# MARSHALL PLAN REPORT

## Visual display of the 9-state profile hidden markov model

prepared for the
Austrian Marshall Plan Foundation

in cooperation with
Salzburg University of Applied Sciences
Information Technology & Systems Management
and
Bowling Green State University
College of Technology

submitted by:
**Martin Schnöll**


Supervisor: FH-Prof. Univ.-Doz. Dr. Stefan Wegenkittl

Salzburg, March 2011

# Details


| | |
|---|---|
| First Name, Surname: | Martin Schnöll |
| Home University: | Salzburg University of Applied Sciences |
| Degree Program: | Information Technology & Systems Management |
| Host University: | Bowling Green State University |
| Title of Thesis: | Visual display of the 9-state profile hidden markov model |
| Academic Supervisor: | FH-Prof. Univ.-Doz. Dr. Stefan Wegenkittl |


# Keywords


| | |
|---|---|
| $1^{st}$ Keyword: | Bioinformatics |
| $2^{nd}$ Keyword: | Sequence Alignment |
| $3^{rd}$ Keyword: | Profile Hidden Markov Model |
| $4^{th}$ Keyword: | Sequence Logo |


# Abstract


Profile Hidden Markov Models (HMMs) are often used in the field of bioinformatics. They allow for the comparison of different proteins in order to determine similar characteristics and functions. Proteins can be represented by a chain of characters and each of these characters symbolizes one of the twenty natural occurring Amino Acids (AAs). The purpose of Profile HMMs is that such a model is generated for a specific protein family and then any protein sequence can be aligned to this model in order to determine the degree of the similarity. The main focus of this report is on the implementation of a visual display of the 9-state Profile Hidden Markov Model (HMM) and on the rendering of transition and emission probabilities in user friendly ways. Furthermore, this visual display will be integrated in the HMModeler protein modelling software developed at Salzburg University of Applied Sciences and Salzburg University.

# Contents

# List of Figures

# List of Tables

# Listings

# 1

# Introduction

## 1.1 Preview

Nowadays, information technologies are indispensable in supporting molecular biology. The amount of data to be processed in this field is enormous and can only be treated effectively with the help of computers.

There are a lot of different software products and tools available for bioinformatics. This thesis deals with some of these tools, especially with the UCSF Chimera [8] add-on HMModeler [12] protein modelling software developed at Salzburg University of Applied Sciences and Salzburg University. The goal is to extend the current software by a visual display. This visual display will illustrate the 9-state Profile HMM which is a prominent tool in the field of bioinformatics and allows for the comparision of different proteins in order to determine similar characteristics and functions. Furthermore, transition and emission probabilities will be rendered in user friendly ways.

First of all, fundamentals about proteins and Profile HMM will be explained in the first chapter. Second, the programming language Python and relevant bioinformatic software will be presented. Furthermore, Chapter 3 is the main part which deals with the implementation of the visual display itself and Chapter 4 consists of the future outlook, including facts about the integration in the HMMModeler and information on sequence logos, which are another way of comparing different proteins.

## 1.2 Macromolecules

Macromolecules like Deoxyribonucleic Acids (DNAs) or proteins play a major role in biology. The number of different macromolecules is enormous and molecular biology analyzes the different characteristics and functions of such molecules.
Macromolecules consist of a sequence of certain components. Proteins, which are the main focus of this paper, consist of a sequence out of the twenty naturally occurring AAs. Table 1.1 lists the twenty naturally occuring AAs under specification of the one letter code and the three letter code. Especially the one letter code is relevant for protein sequences.

| **one letter code** | **three letter code** | **name of the Amino Acid (AA)** |
|---|---|---|
| A | Ala | Alanine |
| C | Cys | Cysteine |
| D | Asp | Aspartic Acid |
| E | Glu | Glutamic Acid |
| F | Phe | Phenylalanine |
| G | Gly | Glycine |
| H | His | Histidine |
| I | Ile | Isoleucine |
| K | Lys | Lysine |
| L | Leu | Leucine |
| M | Met | Methionine |
| N | Asn | Asparagine |
| P | Pro | Proline |
| Q | Gln | Glutamine |
| R | Arg | Arginine |
| S | Ser | Serine |
| T | Thr | Threonine |
| V | Val | Valine |
| W | Trp | Tryptophan |
| Y | Tyr | Tyrosine |

Table 1.1: Alphabet for protein sequences [6, p.31]

In general, the most suitable method to determine the characteristics and functions of proteins is analyzing their topology or physical structure. However, this technique involves a lot of work and equipment. Furthermore, as described in [6] its sequence entirely determines the 3D-structure of a protein. Therefore, bioinformatics work with

sequences instead of structures wherever possible because they are much easier and more cost-efficient to determine. These sequences are a chain of characters with a finite length and every character represent a certain AA.
A common technique to determine specific protein characteristics, especially to determine the similarity of proteins, is sequence alignment.

## 1.3 Sequence Alignment

Sequence alignment is a method to align a certain sequence to other sequences to determine their similarity. However, sequence alignment is not trivial to perform. Sequences usually do not have the same length and it is also expensive to find out which AA or character of the sequence matches with another AA. Sequences have changed their composition through evolution. Therefore, there might be some new characters in the sequences and there might have been also some deletions of AAs. Although the difference in terms of character composition and single deletions and insertions might be significant, tertiary (3D) structure and function as such are often highly preservered.
Basically, there are two distintive alignment methods: The pairwise sequence alignment and the Multiple Sequence Alignment (MSA). As the name suggests, the MSA works with three or more protein sequences. This alignment method is very cost-intensive and complex algorithm are necessary. However, to accurately describe protein families, which normally consists of a great deal of different sequences, MSA is essential. Profile HMMs, which are the focus in the next chapter, work with such Multiple Sequence Alignments (MSAs).

## 1.4 Profile Hidden Markov Models

A method to align sequences and to numerically determine similarities in the form of similarity measures is provided through HMMs. In bioinformatics, so-called Profile HMMs are of particular importance [4].
The basic idea of a so-called Profile HMM is that such a model is created for a specific

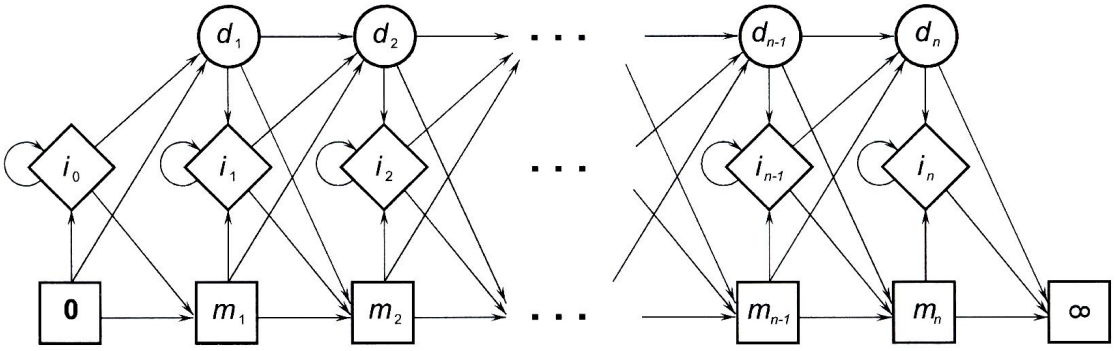


Figure 1.1: Profile HMM, [6, Chapter 16]

protein family and then any given query sequence can be aligned to this specific Profile HMM. The result of this process is a score which describes the similarity of the query sequence to the protein family. A high score usually relates to a high similarity.
Basically, the core of a Profile HMM as described in [4] consists of three types of states: match states, insert states and delete states. These states are connected with arrows representing possible transitions between the states.
Generally, insert and match states emit symbols of the query sequence, whereas delete states are silent states used for shortcuts in the model.
As an example Figure 1.1 shows a Profile HMM with the match states $m_1$,...,$m_n$, insert states $i_0$,...,$i_n$ and delete states $d_1$,...,$d_n$.

Match states are symbolized as squares, insert states as rhombuses and delete states as circles. Therefore, the first square in Figure 1.1, which acts as the starting point of the model, is equivalent to the match state $m_0$ and the last square, which acts as the end point of the model, represents the match state $m_\infty$. What is not shown in Figure 1.1 are the flanking states at the beginning and the end of the Profile HMM. These states are responsible for the initialization and the termination of the process and will be part of the visual display as shown in Chapter 3.
If, during the alignment of the query sequence, the model is in the match state $m_i$, this means that the AA or current character $x_j$ of the query sequence is generated with respect to the emission frequencies of the respective state. This distribution is derived from proteins that are known to belong to the protein family modeled by the HMM. By this, AAs that occur with a high frequency in those sample proteins will

produce a higher local score if emitted by the match state. In insert states, a so-called background distribution is used instead and no position specific scores are generated by emissions in such states. A high global score is thus achieved if many match states can reproduce parts of the query sequence with high local scores. The degree of the matching is defined through the emission frequency of the match state.
The insert state $i_i$ describes the insertion of an AA in the query sequence through evolution. If a new AA was inserted in the query sequence, that does not mean that the function of this particular protein has necessarily changed or that this protein is not part of the protein family anymore. Therefore, the process goes into the insert state and a character, according to the emission frequencies of this state, is emitted. The following example in Table 1.2 should clarify this process.

| **query sequence:** | S | E | A |
|---|---|---|---|
| **MSA:** | S | - | A |
| **HMM:** | match | insert | match |

Table 1.2: Example of an insert state

Let us suppose that the sequence "SA" is part of the MSA and the part of the query sequence, which is aligned to the model, is "SEA" like in Table 1.2. The letters and AAs "S" and "A" of the query sequence are matching well with the MSA at this position. However, the letter "E" is new and was inserted through evolution. Therefore, the process goes from the match state $m_i$ (letter "S") in the insert state $i_i$, where the letter "E" is emitted. Finally, from the insert state $i_i$ the process goes into the match state $m_{i+1}$, where the letter "A" is emitted.
The third state of the Profile HMM is the delete state. This state is relevant if, through evolution again, an AA was deleted in the query sequence, but still occurs in the MSA. To demonstrate this situation, consider the following example in Table 1.3.

| **query sequence:** | P | - | F |
|---|---|---|---|
| **MSA:** | P | T | F |
| **HMM:** | match | delete | match |

Table 1.3: Example of an delete state

Let us suppose that the sequence "PF" of the query sequence is aligned to the sequence "PTF". So, the letter and AA "T" is missing in the query sequence, e.g. by having been deleted through evolutionary processes. Therefore, the Profile HMM goes from the match state $m_i$ (letter "P") into the delete state $d_{i+1}$, where no AA but a gap (letter "-") is emitted. Furthermore, the process goes into the match state $m_{i+2}$ and "F" is emitted.

# 2

# Python and Bioinformatics

## 2.1 Why Python?

Mitchell L. Model describes the programming language Python in his book [7] as a "beautiful language". Python is effective and easy to learn for new programmers, but it is also practical in the advanced computer science field. On the one hand, simple scripts can be written effortlessly and, on the other hand, also sophisticated advanced applications are feasible. However, for programmers with experience in other programming languages like C++ or Java, the transition to work with Python might be confusing and curious at the beginning. The main reason for this is based on the different and the uncommon syntax of the language. In Python the end of statements is not marked by a semicolon or any other similar character. Only the end of the line signalizes the end of a statement. Furthermore, there are no braces used for grouping statements into functions or methods like in other languages. This functionality is given through the indention of compound statements relative to the lines of code that introduce them. Also classes are only defined through such indentions. [7]
Listing 2.1 shows a typical Python class with the indented expressions for defining the class and the two functions.

```
1 class Dog:
2         def run(self):
3                 print('I am running!')
4         def bark(self):
5                 print('I am barking!')
```

Listing 2.1: Simple Python Class

Through this indention and the abandonment of semicolons, terminal keywords and braces, which primarily are for the benefit of the compiler, Python code is easier to read to the human eye. Mitchell L. Model describes this fact like this: "Python frees the programmer from the drudgery of serving as a compiler assistant." [7, p.XVII]
This makes it easier for new programmers like biologists without any programming skills to work with Python. Python is a powerful language. "Its skeleton is procedural, and it has been significantly influenced by functional programming, but it has evolved into a fundamentally object-oriented language." [7, p.XVII]
Another important aspect, why Python is often used in bioinformatics, is the fact that Python is free to use, even for commercial products.
An example for a bioinformatics related software written in Python is shown in the next chapter with the extensible molecular modeling system Chimera, developed at University of California, San Francisco (UCSF).

## 2.2 UCSF Chimera

*UCSF Chimera*[1] is an extensible visualization system, developed by the Computer Graphics Laboratory at UCSF. In comparison to their first molecular visualization system called MMS/MIDS in 1976, the primary goal while implementing Chimera was extensibility. Furthermore, Chimera is portable to a wide variety of platforms, it includes state-of-the-art graphics capabilities and it provides both, a graphical menu and a command-line interface.
To ensure the extensibility, the software is divided into a core, which provides basic

[1] http://www.cgl.ucsf.edu/chimera

services and molecular graphics capabilities, and extensions, which provide higher level functionality.

This design of using extensions as a main part of Chimera makes sure "that the extension mechanism is robust enough to handle the needs of outside researchers wanting to extend Chimera in novel ways." [8, p.2]

Additional to the Python layer, the Chimera core also consists of a C++ layer for time-critical operations like graphics rendering. However, all C++ functions are accessible also from the Python layer. Extensions for Chimera either have to be written entirely in Python or in a combination of Python and C/C++. [8]

The main focus of this thesis is on the HMModeler extension for Chimera developed at Salzburg University of Applied Sciences and Salzburg University.

## 2.3 HMModeler

HMModeler is part of a toolset for UCSF Chimera, which was developed in a common effort by Salzburg University of Applied Sciences and Salzburg University, that allows an efficient computing of multiple structure alignments for protein families and includes also corresponding Profile HMMs. Whereas PSC++, the second part of the toolset, is responsible for building multiple structure alignments from alternative pairwise solutions, HMModeler provides a comfortable GUI for editing the alignments. Furthermore, the user is also able to add supplementary biological information to the model. First, predetermined breaking points, specified through the acceptance of insertions and deletions, the degree of conservation and AA sets are getting fine-tuned column-wisely. This ensures that it is possible to determine the skeleton of the HMM without observing the stochastic nature and the numerous transition and emission probabilities of the model itself. Finally, HMModeler then generates a 9-state Profile HMM for the protein family. [12]

What was not included so far in the HMModeler extension is a user-friendly visual display of the model. This was the aim of this work and is presented in the following chapter.

# 3

# Visual display of the 9-state Profile Hidden Markov Model

This chapter deals with the implementation of the visual display of the 9-state Profile HMM. First of all, the implementation with all states including start, end and flanking states will be explained. Furthermore, the application itself including class diagrams will be outlined.

## 3.1 Implementation

As mentioned in 1.4 the implemented visual display, which will be integrated into the HMModeler, also includes start, end and flanking states which are all silent. Silent means that this states do not emit symbols, whereas non silent states like all match and insert states emit symbols. Figure 3.1 shows a customized screenshot of the visual display with all the start, end and flanking states. Also the emission frequencies of the match states are shown.
As described in [10] and [11] the Profile HMM consists of the following states, which are significant if the query sequence s is aligned to the HMM of a protein family:

- Begin State (B):
  This is the starting state for each alignment process of s to the HMM.
- Q-state Begin (QB):
  This Q-state represents possible initial parts of s that are not aligned to the

match or insert states in the core of the HMM.

- Flanking Begin (FB):

  This silent state is essential in order that a transition is possible to any of the match states. The probabilities of these transitions, the so-called Introtransitions, are shown below every match state in red, e.g. "-> 0.7" in the first column of Figure 3.1.

- Match State (M):

  Every column has its match state, which is non silent and therefore emit a symbol. If the model is in the match state while s is aligned to the HMM, this means that the AA is generated with respect to the emission frequencies of the respective state.

- Insert State (I):

  Every column except the last column has its insert state, which is non silent and therefore emit a symbol. The insert states represent evolutionary insertions of AAs in the query sequence s.

- Delete State (D):

  Every column except the first and the last column has its delete state, which is silent and therefore do not emit a symbol. The delete states represent evolutionary deletions of AAs in the query sequence s.

- Flanking End (FE):

  This is a silent state which acts as a collecting point for a transition from any match state to the final Insert and End state. The probability of these transitions is shown below every match state in red, e.g. "-> 0.00316" in the first column of Figure 3.1.

- Q-state End (QE):

  This Q-state represents possible end parts of s that are not aligned to the match or insert states in the core of the HMM.

- End State (E):

  This is the end state for each alignment process of s.

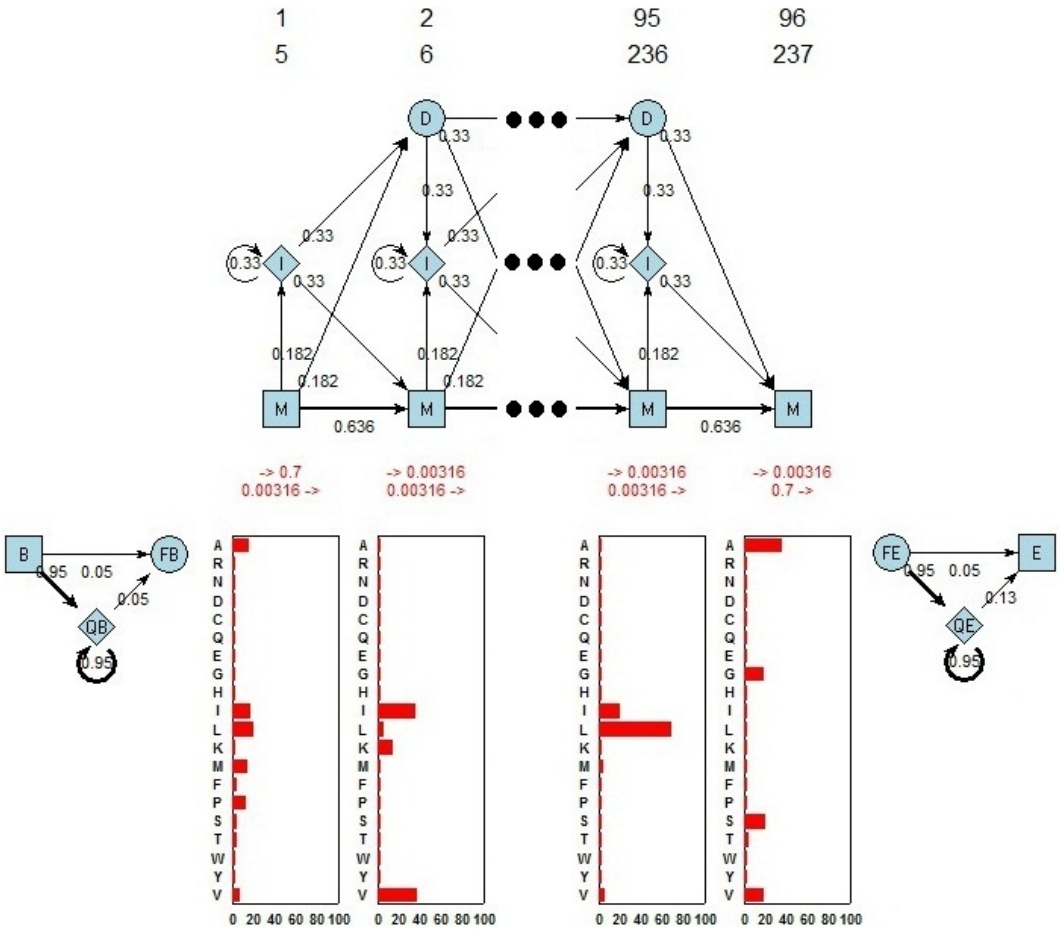


Figure 3.1: Customized screenshot of the visual display

Also included in the visual display are histograms which show the emission frequencies of the match states. This distribution is derived from the MSA. If the HMM is in the match state during the alignment process, AAs that occur with a high frequency in the MSA will produce a higher local score. High local scores result in a high global score and this again means that the degree of the matching between the query sequence and the MSA is higher. Therefore, the probability is higher that the aligned protein is part of the from the HMM represented protein family.

## 3.2 Coding

The UCSF Chimera extension HMModeler was written in Python 2.5 and therefore also the visual display was programmed in Python 2.5 to ensure that the visual display

can be integrated into the current software. Furthermore, to make the integration easier, the visual display was programmed with the use of the design pattern Model-View-Controller (MVC).

## 3.3 Model-View-Controller

The MVC design pattern consists of three main parts. The model is the application object and contains the necessary data, the view is responsible for the presentation of the data on the output device like the screen and the controller defines the way how the application reacts to user inputs. This design pattern ensures that the applications is divided in reasonable sections. Every software engineer who might have to work with an application which was developed with the use of MVC, knows where he or she can change the data model (Model), the screen presentation (View) or the interaction with the user (Controller). [3]
What is different in the current version of the application to the MVC is that currently, the controller just initializes the model and then calls the view as shown in Listing 3.1. So the view also handles the user inputs. The reason for that is that just the view works with the standard library module Tkinter, which is a portable library for constructing a Graphical User Interface (GUI) [5]. Furthermore, through the later integration into the HMModeler the user interface of the visual display will be changed and adapted to the requirements of the HMModeler anyway.

```
1 import model
2 import view
3
4 __hmm = model.Model()
5 __window = view.View(__hmm)
```

Listing 3.1: controller.py

The model manages and contains all the data which is necessary for producing the visual display. This includes, among other data, the emission frequencies of all match states, the transition frequencies between the states, the Intro- and Outrotransitions and information on which columns of the MSA are used for creating the HMM. This information is essential, because not all MSA columns are significant for the HMM. MSA columns which will not be included in the HMM are often columns with gaps resulting from insertions or deletions in the sequences. These columns are not really representative for the protein family and will therefore not be included in the HMM. Which columns of the MSA are used in the HMM are apparent on the top of the visual display shown in Figure 3.1. Column 1 represents column 5 of the MSA, column 2 represents column 6 of the MSA and so forth. Usually, there are a lot of columns which are not used to model the protein family and therefore not relevant for the MSA, e.g. in Figure 3.1 column 96 represents column 237 of the MSA.
Besides the storage of the data, the model is also responsible for reading the data from a file. At the moment, the stand-alone application reads all the necessary data from a .hmm-File. Later on, in the HMModeler this should happen automatically and no extra saving and reading operations of a .hmm-File should be necessary to create the visual display.

Figure 3.2 shows the class diagram of the model. All attributes are private and are only accessible through the respective get method. All attributes of the model class are being read from a .hmm-File and this assignment is completed by the method readHMMFile(in filename: String). For being more flexible in the reading process regarding to the structure of the input file, every section in the file relates to a specific state in the reading process, e.g. the state "EF" means that currently the emission frequencies are being read. This ensures that also a random arrangement of the different sections in the file is supported and does not result in an error. This functionality is given through the private method checkState(in line: String, in previous_state: String), which searches for the section keywords like "EF" or "Introtransitions" in every line and changes the current reading state if necessary.

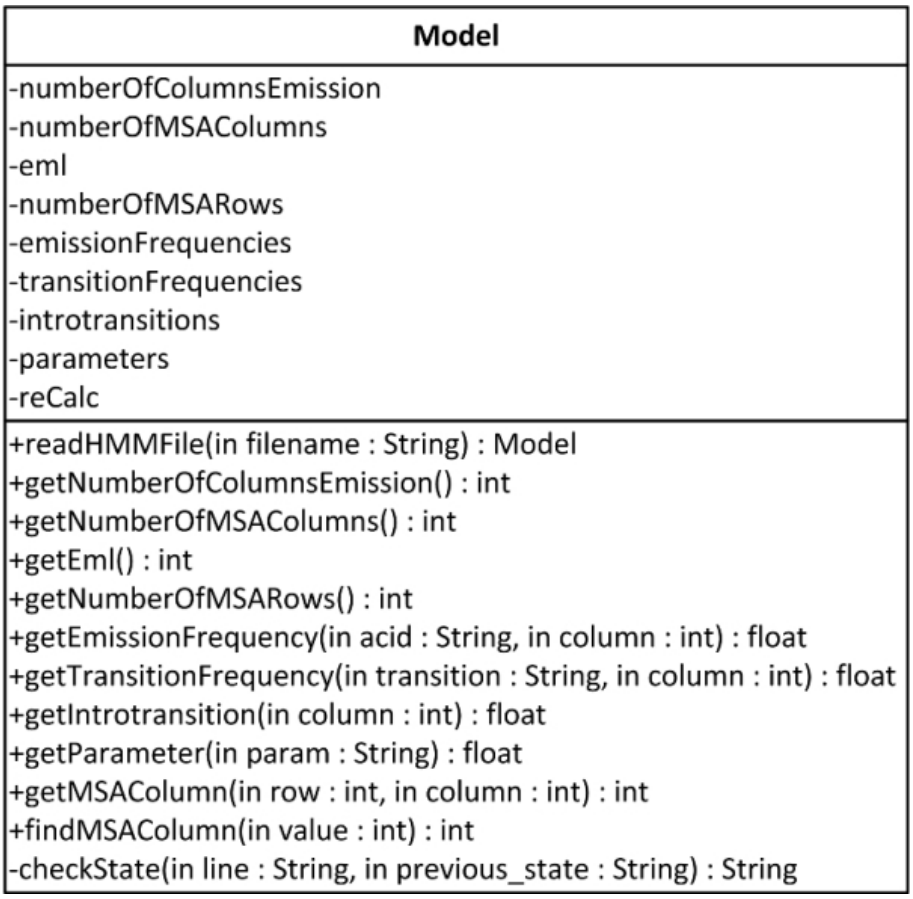


Figure 3.2: Class diagram of the model

Figure 3.3 shows the class diagram of the view. The view is responsible for creating and drawing the visual display itself including all graphical elements like the squares, arrows and histograms. Furthermore, the view also handles the interaction with the user through the menu bar. The menu bar consists of three menu items: File, View and Help. The ⟨*file*⟩ menu item is used for reading a file and for terminating the application. Through the ⟨*view*⟩ menu item it is possible to set the focus to a specific column or alternatively to a specific MSA value. This option can be used as an alternative to the scrollbar and is helpful if the HMM consists of a large number of columns. The ⟨*help*⟩ menu item just displays an about box with information on the visual display.
As shown in Figure 3.3, the canvas is the drawing page of the visual display. The method plotHMM() is the main method and draws all the graphical elements including the histograms for the emission frequencies.

## 3.4 Environment

The algorithm was tested on the operating system Microsoft Windows 7 Professional. For source code development the software *Eclipse SDK 3.6.0*[1] and the Python Integrated Development Environment (IDE) *Pydev*[2] was used. The whole setup was tested on a Toshiba Satellite A100-733 laptop with an Intel Core 2 CPU with 1.83GHz and 3GB RAM.

[1] http://www.eclipse.org
[2] http://pydev.org

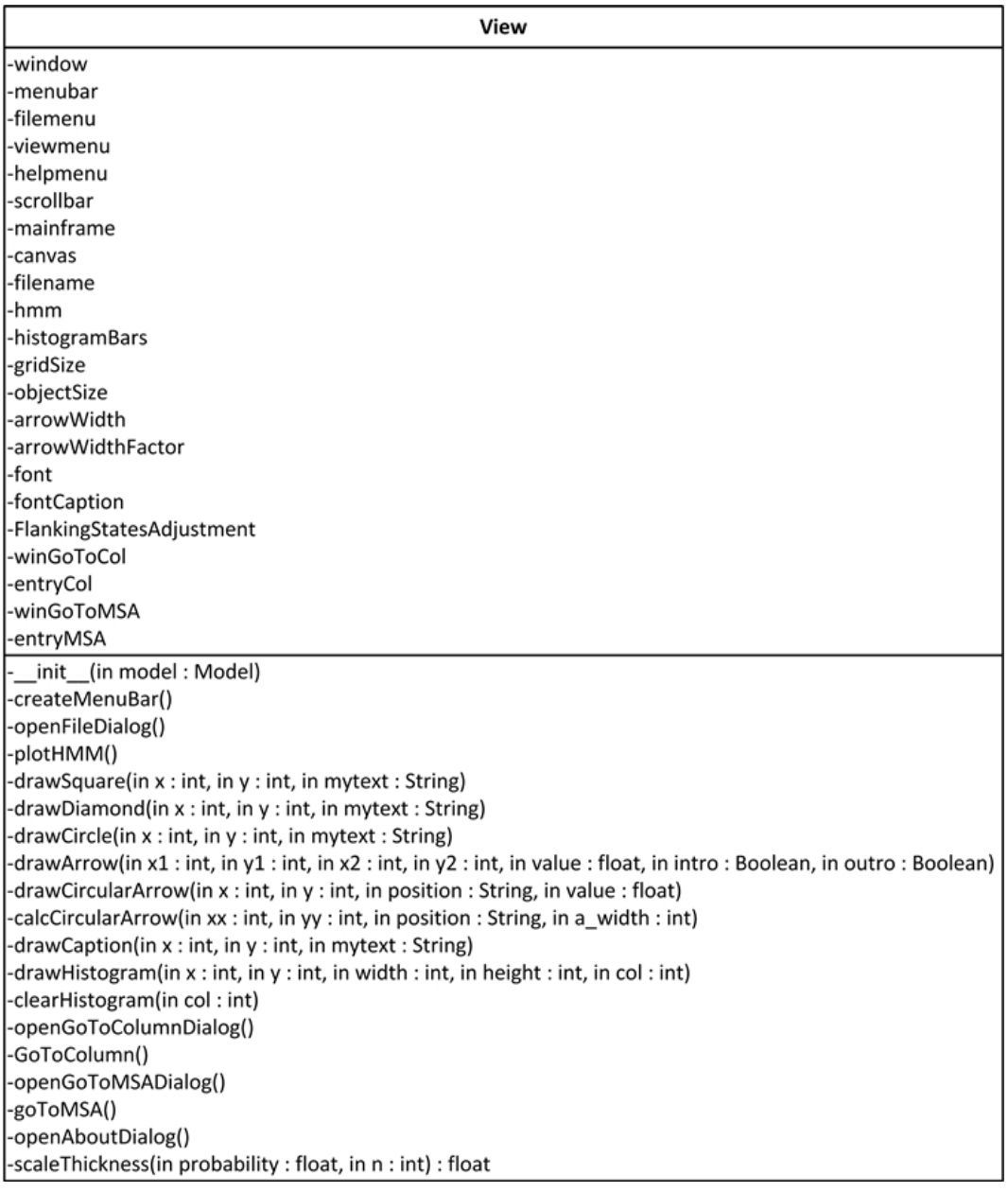


Figure 3.3: Class diagram of the view

# 4

# Outlook

## 4.1 Integration in HMModeler

The visual display of the Profile HMM will be integrated in the UCSF Chimera extension HMModeler later on. Currently, a research group of the Information Technology and Systems Management Masters program at Salzburg University of Applied Sciences works on upgrading the HMModeler. They will use the visual display developed for this paper and will integrate it into the current software.

As described in [9], UCSF Chimera is a highly extensible program for interactive visualization and analysis of molecular structures and related data. High-quality images and animations can be generated. Furthermore, Chimera includes complete documentation and can be used for free in an academic manner. These are only a few reasons why this software was chosen for developing the HMModeler.

One major requirement on the visual display was that the application reacts to changes on the Profile HMM at run time with good performance. HMModeler itself allows that the model can be edited and fine adjusted easily by adding supplementary biological information. Therefore, also the visual display has to support this feature in order to avoid performance problems. It could be necessary that single values, but also larger arrangements like entire columns or histograms have to be updated, added or even deleted. It would be undesirable that the entire visual display has to be redrawn with each change. Therefore, specific methods in the application have to ensure that all objects in the visual display can be updated, added and deleted after the entire Profile

HMM was drawn for the first time. One example for such a necessary method is shown in Figure 3.3. The method clearHistogram(in col : int) allows, under specification of the column number, the erasing of a entire histogram. If, for example, the distribution of the emission frequencies of a specific match state was changed by the user in the HMModeler, this method can be used to both erase the old histogram and draw a new and updated histogram by calling the method drawHistogram(in x : int, in y : int, in width : int, in height : int, in col : int).

## 4.2 Sequence Logos

Another way for illustrating MSAs is provided by sequence logos. As described in [2], sequence logos were developed by Tom Schneider and Mike Stephens and are a graphical representation of a MSA. Such a logo consists of stacks of symbols and each stack represents a specific position in the sequence. Each stack includes two significant magnitudes. On the one hand, the sequence conservation, which is indicated by the overall height of each stack and measured in bits, and on the other hand, the relative frequency of each AA, which is indicated by the height of the corresponding symbol. The sequence conservation at a particular position, $R_{seq}$, is defined as the difference between the maximum possible entropy and the entropy of the observed symbol distribution:

$$R_{seq} = S_{max} - S_{obs} = \log_2 N - (-\sum_{n=1}^{N} p_n \log_2 p_n)$$

Figure 4.1: Sequence conservation formula [2]

Here, as described in [2], $p_n$ is the observed frequency of symbol $n$ at a specific sequence position and $N$ is the number of different symbols for the sequence type. $N = 20$ and the maximum sequence conservation per stack is $\log_2 20 \approx 4.32 bits$ for protein sequences, because the alphabet for these sequences are the 20 naturally occuring AAs. Alternatively, sequence logos are also often used for Deoxyribonucleic Acid (DNA) and then the maximum sequence conservation per stack would be $\log_2 4 \approx 2 bits$.

Figure 4.2 shows an example of such a sequence logo for protein sequences. Here, the

length of the MSA is 20. As it can be seen in Figure 4.2, G is highly conserved at position 9 and occurs with a high frequency at this position. Furthermore, also T and R at positions 14 and 17, occur respectively, with a high frequency. On the other hand, at position 6, for example, the sequence conservation is relatively small. This means that at this position there are a lot of different almost equiprobable AAs in the MSA.

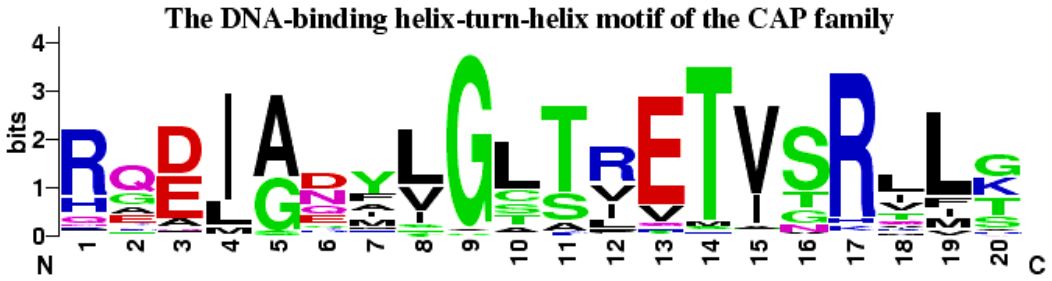


Figure 4.2: Sequence logo of the Catobolite Activator Protein (CAP) [1]

## 4.3 Conclusion

The development and implementation of the visual display was successful and meets the defined requirements. Also, the emission and transition probabilities were rendered in user friendly ways. The display is now ready to be integrated into the current version of the HMModeler by the research group of the Information Technology and Systems Management Masters program at Salzburg University of Applied Sciences.
This paper will also be used as a basis for writing a bachelor thesis about visualization methods for Profile HMMs.

# Bibliography


[1] Computational Genomics Research Group, University of California, Berkeley: *WebLogo: A sequence logo generator*. `http://weblogo.berkeley.edu/` (20.02.2011).

[2] Crooks, G.E., Hon, G., Chandonia, J.M., and Brenner, S.E.: *WebLogo: A sequence logo generator*. Genome Research, 14:1188–1190, 2004.

[3] Gamma, E., Helm, R., Johnson, R.E., and Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, Amsterdam, 1st edition, 1994.

[4] Graf, R.: *Multiples Sequenzalignment mit Hidden Markov Modellen*. Master's thesis, Salzburg University of Applied Sciences, 2010.

[5] Lutz, M.: *Python: kurz & gut*. O'Reilly Verlag, Köln, 3rd edition, 2005.

[6] Merkl, R. and Waack, S.: *Bioinformatik interaktiv: Grundlagen, Algorithmen, Anwendungen*. WILEY-VCH Verlag, Weinheim, 2nd edition, 2009.

[7] Model, M.L.: *Bioinformatics Programming Using Python*. O'Reilly Media, Sebastopol, 2009.

[8] Pettersen, E.F., Goddard, T.D., Huang, C.C., Couch, G.S., Greenblatt, D.M., Meng, E.C., and Ferrin, T.E.: *UCSF Chimera–a visualization system for exploratory research and analysis*. J Comput Chem, 25(13):1605–1612, Oct 2004.

[9] Resource for Biocomputing, Visualization, and Informatics, University of California, San Francisco: *UCSF Chimera Home Page*. `http://www.cgl.ucsf.edu/chimera` (13.02.2011).

[10] Umshaus, W.: *Improvement of the scoring schemes in hmm protein anaylsis: e-values, p-values and alternatives to viterby-scoring.* Marshall Plan Report, Salzburg University of Applied Sciences, 2010.

[11] Umshaus, W.: *Scoring schemes in hmm protein analysis: Forward algorithm and p-values.* Bachelor thesis, Salzburg University of Applied Sciences, 2010.

[12] Wegenkittl, S., Auer, F., Bindreither, D., and Lackner, P.: *Expert knowledge enhanced structure based profile hmms for protein sequence families.* Poster presentation at the 3DSIG 2009, 2009.

# List of Abbreviations

**AA** Amino Acid

**AAs** Amino Acids

**DNA** Deoxyribonucleic Acid

**DNAs** Deoxyribonucleic Acids

**GUI** Graphical User Interface

**HMM** Hidden Markov Model

**HMMs** Hidden Markov Models

**IDE** Integrated Development Environment

**MSA** Multiple Sequence Alignment

**MSAs** Multiple Sequence Alignments

**MVC** Model-View-Controller

**UCSF** University of California, San Francisco