23            }
24        } else {
25            throw new Exception(
26                    "Impossible␣to␣get␣output␣for␣request␣ID␣named␣
                          " + id
27                    + "␣because␣it␣does␣not␣exists!\n");
28        }
29    }
```

Listing 3.51: NCBI QBlast Services Anfrageergebnis auslesen

Der Test ob der Blastservices verfügbar ist wird über die nachfolgende Methode realisiert. Sollte die Webseite nicht verfügbar sein, wird eine Fehlermeldung erzeugt.

```
1     public void printRemoteBlastInfo() throws Exception {
2         try {
3             OutputStreamWriter out = new OutputStreamWriter(uConn.
                  getOutputStream());
4
5             out.write("CMD=Info");
6             out.flush();
```

```
7
8               // Get the response
9               BufferedReader rd = new BufferedReader(new
                    InputStreamReader(uConn.getInputStream()));
10
11              String line = "";
12
13              while ((line = rd.readLine()) != null) {
14                  System.out.println(line);
15              }
16
17              out.close();
18              rd.close();
19          } catch (IOException e) {
20              throw new Exception(
21                      "Impossible to get info from QBlast service at
                            this time. Check your network connection.\n"
                            );
22          }
23      }
```

Listing 3.52: NCBI QBlast Services Verfügbarkeitstest

Das Setzen der Eigenschaften, muss eigentlich für jede Anfrage erneut passieren, da man inzwischen Referenzwerte hat, sollten die gesetzten Werte für sehr viele Sequenzen reichen.

```
1       private URLConnection setQBlastServiceProperties(URLConnection
            conn) {
2
3           URLConnection tmp = conn;
4
5           conn.setDoOutput(true);
6           conn.setUseCaches(false);
7
8           tmp.setRequestProperty("User-Agent", "Biojava/
                NCBIQBlastService");
9           tmp.setRequestProperty("Connection", "Keep-Alive");
10          tmp.setRequestProperty("Content-type",
11                  "application/x-www-form-urlencoded");
12          tmp.setRequestProperty("Content-length", "200");
13
14          return tmp;
15      }
16  }
```

Listing 3.53: NCBI QBlast Services Eigenschaften setzen

3.7.2 PDB

Das Anbinden der PDB ist sehr einfach, man benötigt nur den PdbWebServiceServiceLocator. Über diesen Locator muss ein PdbWebService angefordert werden. Über die Keywortsuche kann u.a. nach PDB IDs oder Namen gesucht werden. Als Rückgabe werden die gefundenen PDB IDs übergeben. Mit diesen IDs lassen sich die Ketten und somit die einzelnen Sequenzen auslesen.

```
1  public String scanpdb(String search) {
2          try {
3              PdbWebServiceServiceLocator locator = new
                   PdbWebServiceServiceLocator();
4              URL url = new URL("http://www.pdb.org/pdb/services/
                   pdbws");
5              PdbWebService p = locator.getpdbws(url);
6              String[] ergebnisse = p.keywordQuery(search, false,
                   false);
7              StringBuffer sb = new StringBuffer();
8              for (String string : ergebnisse) {
9                  String[] sequences_chains = p.getChains(string);
10                 sb.append(string + "\n");
11                 for (String string1 : sequences_chains) {
12                     sb.append(p.getSequenceForStructureAndChain(
                           string, string1) + "\n");
13                 }
14             }
15             return sb.toString();
16         } catch (RemoteException ex) {
17             Logger.getLogger(Webservices.class.getName()).log(Level
                   .SEVERE, null, ex);
18         } catch (ServiceException ex) {
19             Logger.getLogger(Webservices.class.getName()).log(Level
                   .SEVERE, null, ex);
20         } catch (MalformedURLException ex) {
21             Logger.getLogger(Webservices.class.getName()).log(Level
                   .SEVERE, null, ex);
22         }
23         return null;
24     }
```

Listing 3.54: PDB Webservice

4 Probleme mit BioJava

4.1 Allgemeine Probleme

Die Probleme der Bibliothek sind sehr zahlreich und erfordern oft die Nach- oder Neuprogrammierung von Teilabschnitten der entsprechenden Stellen. Die Lösung dieser Probleme, ist jedoch dank der LGPL 2.1 einfach, da der Quellcode öffentlich ist.
Ein viel grösseres Problem ist die fehlende Unterstützung durch Beispiele und die fehlenden Erklärungen in der Dokumentation. In dem Falle, wo Beispiele vorhanden waren, waren diese meistens für die Version 1.X. Da die Umstrukturierung zur Version 3.X u.a. Strukturänderungen enthielt, sind die viele der Beispiele nicht mehr gültig.

4.2 Probleme in der Phylogenie

In der Erarbeitung des Phylogeniepaketes, ist ein Fehler aufgetreten, welcher in der Bibliothek lag. Der Fehler ist bei der Erstellung der Distanzmatrix aufgetreten. Es ist von dem Algorithmus auf eine Element zugegriffen wurde, welches nicht in der Liste existierte. Er wurde durch die Neuimplementierung des Distanzmatrixalgorithmus umgangen.

4.3 Probleme im Webservice beim Blast

Beim Starten des Webservice zum Blast, ist es zu Fehlern gekommen, welche nur indirekt am Programm lagen, sondern an der NCBI, welche zum Zeitpunkt der Arbeit seine Struktur im Bezug auf Webservices geändert hatte. Diese Änderungen hatte zur Folge, dass nicht mehr alle Funktionen in der vorimplementierten Weise nutzbar waren. Diese Fehler wurden durch eine nachträgliche Anpassung korrigiert.
Ein weiteres Problem ist die Übergabe des Ergebnisses der Blastanfrage, welche nur die reine Ergebnisseite in HTML enthielt und somit erst noch weiter gefiltert werden muss.

4.4 Probleme bei der Transformation der Ergebnisse

Die Berechnung und Bearbeitung der Informationen funktioniert in den meisten Fällen ohne grössere Probleme, jedoch erfordert es noch sehr viel Aufwand die einzelnen Ergebnisse in das richtige Format für die Darstellung oder Weiterverarbeitung zu transformieren. In den Ergebnisse des Blastservices ist meistens nur ein Teil der Webseite herausgefiltert und dabei werden teilweise unwichtige Informationen die den Benutzer nicht interessieren mit angezeigt.
In einigen Fällen ist durch das Programm nicht zuerkennen woher einige der Kommandozeilenergebnisse kommen. Diese Ergebnisse sind sehr oft genauer als die Ergebnisse über die definierten Ausgabenformate von „BioJava“.

4.5 Probleme bei den Versionen

Ein sehr großes Problem ist die Umstellung von Version 1.X auf 3.X der freien Bibliothek. Diese Änderung hatte zur Folge, dass nicht alle Lösungen in die neue Version übernommen werden können. Sollten sie übernommen werden, dann ist dies nur über einen sehr großen Aufwand möglich. Dieser Aufwand ist fast der Neuentwicklung gleichzusetzen. Des weiteren müssen die Entwickler von Projekten umdenken, da einige Funktionen gestrichen wurden.

5 BioJava für den Einsatz in Forschung und Lehre Tool

Im Rahmen der Forschung und Lehre soll es dieses Tool ermöglichen die Arbeit einfacher zu gestalten und die Ausbildung der Studenten voran treiben. Das Tool setzt dabei unter anderem auf den Funktionsumfang der BioJava Bibliothek, SwingX, PDBWebsrvice oder XML-Technologie.
Das Tool soll für alle Benutzer ansprechend sein und leicht zu erlernen. Ein integrierte Hilfe soll den Benutzer Hilfestellungen im Umgang mit dem Tool da bieten.
Eine Möglichkeit für die Internationalisierung im Zuge der Entwicklung ist bei diesen Tool auch ein wichtiger Punkt. Dieser Punkt soll es ermöglichen, dass das Tool nicht nur an der Hochschule Mittweida, sondern auch weltweit, eingesetzt wird.

5.1 Konzeption

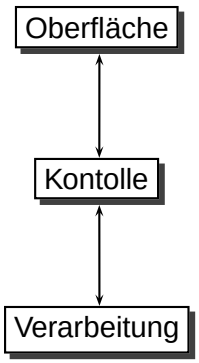

Abbildung 5.1: Konzeption des Tools

Die Reduzierung der Benutzung von unterschiedlichen Tools und Datenbanken soll durch das Tool erreicht werden, in der Form, dass das Tool dies selber macht und nicht mehr der Benutzer. Durch diese Hilfe ist gleichzeitig die Verringerung der Fehler mit eingeplant. Durch die Anbindung der PDB mit mehr als 76000 Strukturen[28] und der NCBI mit Zugriff auf viele Datenbanken hat der Benutzer Zugriff auf eine sehr grosse Datenmenge.
Die Entwicklung einer einfach zu bedienenden Oberfläche hatte eine genau so hohe Priorität, wie die Umsetzung der einzelnen Funktionen. Im Konzept wurde bereits eine an die Möglichkeit gedacht, dass es eine multi Sprachunterstützung geben kann. Um die Einbindung in andere Projekte zu erleichtern, sollte auch die Möglichkeit geschaffen werden, das Quellcode für den jeweiligen Bereich automatisch generiert wird. Für die Verwendung des Tools wird auch eine interne Hilfe vorhanden sein.

[28] Vgl. [18]

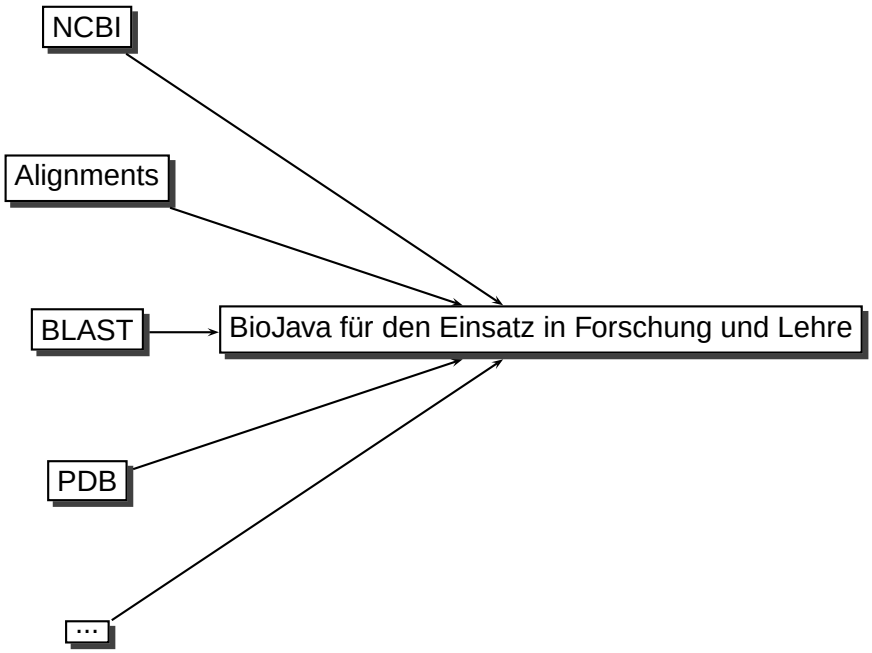

Abbildung 5.2: Konzeption II des Tools

5.2 Arbeiten mit Sequenzen

Das Arbeiten mit Sequenzen ist in der Bioinformatik unumgänglich. Aus diesen Grund, stellt das Tool drei Abschnitte für den Umgang mit Sequenzen bereit. Die Einteilung wurde dabei von der BioJava Bibliothek übernommen.

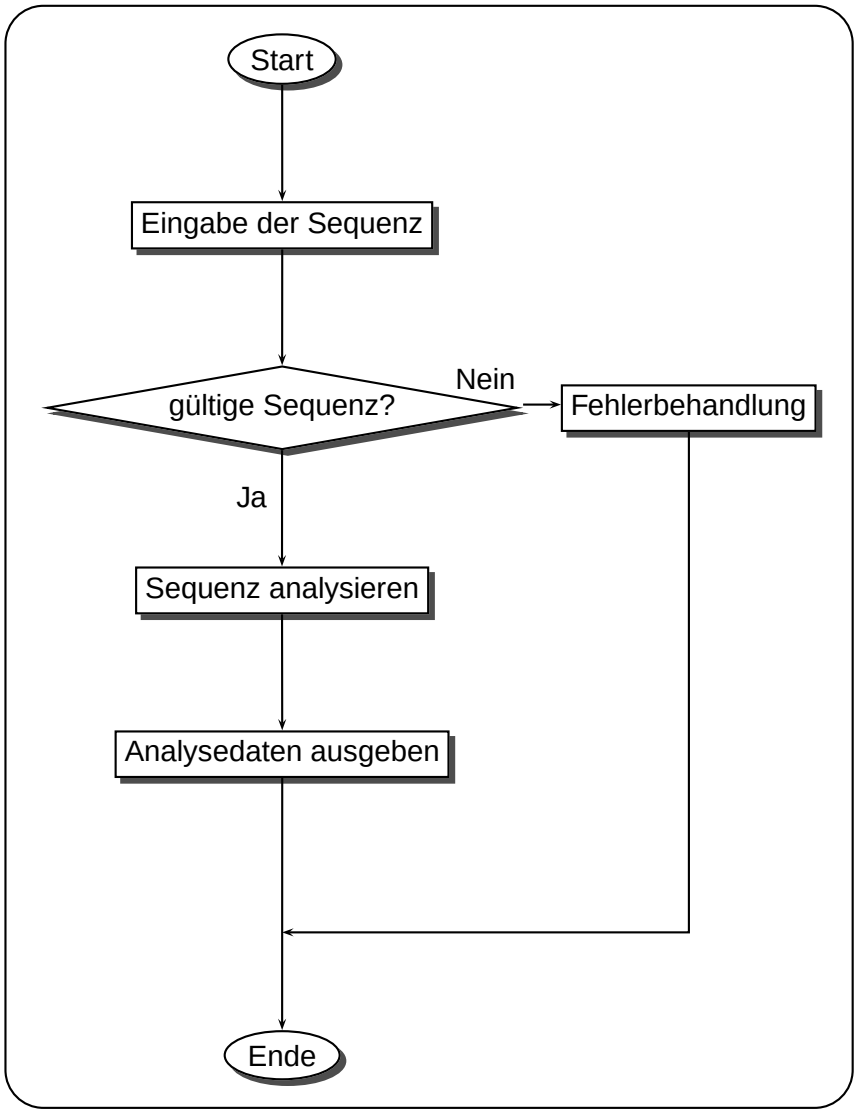

Abbildung 5.3: Allgemeiner Ablauf für das Arbeiten mit Sequenzen

5.2.1 Erstellung und Analyse einer DNA Sequenz

Die Erstellung und Analyse einer DNA Sequenz verlauft sehr einfach. Im Programm muss man nur den Punkt „Sequences“ wählen und erhält auf der linken Seite eine Auswahl der zur Verfügung stehenden Gebiete. Aus diesen Gebieten sucht man sich den Punkt „Analyse DNA“ aus und klick darauf. Im anschliessend erscheinenden Fenster muss der Benutzer nur noch seine DNA Sequenz eingeben und den Vorgang zur Analyse starten. Als Ergebnis bekommt der Benutzer eine kurze Informationsdarstellung über einige Eigenschaften.

5.2.2 Erstellung und Analyse einer Chromosom Sequenz

Zur Erzeugung sowie derer Verarbeitung ist die Chromosomsequenz erforderlich. Um diesen Typ von Sequenzen zu erstellen, muss wie bei der Erstellung der DNA Sequenz vorgegangen werden, nur mit dem Unterschied das der Punkt „Chromosome Sequence“ gewählt wird. Auch hier werden als Ergebnis einige der Eigenschaften der Chromosomsequenz ausgegeben.

5.2.3 Erstellung und Analyse einer Gene Sequenz

Äquivalent zu den beiden voraus gegangenen Sequenztypen muss eine Sequenz eingetragen werden, welche als Chromosomsequenz interpretiert wird. Des weiteren kommt der Start und Ende der Gensequenz hinzu, sowie der Strand. Die Auswahl dieser Art erfolgt über den Punkt „Gene Sequence“. Die Darstellung der Eigenschaften erfolgt analog zur DNA- oder Chromosomsequenz.

5.2.4 Erstellung und Analyse einer Intron, Exon oder Transcript Sequenz

Die Erstellung der Intron, Exon oder Transcript Sequenz erfolgt über den Punkt „Intron Sequence“, „Exon Sequence“ oder „Transcript Sequence“. Die Definitionen einer Gensequenz gelten auch hier und es müssen Start und Ende des Intron-, Exon- oder Transcriptbereich angegeben werden. Das Ergebnis ist an die Darstellung der anderen Sequenztypen angepasst.

5.2.5 DNA Translation

Die Translation der DNA in Proteine kann über den Punkt „DNA Translation“ erfolgen. Der Benutzer kann über vier verschiedene Methoden eine Translation durchführen. Die Varianten „Quick und Dirty“, „Translating in Multiple Frames“ und „Using a TranscriptionEngine“ benötigen keine weiteren Angaben. Die Variante „Translating in a Different Frame“ erfordert die Auswahl des Frames, in welche die Sequenz übersetzt werden soll.

5.2.6 Erstellung und Analyse einer RNA Sequenz

Die Erstellung und Analyse der RNA Sequenz erfolgt genau wie bei der DNA Sequenz, jedoch wird nicht zwischen den einzelnen Unterarten unterschieden. Die signifikanten Eigenschaften werden dargestellt.

5.2.7 Erstellung und Analyse einer Protein Sequenz

Die Verarbeitung der Protein Sequenz erfolgt über drei Wege. Diese sind: die Erstellung mittels der eigentlichen Sequenz, der Accession ID und der PDB ID. Nach der Eingabe erfolgt die Verarbeitung und die Ausgabe der Analyseergebnisse. Die Arbeit mit der Accession bzw. PDB ID erfordert die Kenntnis über den die jeweilige ID. Es werden auch nicht nur Teile der Sequenz geladen, sondern die Gesamte.

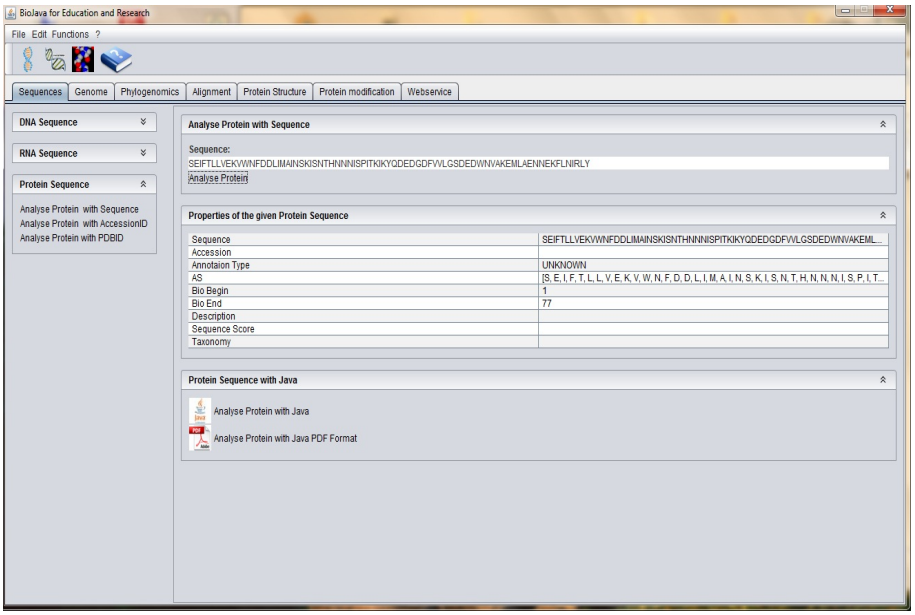

Abbildung 5.4: Darstellungsform eines Proteins im Tool

5.3 Arbeiten mit Dateien

Das Arbeiten mit Dateien ist sehr einfach gehalten wurden.

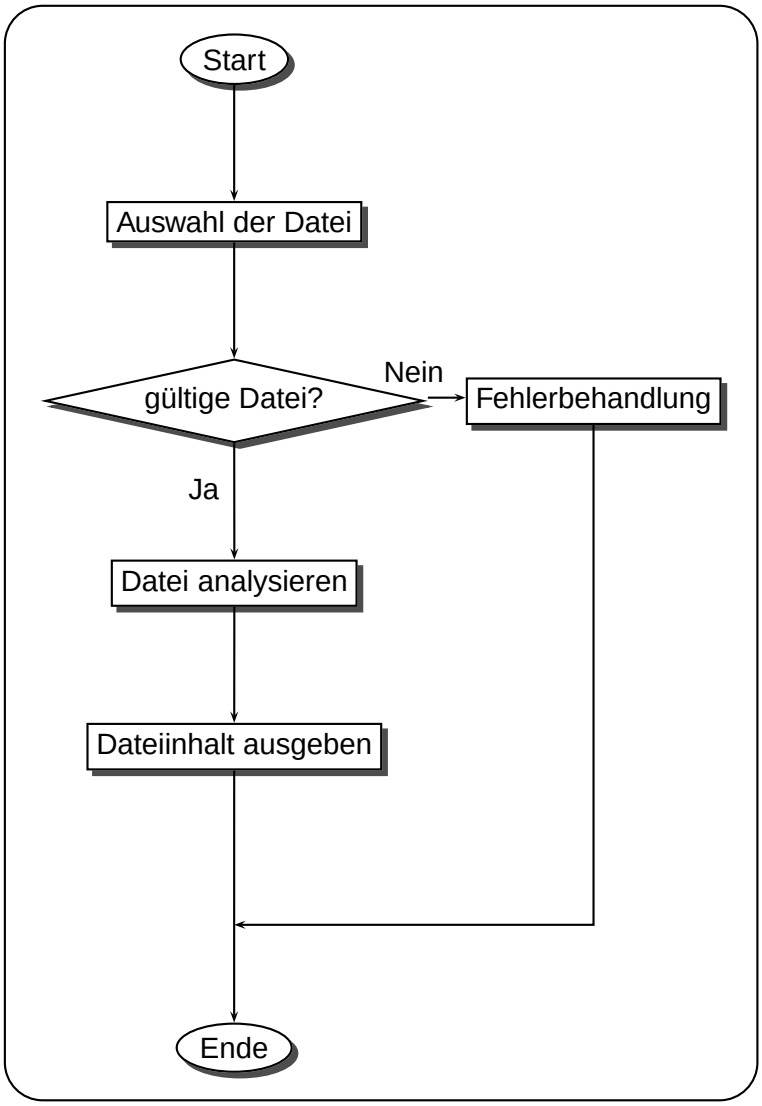

Abbildung 5.5: Allgemeiner Ablauf für das Arbeiten mit Dateien

5.3.1 Lesen von GTF, GFF2 und GFF3 Dateien

Das Lesen einer GTF, GFF2 oder GFF3 Datei erfordert die Auswahl einer entsprechenden Datei. Die Datei wird dann vom Tool eingelesen und bearbeitet. Als Ausgabe ist der Inhalt der Datei angezeigt.

5.3.2 Schreiben von GFF3 Dateien

Das Schreiben einer GFF3 Datei erfolgt exemplarisch für die Struktur 1pqs. Der Benutzer muss hier zu nur den Speicherort angeben.

5.3.3 Lesen und Schreiben von Fasta Dateien

Der Benutzer braucht für das Lesen nur eine Fasta Datei auswählen und das Programm erledigt den Rest, es wird die jeweilige Sequenz angegeben und die Annotation. Das Schreiben der Fasta Dateien wird in der aktuellen Version noch mit fest eingestellten Werten simuliert und erfordert nur die Angabe einer Zieldatei.

5.4 Zeichnen von phylogenetischen Bäumen

Die Erstellung dieser Bäume erfordert vom Benutzer die Eingabe von mehreren Sequenzen. Nach der Eingabe, wird der Baum automatisch berechnet und dargestellt. Als Baumkonstruktionsalgorithmus wird in dieser Version die Methode Neighbor Joining und als Distanzmatrix die PID Variante benutzt.

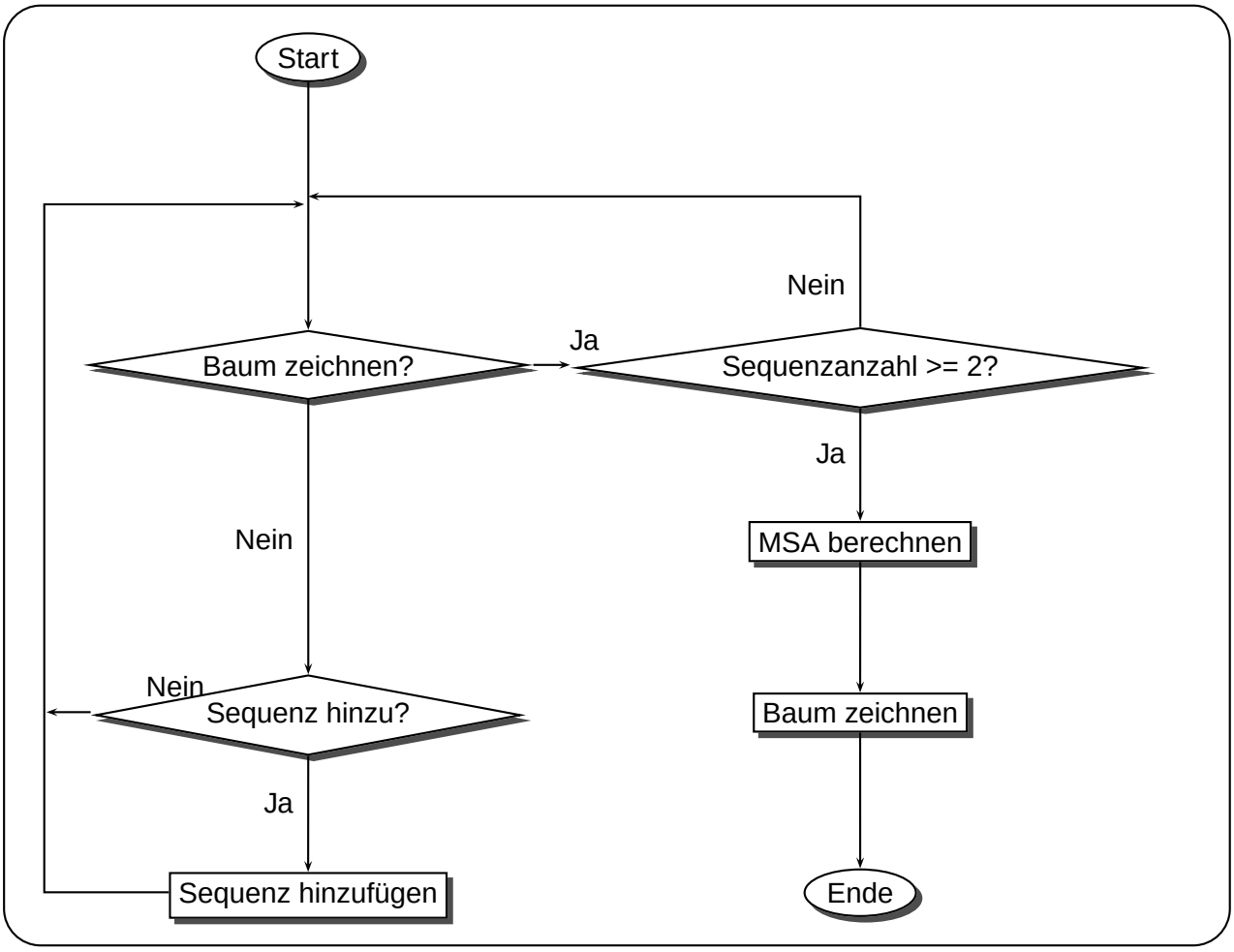

Abbildung 5.6: Allgemeiner Ablauf für das Erzeugen phylogentischer Bäume

Die Darstellung erfolgt im Tool über mxGraph, welches einen Baum erhält. Dieser Baum wird mit Hilfe des Newick String erstellt. Die Ermittlung findet rekursiv statt und ist nicht in der „BioJava“ Bibliothek enthalten.

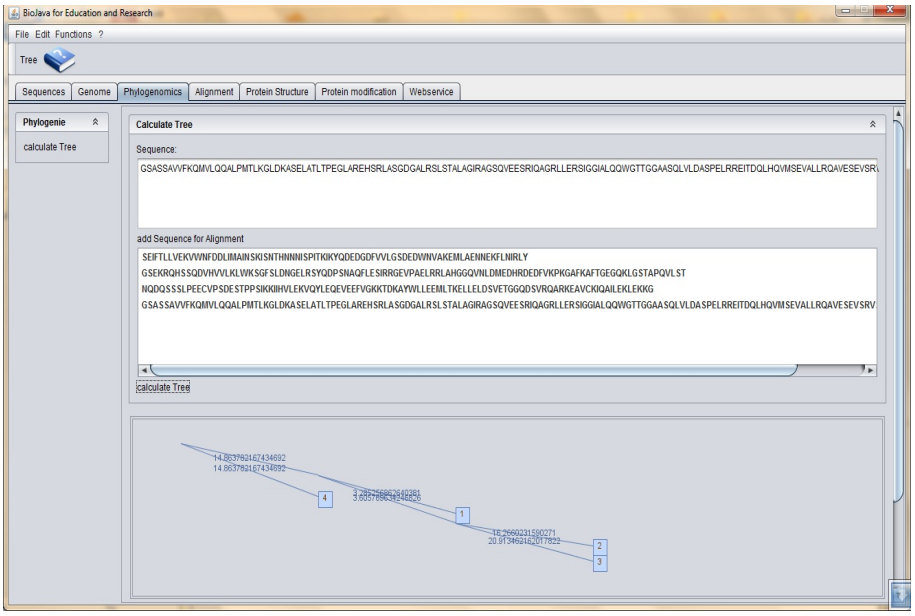

Abbildung 5.7: Darstellungsform eines Baumes im Tool

5.5 Ermittlung von Alignments

5.5.1 Ermittlung des Globalen bzw. Lokalen Alignments

Zur Ermittlung des globalen bzw. lokalen Alignments muss man nur zwei Sequenzen angeben. Es wird der Neddleman Wunsch Algorithmus für die globale Ausrichtung verwendet und Smith Waterman für die Lokale.

5.5.2 Ermittlung des MSA

Die Ermittlung eines MSA beginnt mit der Eingabe mehrerer Sequenzen. Diese Sequenzen werden miteinander aligniert. Als Ergebnis bekommt der Benutzer ein MSA.

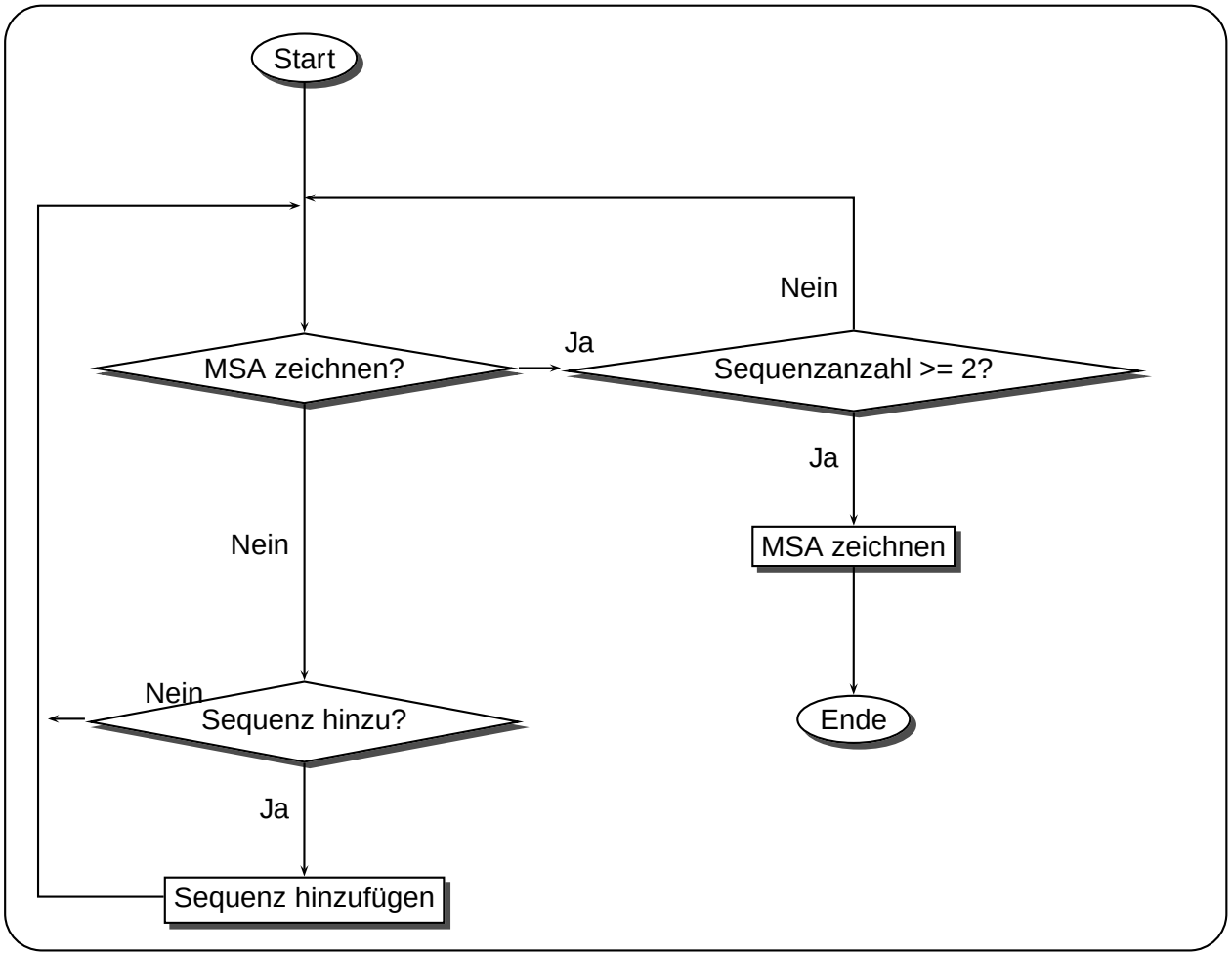

Abbildung 5.8: Allgemeiner Ablauf für das Arbeiten mit Multiple Sequence Alignment

5.6 Analyse von PDB und mmcif Dateien

Nach der Angabe einer PDB Datei erhält der Benutzer verschiedene Aussagen, wie den Inhalt. Mit diesen Daten kann der Benutzer danach selber verfahren. Die Analyse der mmcif Datei ist äquivalent zur PDB Datei.

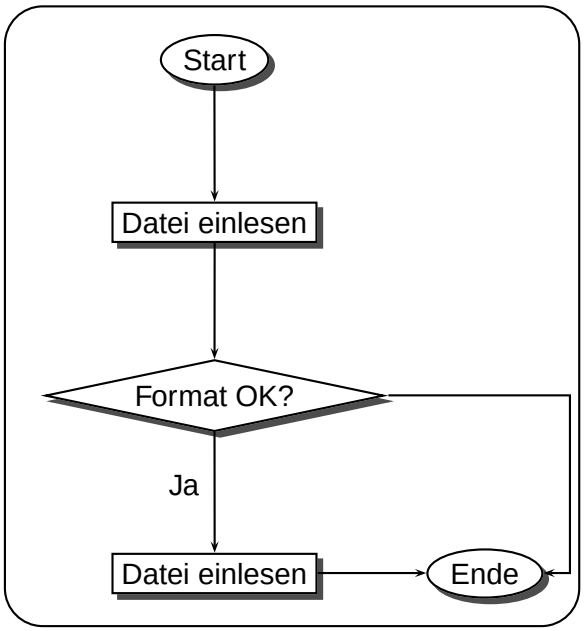

Abbildung 5.9: Allgemeiner Ablauf für das Arbeiten und Analysieren von pdb oder mmcif Dateien

5.7 Suchen nach Sequenzmodifikationen

Zur Suche der Modifikationen muss der Benutzer ein PDB oder mmcif File angeben, aus dieser Datei werden dann die Modifikationen bestimmt. Die Modifikation werden in A dargestellt.

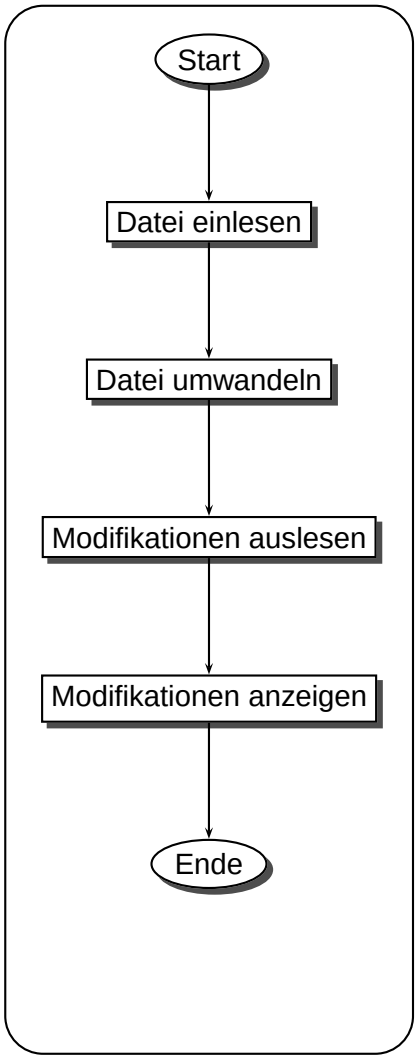

Abbildung 5.10: Allgemeiner Ablauf für das finden von Modifikatioen aus pdb oder mmcif Dateien

5.8 Arbeiten mit Blast und der PDB

5.8.1 Arbeiten mit Blast

Das Arbeiten mit Blast unterteilt sich wie bereits erläutert in die 5 Kategorien. Nach der Auswahl der Kategorie muss der Benutzer nur noch die Sequenz eingeben und auf „Blast“ klicken. Durch die prototypische Version des Tools und der Problem sind hier nur die Standardparameter gesetzt.

5.8.2 Arbeiten mit der PDB

Das Arbeiten mit der PDB ist über dieser Tool sehr einfach, man muss nur ein Suchbegriff, wie einen Namen oder eine PDB ID, eingeben. Das Ergebnis kann man sich dann über die Detailansicht genauer ansehen. In der aktuellen Version werden einige Informationen zur Struktur angezeigt, sowie eine grafische 3D Darstellung der Struktur.

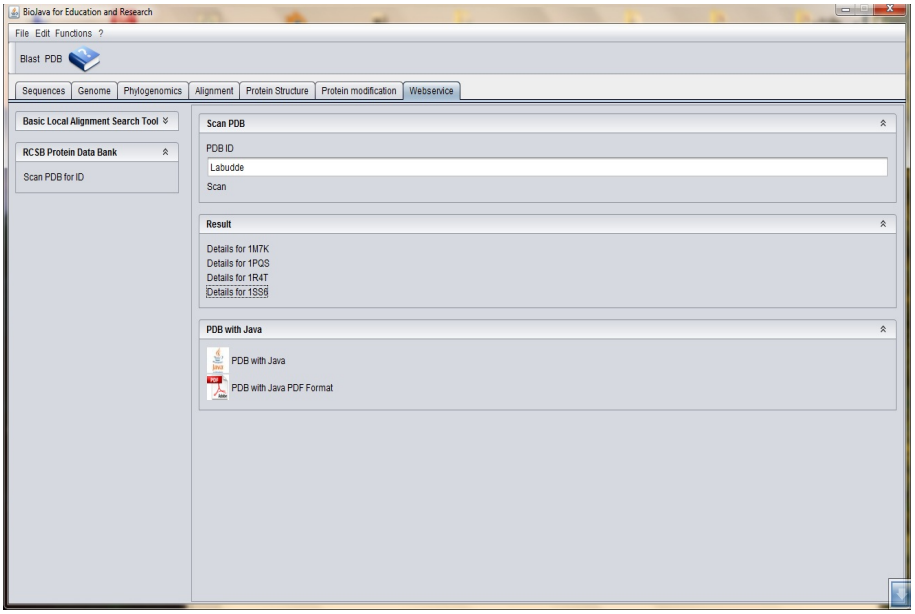

Abbildung 5.11: Listendarstellungsform einer PDB Anfrage

6 Fazit

Die freie Bibliothek „BioJava“ stellt eine erste Grundlage für den Umgang mit biologischen und chemischen Daten da. Sie kann in vielen Dingen überzeugen, jedoch muss man noch sehr viel Arbeit in die Einarbeitung und die Transformation der Daten investieren. Einige Funktionen sollten nicht benutzt werden, da diese zu Fehlern oder ungenauen Ergebnissen führen.
„BioJava“ sollte erst ab einer gewissen Programmgrösse benutzt werden, für den Einsatz in kleineren Programmen liegen der Aufwand und das Ergebnis zu weit auseinander um effizient damit zu arbeiten.
Die Grundanbindung des Blastservices sollte unter keinen Umständen benutzt werden, denn man muss diesen Service per Hand anpassen, was sehr viel Aufwand und Erfahrung erfordert.
Die Einarbeitung in die Gesamtheit der Bibliothek erfordert zum einen das fundierte Wissen eines Informatikers und zum anderen die Grundlagen der Biologie und Chemie. Wenn man den Umfang und die Probleme mit den Wissensstand der Ausbildung vergleicht, so ist diese Bibliothek vornehmlich von Informatikern ein zusetzen und wenn dies bedeutet, dass biologische und chemische Grundlagen erst erarbeitet werden müssen.

7 Ausblick

Diese Arbeit hat den ersten Schritt für die Zusammenführung vieler unterschiedlicher Anwendungen in einen Tool gezeigt. Dieser Schritt sollte der erste und auch nicht der Letzte sein.
Die Erweiterung um weitere Funktionen und Möglichkeiten sind eigentlich nur durch die Hardware der jeweiligen Systeme begrenzt, so könnte man versuchen fast alle biologischen Datenbanken in das System mit aufzunehmen.
Bei der Arbeit mit der NCBI sollte auf die fertige Version des Webservices gewartet werden und nicht die BioJava Bibliothek benutzt werden, da diese doch sehr anfällig für Fehler ist.
Das Tool selber sollte unter anderen eine Integration von möglichen Praktikumsaufgaben bekommen und die Hilfe sollte erweitert werden. Die Einbindung von zusätzlichen externen Tools sollte mit Vorsicht bedacht werden.
Die Möglichkeit über einen Workflow für die Bearbeitung eines Problems sollte in einer späteren Version des Tools nachgedacht werden und was die beste Umsetzung dafür darstellt.
Da sich die Version in der Zeit der Masterarbeit geändert hat, sollte die neue Version als Grundlage genommen werden. Die aktuelle Version hat noch weitere Möglichkeiten, welche nicht mit in die Arbeit eingeflossen sind.
Für die neuen Versionen der „BioJava“ Bibliothek sollte zwingend ein Interfacesystem entwickelt werden, so dass auch alter Versionen mit den neuen Varianten arbeiten können und dass nicht nach jedem Releasewechsel das Projekt neu geschrieben werden muss.

Anhang A: Modifikationen Liste

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0001		AA0406	MOD:00814	natural	attachment	glycoprotein
0002		AA0151	MOD:00160	natural	attachment	glycoprotein
0003	CSP	AA0034	MOD:00043	natural	modified residue	phosphoprotein
0004	NEP	AA0035	MOD:00044	natural	modified residue	phosphoprotein,
0005	HIP	AA0036	MOD:00045	natural	modified residue	phosphohistidine, phosphoprotein
0006	SEP	AA0037	MOD:00046	natural	modified residue	phosphoprotein
0007	TPO	AA0038	MOD:00047	natural	modified residue	phosphoprotein
0008	PTR	AA0039	MOD:00048	natural	modified residue	phosphoprotein
0009		AA0045	MOD:00054	natural	attachment	acetylated amino end
0010		AA0049	MOD:00058	natural	attachment	thioether bond, acetylated amino end
0011	ALY	AA0055	MOD:00064	natural	modified residue	acetyllysine
0012	TYS	AA0172	MOD:00181	natural	modified residue	sulfoprotein
0013	MLZ	AA0076	MOD:00085	natural	modified residue	methylated amino acid
0014	AGM	AA0272	MOD:00277	natural	modified residue	methylated amino acid
0015	M3L	AA0074	MOD:00083	natural	modified residue	methylated amino acid
0016	MME	AA0064	MOD:00073	natural	modified residue	thioether bond, methylated amino acid
0017	HYP	AA0030	MOD:00039	natural	modified residue	hydroxylation
0018		AA0025	MOD:00034	natural	crosslink2	redox-active center, disulfide bond
0019	SCH	AA0101	MOD:00110	hypothetical	modified residue	disulfide bond
0020		AA0139	MOD:00148	natural	crosslink3	3Fe-4S, iron-sulfur protein, metalloprotein
0021		AA0137	MOD:00146	natural	crosslink4	2Fe-2S, iron-sulfur protein, metalloprotein
0022	FME	AA0021	MOD:00030	natural	modified residue	thioether bond, blocked amino end, formylation, pretranslational modification
0023	CSE	AA0022	MOD:00031	natural	modified residue	pretranslational modification, selenium, selenocysteine
0024	AHB	AA0026	MOD:00035	natural	modified residue	hydroxylation
0025	BHD	AA0027	MOD:00036	natural	modified residue	hydroxylation

Tabelle A.1: Modifikationen Liste 1

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0026	LYZ	AA0028	MOD:00037	natural	modified residue	hydroxylation
0027	HY3	AA0029	MOD:00038	natural	modified residue	hydroxylation
0028	PCA	AA0031	MOD:00040	natural	modified residue	blocked amino end, pyroglutamic acid
0029	CGU	AA0032	MOD:00041	natural	modified residue	carboxyglutamic acid
0030	PHD	AA0033	MOD:00042	natural	modified residue	phosphoprotein
0031	DDE	AA0040	MOD:00049	natural	modified residue	diphthamide
0032	SCY	AA0056	MOD:00065	natural	modified residue	thioester bond
0033		AA0059	MOD:00068	natural	attachment	blocked amino end, lipoprotein, myristoylation
0034	MAA	AA0061	MOD:00070	natural	modified residue	methylated amino end
0035	MEA	AA0065	MOD:00074	natural	modified residue	methylated amino end
0037	2MR	AA0067	MOD:00076	natural	modified residue	methylated amino acid
0038	DA2	AA0068	MOD:00077	natural	modified residue	methylated amino acid
0039	MEN	AA0070	MOD:00079	natural	modified residue	methylated amino acid
0040	MEQ	AA0071	MOD:00080	natural	modified residue	methylated amino acid
0041	MHS	AA0073	MOD:00082	natural	modified residue	methylated amino acid
0042	MLY	AA0075	MOD:00084	natural	modified residue	methylated amino acid
0043	AAR	AA0082	MOD:00091	natural	modified residue	amidated carboxyl end
0044	CY3	AA0085	MOD:00094	natural	modified residue	amidated carboxyl end
0045	CLE	AA0091	MOD:00100	natural	modified residue	amidated carboxyl end
0046	NFA	AA0094	MOD:00103	natural	modified residue	amidated carboxyl end
0047	LPD	AA0095	MOD:00104	natural	modified residue	amidated carboxyl end
0049	VLM	AA0100	MOD:00109	natural	modified residue	amidated carboxyl end
0050	CMT	AA0105	MOD:00114	natural	modified residue	methylated carboxyl end

Tabelle A.2: Modifikationen Liste 1

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0051	P1L	AA0106	MOD:00115	natural	modified residue	lipoprotein, palmitoylation, thioester bond
0052		AA0106	MOD:00115	natural	attachment	lipoprotein, palmitoylation, thioester bond
0053	KCX	AA0114	MOD:00123	natural	modified residue	
0054	MCL	AA0115	MOD:00124	natural	modified residue	
0055		AA0117	MOD:00126	natural	attachment	biotin
0056		AA0124	MOD:00133	natural	crosslink2	isopeptide bond
0057		AA0124	MOD:01484	natural	crosslink2	isopeptide bond
0058		AA0125	MOD:00134	natural	crosslink2	blocked carboxyl end, isopeptide bond
0059		AA0126	MOD:00135	natural	crosslink2	blocked carboxyl end, isopeptide bond
0060		AA0216	MOD:00221	natural	crosslink2	blocked amino end, isopeptide bond
0061		AA0294	MOD:00299	natural	crosslink2	isopeptide bond
0062		AA0360	MOD:00365	natural	crosslink2	blocked amino end, isopeptide bond
0063		AA0440	MOD:00924	natural	crosslink2	blocked amino end, isopeptide bond
0064		AA0118	MOD:00127	natural	attachment	lipoamide, redox-active center
0065	LLP	AA0119	MOD:00128	natural	modified residue	phosphoprotein, pyridoxal phosphate
0066	LYR	AA0120	MOD:00129	natural	modified residue	chromoprotein, retinal
0067		AA0120	MOD:00129	natural	attachment	chromoprotein, retinal
0068		AA0121	MOD:00130	natural	attachment	
0069		AA0123	MOD:00132	natural	crosslink2	
0070		AA0131	MOD:00140	natural	attachment	chromoprotein, phycocyanobilin, thioether bond
0071		AA0132	MOD:00141	natural	attachment	chromoprotein, phycoerythrobilin, thioether bond
0074		AA0140	MOD:00149	natural	crosslink4	4Fe-4S, iron-sulfur protein, metalloprotein
0075		AA0141	MOD:00150	natural	crosslink2	iron-sulfur protein, metalloprotein, molybdenum

Tabelle A.3: Modifikationen Liste 3

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0076		AA0141	MOD:00150	natural	crosslink2	iron-sulfur protein, metalloprotein, molybdenum
0077		AA0142	MOD:00151	natural	attachment	metalloprotein, molybdenum, molybdopterin, phosphoprotein
0078		AA0143	MOD:00152	natural	attachment	phosphoprotein, FAD, flavoprotein, thioether bond
0079		AA0144	MOD:00153	natural	attachment	phosphoprotein, FAD, flavoprotein
0080		AA0145	MOD:00154	natural	attachment	phosphoprotein, FAD, flavoprotein
0081	TPQ	AA0147	MOD:00156	natural	modified residue	quinoprotein, topaquinone
0082	TRQ	AA0148	MOD:00157	natural	modified residue	quinoprotein
0083		AA0149	MOD:00158	natural	crosslink2	quinoprotein
0084		AA0150	MOD:00159	natural	attachment	phosphopantetheine, phosphoprotein
0085	GTH	AA0155	MOD:00164	natural	modified residue	glycoprotein
0088	BTR	AA0179	MOD:00188	natural	modified residue	bromine
0089	DHA	AA0181	MOD:01168	natural	modified residue	
0090	CRQ	AA0183	MOD:00191	natural	modified residue	
0091	CRQ	AA0189	MOD:00197	natural	crosslink1	chromoprotein
0092	CRO	AA0183	MOD:00191	natural	crosslink1	
0093	CRO	AA0184	MOD:00192	natural	crosslink1	chromoprotein
0094	FGL	AA0185	MOD:01169	natural	modified residue	
0095	DAL	AA0191	MOD:00862	natural	modified residue	D-amino acid
0098	DIL	AA0192	MOD:00199	natural	modified residue	D-amino acid
0099	DSG	AA0196	MOD:00203	natural	modified residue	D-amino acid
0100	DTR	AA0198	MOD:00205	natural	modified residue	D-amino acid

Tabelle A.4: Modifikationen Liste 4

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0101	DTH	AA0199	MOD:00863	natural	modified residue	D-amino acid
0102	DVA	AA0200	MOD:00705	natural	modified residue	D-amino acid
0103	CSO	AA0205	MOD:00210	natural	modified residue	
0104	CSX	AA0205	MOD:00210	natural	modified residue	
0105		AA0206	MOD:00211	natural	crosslink2	blocked carboxyl end, thioester bond
0106		AA0207	MOD:00212	natural	attachment	chromoprotein, hydroxylation, photoreceptor, thioester bond
0107	ARO	AA0215	MOD:00220	natural	modified residue	hydroxylation
0108		AA0220	MOD:00225	natural	attachment	phosphoprotein, flavoprotein, FMN, thioether bond
0109		AA0221	MOD:00226	natural	attachment	phosphoprotein, FAD, flavoprotein
0110	GPL	AA0228	MOD:00233	natural	modified residue	phosphoprotein
0111	SNC	AA0230	MOD:00235	natural	modified residue	
0112		AA0233	MOD:00238	natural	crosslink2	quinoprotein
0113	SMC	AA0234	MOD:00239	natural	modified residue	methylated amino acid, thioether bond
0114		AA0250	MOD:00255	natural	crosslink2	
0115	OMT	AA0251	MOD:00256	natural	modified residue	
0116		AA0252	MOD:00257	natural	attachment	thioether bond
0117		AA0258	MOD:00263	natural	attachment	chromoprotein, phycoviolobilin, thioether bond
0118		AA0260	MOD:00265	natural	crosslink2	chromoprotein, phycourobilin, thioether bond
0119	CSD	AA0262	MOD:00267	natural	modified residue	
0120	CSW	AA0262	MOD:00267	natural	modified residue	
0121		AA0264	MOD:00269	natural	attachment	phosphoprotein
0122	GL3	AA0265	MOD:00270	natural	modified residue	
0123		AA0266	MOD:00271	natural	crosslink3	chromoprotein, heme, iron, metalloprotein, thioether bond
0124		AA0271	MOD:00276	natural	crosslink3	chromoprotein, heme, iron, metalloprotein, thioether bond
0125	CSS	AA0269	MOD:00274	natural	modified residue	

Tabelle A.5: Modifikationen Liste 5

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0126		AA0270	MOD:00275	natural	crosslink2	
0127	MGN	AA0273	MOD:00278	natural	modified residue	methylated amino acid
0128	CSZ	AA0277	MOD:00282	hypothetical	modified residue	selenium
0129		AA0280	MOD:00285	natural	crosslink3	chromoprotein, heme, hydroxylation, iron, metalloprotein
0130		AA0283	MOD:00288	natural	crosslink2	quinoprotein
0131		AA0284	MOD:00289	natural	crosslink4	4Fe-4S, iron-sulfur protein, metalloprotein
0132		AA0285	MOD:00290	natural	crosslink4	4Fe-4S, iron-sulfur protein, metalloprotein
0133		AA0286	MOD:00291	natural	crosslink4	4Fe-4S, iron-sulfur protein, metalloprotein
0134		AA0288	MOD:00293	hypothetical	crosslink4	4Fe-4S, iron-sulfur protein, metalloprotein
0135		AA0289	MOD:00294	hypothetical	crosslink4	4Fe-4S, iron-sulfur protein, metalloprotein
0136		AA0298	MOD:00303	natural	crosslink7	copper, metalloprotein
0137		AA0300	MOD:00305	natural	crosslink7	iron-sulfur protein, metalloprotein
0138	SNN	AA0302	MOD:00307	natural	modified residue	blocked carboxyl end, protein splicing
0139		AA0310	MOD:00315	natural	crosslink6	iron-sulfur protein, metalloprotein, Ni-4Fe-5S, nickel
0140	DMH	AA0311	MOD:00316	hypothetical	modified residue	methylated amino acid,
0141		AA0313	MOD:00318	natural	crosslink2	quinoprotein, thioether bond
0142		AA0314	MOD:00319	natural	crosslink2	lanthionine, thioether bond
0143		AA0315	MOD:00320	natural	crosslink2	thioether bond
0144	HIC	AA0317	MOD:00322	natural	modified residue	methylated amino acid
0146	HTR	AA0322	MOD:00327	hypothetical	modified residue	hydroxylation
0147		AA0326	MOD:00331	hypothetical	crosslink4	3Fe-4S, iron-sulfur protein, metalloprotein
0148		AA0329	MOD:00334	natural	attachment	chromoprotein, heme, iron, metalloprotein
0149		AA0330	MOD:00335	natural	crosslink2	
0150		AA0331	MOD:00336	hypothetical	crosslink4	2Fe-2S, iron-sulfur protein, metalloprotein

Tabelle A.6: Modifikationen Liste 6

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0151		AA0334	MOD:00339	natural	attachment	iron-sulfur protein, metalloprotein
0152	IML	AA0336	MOD:00341	hypothetical	modified residue	methylated amino end
0153	MLE	AA0337	MOD:00342	hypothetical	modified residue	methylated amino end
0154		AA0339	MOD:00344	natural	attachment	blocked amino end, lipoprotein, palmitoylation
0155		AA0340	MOD:00345	natural	crosslink2	thioether bond
0156		AA0341	MOD:00346	natural	crosslink2	D-amino acid, thioether bond
0157		AA0342	MOD:00347	natural	crosslink2	D-amino acid, thioether bond
0159		AA0348	MOD:00353	natural	crosslink3	
0160		AA0351	MOD:00356	natural	attachment	phosphoprotein, flavoprotein, FMN, thioether bond
0161		AA0355	MOD:00360	natural	attachment	copper, metalloprotein, molybdenum, molybdopterin, phosphoprotein
0162		AA0356	MOD:00361	natural	crosslink3	4Fe-4S, iron-sulfur protein, metalloprotein, S-adenosyl-L-methionine
0163		AA0357	MOD:00362	natural	crosslink4	2Fe-2S, iron-sulfur protein, metalloprotein
0164	OSE	AA0361	MOD:00366	natural	modified residue	sulfoprotein
0165	CXM	AA0363	MOD:00368	natural	modified residue	blocked amino end
0166	OAS	AA0364	MOD:00369	natural	modified residue	
0167		AA0366		natural	crosslink6	calcium, manganese, metalloprotein
0168		AA0367	MOD:00372	natural	crosslink2	
0169		AA0368	MOD:00373	natural	crosslink2	
0170		AA0372	MOD:00377	natural	attachment	phosphoprotein
0171	CRU	AA0378	MOD:00383	hypothetical	crosslink1	chromoprotein
0172	CRU	AA0183	MOD:00191	natural	modified residue	
0173	CH6	AA0379	MOD:00384	natural	crosslink1	chromoprotein
0174	CH6	AA0183	MOD:00191	natural	modified residue	
0175	NRQ	AA0379	MOD:00384	natural	crosslink1	chromoprotein

Tabelle A.7: Modifikationen Liste 7

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0176	NRQ	AA0183	MOD:00191	natural	modified residue	
0177	NYG	AA0380	MOD:00385	hypothetical	crosslink1	
0178	NYG	AA0183	MOD:00191	natural	modified residue	
0179	CR7	AA0381	MOD:00386	hypothetical	crosslink1	
0180	CH7	AA0382	MOD:00387	natural	crosslink1	
0181	CR7	AA0183	MOD:00191	natural	modified residue	
0182		AA0395	MOD:00802	natural	attachment	metalloprotein, vanadium
0183		AA0396	MOD:00803	natural	crosslink2	thioether bond
0184		AA0411	MOD:00821	hypothetical	crosslink2	blocked carboxyl end, thioester bond
0185		AA0412	MOD:00822	natural	crosslink2	blocked carboxyl end, thioester bond
0186		AA0413	MOD:00823	natural	crosslink2	blocked carboxyl end, thioester bond
0187		AA0414	MOD:00825	natural	crosslink2	blocked carboxyl end, thioester bond
0188		AA0415	MOD:00826	hypothetical	crosslink2	blocked carboxyl end, thioester bond
0189		AA0416	MOD:00827	hypothetical	crosslink2	blocked carboxyl end, thioester bond
0190		AA0417	MOD:00828	natural	crosslink2	blocked carboxyl end, thioester bond
0191		AA0418	MOD:00829	natural	crosslink2	blocked carboxyl end
0192	TH5	AA0423	MOD:01171	natural	modified residue	
0193		AA0428	MOD:01142	natural	attachment	chromoprotein, dihydrobiliverdin, thioether bond
0194		AA0429	MOD:01143	natural	crosslink2	chromoprotein, dihydrobiliverdin, thioether bond
0195		AA0430	MOD:01176	natural	crosslink2	
0196	HIQ	AA0431	MOD:01177	natural	modified residue	
0197	OHS	AA0432	MOD:01178	natural	modified residue	
0199		AA0436	MOD:01182	natural	crosslink2	phosphoprotein, FAD, flavoprotein, thioether bond
0200		AA0437	MOD:01183	hypothetical	crosslink2	redox-active center, selenium

Tabelle A.8: Modifikationen Liste 8

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0201		AA0438	MOD:00864	natural	crosslink4	2Fe-2S, metalloprotein
0202	LED	AA0444	MOD:01374	hypothetical	modified residue	
0204	ILX	AA0449	MOD:01378	natural	modified residue	hydroxylation
0205		AA0451	MOD:01380	natural	crosslink2	hydroxylation
0207		AA0358	MOD:00363	natural	crosslink2	redox-active center, selenium, selenocysteine
0208	FGL	AA0458	MOD:01384	hypothetical	modified residue	
0210	OLT	AA0464	MOD:01387	natural	modified residue	methylated amino acid
0213		AA0472	MOD:01395	natural	crosslink2	blocked amino end
0214	AHB	AA0478	MOD:01401	natural	modified residue	hydroxylation
0215		AA0490	MOD:01442	natural	crosslink2	
0217	CXM	AA0493	MOD:01446	hypothetical	modified residue	thioether bond, blocked amino end, formylation, pretranslational modification
0218	PED	AA0494	MOD:01454	natural	modified residue	phosphoprotein, blocked amino end
0219		AA0495	MOD:01585	natural	crosslink2	blocked carboxyl end
0220		AA0496	MOD:01586	natural	crosslink2	blocked carboxyl end
0221		AA0497	MOD:01587	natural	attachment	phosphoprotein
0222		AA0501	MOD:01602	natural	crosslink2	
0223	IYR	AA0509	MOD:01612	natural	modified residue	iodine
0224	TYI	AA0510	MOD:01613	natural	modified residue	iodine
0225		AA0513	MOD:01616	natural	crosslink2	
0226	0AF	AA0520	MOD:01664	natural	modified residue	hydroxylation

Tabelle A.9: Modifikationen Liste 9

ID	PDBCC	RESID ID	PSI-MOD	Auftreten	Kategorie	Schlüsselwörter
0227		AA0522	MOD:01668	natural	attachment	phosphoprotein, FAD, flavoprotein
0228	AEI	AA0525	MOD:01671	natural	modified residue	
0229		AA0041	MOD:00050	natural	attachment	acetylated amino end
0230		AA0042	MOD:00051	natural	attachment	acetylated amino end
0231		AA0043	MOD:00052	natural	attachment	acetylated amino end
0232		AA0044	MOD:00053	natural	attachment	acetylated amino end
0233		AA0046	MOD:00055	natural	attachment	acetylated amino end
0234		AA0050	MOD:00059	natural	attachment	acetylated amino end
0235		AA0051	MOD:00060	natural	attachment	acetylated amino end
0236		AA0052	MOD:00061	natural	attachment	acetylated amino end
0237		AA0053	MOD:00062	natural	attachment	acetylated amino end
0238		AA0054	MOD:00063	natural	attachment	acetylated amino end
0239		AA0354	MOD:00359	natural	attachment	acetylated amino end
0240		AA0081	MOD:00090	natural	attachment	amidated carboxyl end
0241		AA0083	MOD:00092	natural	attachment	amidated carboxyl end
0242		AA0084	MOD:00093	natural	attachment	amidated carboxyl end
0243		AA0085	MOD:00094	natural	attachment	amidated carboxyl end
0244		AA0086	MOD:00095	natural	attachment	amidated carboxyl end
0245		AA0087	MOD:00096	natural	attachment	amidated carboxyl end
0246		AA0088	MOD:00097	natural	attachment	amidated carboxyl end
0247		AA0089	MOD:00098	natural	attachment	amidated carboxyl end
0248		AA0091	MOD:00100	natural	attachment	amidated carboxyl end
0249		AA0092	MOD:00101	natural	attachment	amidated carboxyl end
0250		AA0095	MOD:00104	natural	attachment	amidated carboxyl end
0251		AA0099	MOD:00108	natural	attachment	amidated carboxyl end
0252				natural	undefined	Metal coordination

Tabelle A.10: Modifikationen Liste 10

Anhang B: Proteinsequenzdatenbanken

Datenbankname	Beschreibung
nr	Non-redundant GenBank CDS translations + PDB + SwissProt + PIR + PRF
refseq	Proteinsequenzen vom NCBI Reference Sequence Projekt.
swissprot	SWISS-PROT Proteinsequenzdatenbank
pat	Patent division der GenBank.
month	GenBank CDS translations + PDB + SwissProt + PIR + PRF Veröffentlichungen der letzten 30 Tage.
pdb	Protein Data Bank.
env_nr	Non-redundant CDS translations vom env_nt Einträgen.

[28] entnommen [19]

Anhang C: Nukleotiddatenbanken

Datenbankname	Beschreibung
nr	GenBank + EMBL + DDBJ + PDB Sequenzen (aber nicht EST, STS, GSS, Phase 0,1 oder 2 HTGS-Sequenzen).
refseq_mrna	mRNA Sequenzen vom NCBI Reference Sequence Project.
refseq_genomic	Genom Sequenzen vom NCBI Reference Sequence Project.
est	Database of GenBank + EMBL + DDBJ Sequenzen vom EST Division.
est_human	Teilmenge vom Menschen der EST.
est_mouse	Teilmenge der Maus der EST.
est_others	Teilmenge der EST ohne Mensch und Maus.
gss	Genome Survey Sequence, beinhalten „single-pass genomic“ Daten, „exon-trapped“ und „Alu PCR“ Sequenzen.
htgs	„Unfinished High Throughput Genomic Sequences“: Phasen 0, 1 und 2 sind Fertig, Phase 3 HTG sind in nr.
pat	Nucleotide vom der Patentabteilung der GenBank.
pdb	Protein Data Bank
month	Alle neuen oder überarbeiteten GenBank+EMBL+DDBJ+PDB Sequenzen der letzten 30 Tage
alu_repeats	Selektion aller Alu Wiederholungen der REPBASE.
dbsts	„Database of Sequence Tag Site“ Einträge von der STS Abteilung der GenBank + EMBL + DDBJ.
chromosome	Komplette Genome und Chromosomen vom NCBI Reference Sequence project. Überschneidung mit refseq_genomic.
wgs	Assemblies der „Whole Genome Shotgun“ Sequenzen.
env_nt	Sequenzen von Umweltproben. Überschneiden sich nicht mit nr.

[28] entnommen [19]

Literaturverzeichnis

[1] Bioinformatics Application Note,Vol. 24 no 18, 2008

[2] url:http://genome.ucsc.edu/FAQ/FAQformat.html, verfügbar am 16.06.2011

[3] Graw,Jochen: Genetik - 5. Auflage : Springer, 2010

[4] Lawrence, Eleanor: Henderson's dictionary of Biology - 13. Auflage - London : Pearson Education Limited, 2005

[5] Merkl, Rainer; Waack, Stephan: Bioinformatik Interaktiv: Algorithmen und Praxis - 1. Auflage : Wiley-VCH, 2003

[6] Journal of Molecular Biology. Nr. 147, 1981

[7] Journal of Molecular Biology. Nr. 48, 1970

[8] Pevsner, Jonathan: Bioinformatics and Functional Genomics - 2. Auflage: Wiley-Blackwell, 2009

[9] Janec, Jan E.: Molecular and Genomic Data Identify the Closest Living Relative of Primates. In: Science, Band 318, 2007

[10] Linderstrøm-Lang, K.U.: Proteins and Enzymes. In: Lane Medical Lectures. Bd. 6, S. 1-115. Stanford University Publications, University Series, Medical Sciences, Stanford University Press.

[11] url: http://www.brighthub.com/science/genetics/articles/21915/image/48340/, verfügbar am 15.08.2011

[12] Merkl, Rainer; Waack, Stephan: Bioinformatik Interaktiv: Grundlagen, Algorithmen, Anwendung - 2. Auflage : Wiley-VCH, 2009

[13] Koolman, J.;Röhm, K.-H.: Taschenatlas der Biochemie - 1. Auflage : Thieme, 1994

[14] url:http://www.ebi.ac.uk/2can/tutorials/aa.html, verfügbar am 10.08.2011

[15] Amino acid substitution matrices from protein blocks. In: Proc. Natl. Acad. Sci. USA Vol. 89, 1992

[16] Sokal, R.; Michener, C:A statistical method for evaluating systematicrelationships. In: University of Kansas Science Bulletin 38

[17] url:http://biojava.org/wiki/BioJava:CookBook:Core:Overview, verfügbar am 10.07.2011

[18] url:http://www.rcsb.org/pdb/home/home.do, Stand vom 27.09.2011

[19] url: http://blast.ncbi.nlm.nih.gov/Blast.cgi? CMD=Web&PAGE_TYPE=BlastDocs &DOC_TYPE=ProgSelectionGuide , verfügbar am 25.09.2011

[20] Tool: Netbeans 7.x

[21] Löffler, Georg; Petrides, Petro E.; Heinrich, Peter C.: Biochemie und Pathobiochemie - 8. Auflage : Springer , 2007

[22] Murzin, Alexey G.; Brenner, Steven E.; Hubbard, Tim; Chothia , Cyrus: SCOP: A Structural Classification of Proteins Database for the Investigation of Sequences and Structures. In: Journal of Molecular Biology, Volume 247, 1995

[23] Shindyalov, Ilya N.; Bourne, Philip E.: Protein structure alignment by incremental combinatorial extention (CE) of the optimal path. In: Protein Engineering, Volume 11 Nr. 9, 1998

[24] Yuzhen, Ye; Godzik, Adam: Flexible structure alignment by chaining aligned fragment pairs allowing twists. In: Bioinformatics, Volume 19 Suppl. 2, 2003

[25] wwpdb: Protein Data Bank Contents Guide: Atomic Coordingate Entry Format Description: Version 3.30, 2008

[26] url:http://www.jgraph.com/mxgraph.html, verfügbar am 19.08.2011

[27] Popa, Radu: Between Necessity and Probability - Searching for the Definition and Origin of Life : Springer : 2004

[28] Han, Mira V.; Zmasek, Christian M.:phyloXML - XML for evolutionary biology and comparative genomics : BioMed Central Bioinformatics Nr. 10 - 356 : doi:10.1186/1471-2105-10-355 : 2009

[29] Cristianini, Nello; Hahn, Matthew W.:Introduction to Computational Genomics - A Case Studies Approach : Cambridge University Press : 2006

[30] url:http://www.rcsb.org/pdb/images/1pqs_bio_r_500.jpg, verfügbar am 05.08.2011

[31] url:http://biojava.org/wiki/BioJava:CookBook, verfügbar am 01.04.2011

[32] url:http://www.biojava.org/docs/api/index.html, verfügbar am 01.04.2011

[33] url:http://biojava.org/wiki/BioJava:MailingLists, verfügbar am 01.04.2011

[34] Cerami, Ethan: XML for Bioinformatics : Springer : 2005

[35] Bal, Harshawardhan; Hujol, Johnny: Java for bioinformatics and biomedical applications : Springer : 2007

[36] Hill, Patricia M. ;Warren, David S.: Logic Programming - 25th International Conference, ICLP 2009, Pasadena, CA, USA,July 2009, Proceedings : Springer : 2009

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 21. November 2011