Phylogenetic studies of the vesicular fusion machinery

Dissertation
for the award of the degree
"Doctor rerum naturalium" (Dr. rer. nat.)
Division of Mathematics and Natural Sciences
of the Georg-August-Universität Göttingen

submitted by
Nickias Kienle

from
Tübingen, Germany

Göttingen, 2010

Members of the Thesis Committee:

Dr. Dirk Fasshauer (1^{st} Reviewer)
Research Group Structural Biochemistry, Department of Neurobiology, Max Planck Institute for Biophysical Chemistry

Prof. Dr. Burkhard Morgenstern (2^{nd} Reviewer)
Department for Bioinformatics, Institute for Microbiology and Genetics, University of Göttingen

Prof. Dr. Nils Brose
Department of Molecular Neurobiology, Max Planck Institute of Experimental Medicine

Date of the oral examination:

Declaration of Authorship

Hereby, I confirm that I have created this work (*Phylogenetic studies of the vesicular fusion machinery*) entirely on my own and that I have only used the sources and materials cited.

Göttingen, 31st of May 2010

Nickias Kienle

In accordance with the standard scientific protocol, I will use the personal pronoun "we" to indicate the reader and the writer, or (as explained in Appendix A) my scientific collaborators and myself.

Acknowledgements

First of all, I would like to thank Dirk Fasshauer and Tobias Kloepper for letting me work on this project. Their advice and support led me along my path. I am very grateful for the opportunity and the nice atmosphere throughout the whole time.
Many thanks to Prof. Dr. Burkhard Morgenstern and Prof. Dr. Nils Brose for being members of my committee and for their advice.
Thanks a lot to Anand Radhakrishnan for the C2 domain introduction and my practical intern Annette Weizbauer for her help with the SNAPs.
Furthermore, I would like to thank Gottfried Mieskes for continuous organizational support.
Many thanks to the GGNB for financially supporting my method course at the Cold Spring Harbor Laboratories.
I would like to acknowledge Caro, Esra, Ioanna, Katrin, Sina, Ulrike, Alexander, Anand, Dennis, Matias, Nathan, Pawel, and Tobias who all started out as colleagues, but became friends. Thank you all a lot, it never would have been the same without you.
Zu guter Letzt möchte ich meinen Eltern, meiner Schwester, Luise und Erika (Ruhe in Frieden) danken. Ihr seid die beste Familie die man sich nur vorstellen kann.

Abstract

The eukaryotic cell consists of a large system of membrane delimited compartments. Material exchange between these compartments is mediated by intracellular trafficking vesicles. These vesicles bud from a donor compartment, travel along the cytoskeleton, are tethered, and finally fuse with the membrane of the target compartment. Several key protein families (e.g. SNARE, SNAP, Rab, SM, C2 domain proteins) that are involved in intracellular trafficking are highly conserved not only between species, but also between different trafficking steps. The precise molecular activity of the members of these families is often not well understood and little is currently known about the changes of individual factors during evolution. Unraveling the evolutionary history of these vesicular fusion proteins would fill the gaps and provide more insight into the molecular events. Hence, a technical basis is needed for data handling and to conduct necessary analyses. This thesis describes the development of the highly flexible and efficient *Tracey* management system (database, Java database package, and web interface). With this innovative system, it is possible to classify and analyze even extremely complex and versatile protein families. Consequently, the system was used to analyzed SNARE proteins in fungi, the SNAP family, and C2 domains in fungi.

Kurzfassung

Die eukaryotische Zelle ist in mehrere Kompartimente unterteilt, welche durch Membranen vom Rest der Zelle abgetrennt sind. Stoffaustausch zwischen Kompartimenten geschieht über vesikulären Transport. Vesikel werden am Donorkompartment abgeschnürt, wandern danach entlang des Zytoskeletts, um schliesslich mit der Membran des Zielkompartiments zu fusionieren. Verschiedene Proteinfamilien (z.B. SNARE, SNAP, Rab, SM, C2 Domänen Proteine), die im intrazellulären Transport eine entscheidende Rolle spielen, sind hoch konserviert. Dies gilt nicht nur für unterschiedliche Organismen, sondern auch für die unterschiedlichen Transportschritte innerhalb einer Zelle. Die genau Funktionsweise dieser Proteinfamilien ist häufig unklar und es existieren wenig Hinweise auf deren evolutionärer Entwicklung. Eine Untersuchung der Entstehung dieser Proteinfamilien könnte interessante Einblicke in die molekularen Ereignisse liefern. Zur Durchführung einer solchen Untersuchung bedarf es einiger technischer Voraussetzungen. Die vorliegende Arbeit beschreibt die Entwicklung des flexiblen und leistungsfähigen *Tracey* Verwaltungssystems (Datenbank, Java Datenbankpaket und Webseite). Dieses innovative System erlaubt die Klassifizierung und Analyse selbst von hoch komplexen und mannigfaltigen Proteinfamilien. Darüber hinaus wurde das System eingesetzt, um die SNARE Proteine in Pilzen, die SNAP Familie und die C2 Domänen in Pilzen zu untersuchen.

Contents

List of Figures

List of Tables

List of Abbreviations

aa	amino acid
AM	AccessManager
AMS	AccessManagerServer
AU	Approximately Unbiased
BLAST	Basic Local Alignment Search Tool
Doc2	Double C2 domain protein
ES	ExecutorService
EST	Expressed Sequence Tag
fUnc13	Fungi Uncoordinated Family Member 13
GSP	Groovy Server Pages
HMM	Hidden Markov Model
IQP	Important Quartet Puzzling
IQPNNI	Important Quartet Puzzling and Nearest Neighbor Interchange
JVM	Java Virtual Machine
LE	Log Expectation
Lgl	Lethal giant larvae
MLE	Maximum Likelihood Estimation
MP	Maximum Parsimony
MSA	Multiple Sequence Alignment
MUG190	Meiotically Up-regulated Gene 190 Protein
MUSCLE	Multiple Sequence Comparison by Log-Expectation
MVC	Model-View-Controller
NCBI	National Center for Biotechnology Informatio
NEDD4	Neural Precursor Cell Expressed, Developmentally Down-regulated 4
Npsn	Novel Plant SNARE
nr	non-redundant
NSF	N-ethylmaleimide-sensitive fusion protein
Pfam	Protein family database
PKC	Protein Kinase C
PLC	Phospholipase C
PSD2	Phosphatidylserine Decarboxylase 2
PSI-BLAST	Position-Specific Iterative BLAST

PSSM	Position-Specifc Scoring Matrix
PX	Phox homology
RefSeq	Reference Sequence
RP3	Rabphilin
SM	Sec1/Munc18
SMART	Simple Modular Architecture Research Tool
SNAP	Soluble NSF attachment protein
SNAP-25	Synaptosomal-associated Protein 25 kDa
SNARE	Soluble N-ethylmaleimide-sensitive factor attachment protein receptor
Syb	Synaptobrevin
Syp	Syntaxin of plants
Syt	Synaptotagmin
Syx	Syntaxin
TCB	Three Calcium and Lipid Binding Domains
TMR	Transmembrane region
UPGMA	Unweighted Pair Group Method with Arithmetic mean
WAM	WebAccessManager
WGD	Whole Genome Duplication
XML	Extensible Markup Language

1 Introduction

1.1 Intracellular membrane trafficking

The eukaryotic cell encompasses a large system of intracellular membrane delimited compartments. It is often assumed that this extensive endomembrane system evolved as a result of a phagotrophic lifestyle by invagination of the plasma membrane. For example the Endoplasmic Reticulum builds the starting point for proteins of the secretory pathway. It facilitates protein biogenesis and folding of newly synthesized proteins. In a subsequent step, these proteins get transported to the Golgi apparatus for further post-translational modification and processing. Additionally, the Golgi apparatus serves as a sorting hub for proteins to their target destination. In the endocytic pathway, extracellular material is taken up by the cell. During this process, endosomes and lysosomes/vacuoles are required for sorting and processing of newly internalized molecules.

Vesicles, small, intracellular, membrane-enclosed sacs, are utilized as carriers to mediate material exchange between different compartments. All vesicle transport reactions can be divided into four steps, vesicles bud from a donor organelle, move along the cytosceleton, are tethered, and then fuse with an acceptor organelle (see fig. 1.1-1). The protein machineries involved in these processes (e.g. Coat proteins [1], Rabs [2], soluble N-ethylmaleimide-sensitive factor attachment protein receptors (SNAREs) [3], Sec1/Munc18 (SM) proteins [4]) are highly conserved, not only among all eukaryotes, but

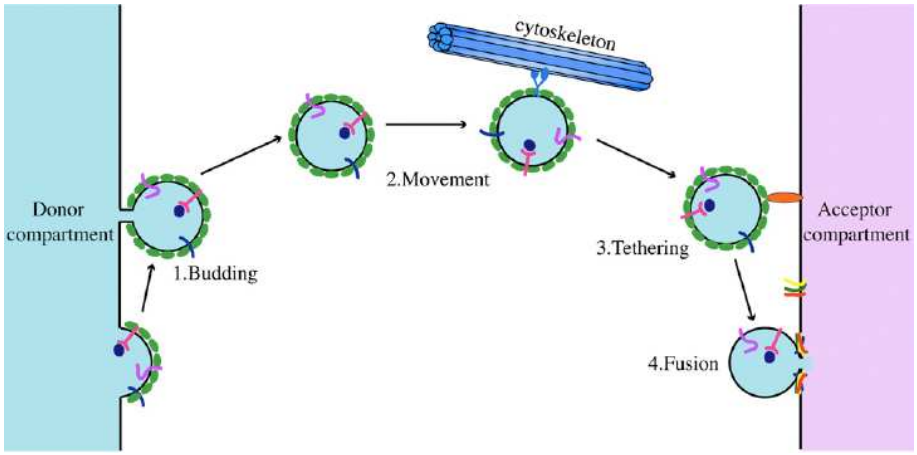

Figure 1.1-1: Distinct steps of an intracellular transport vesicle. In the first step (budding), a vesicle buds from a donor compartment. Subsequently, it moves along the cytosceleton towards its destination. Tethering factors bring the vesicle into place, so that the fusion process can begin. Modified from [10]

also between different vesicular trafficking steps [5, 6, 7] and probably originated by duplication and diversification events of prototypic protein machineries during evolution. This indicates that the proto-eukaryotic ancestor was already equipped with the various compartments found in present cells [8, 9, 10, 11].

An extensively well-studied example for intracellular trafficking is neuronal exocytosis. It takes place at the presynaptic membrane of a chemical synapse and serves interneuronal signaling. Upon Ca^{2+} influx, vesicles filled with neurotransmitters fuse with the presynaptic plasma membrane and secret their cargo into the synaptic cleft. Afterwards, the neurotransmitters diffuse to the receptors of the post-synaptic membrane of the neighboring neuron. Neuronal exocytosis can be divided into four different steps (tethering, docking, priming, and fusion). First, tethering factors capture free vesicles in order to concentrate them at their destined place of work. Second, docking proteins/complexes hold the vesicles in close proximity to the plasma membrane. In the third step, priming factors make arrangements and modifications, so that upon Ca^{2+} influx, fusion can be triggered nearly

instantaneously. In the final step, proteins of the core fusion machinery (e.g. SNAREs, SM) drive the process by merging the vesicle with the presynaptic plasma membrane. Neuronal exocytosis is a highly specialized process that involves various additional factors. Some of these factors contain C2 domains (e.g. Synaptotagmin (Syt), Rabphilin (RP3), double C2 domain protein (Doc2), Unc13) and seem to play an important role as Ca^{2+} sensors.

1.1.1 The core fusion machinery

In all intracellular trafficking steps, the central machinery involved in the fusion process is composed of members of the SNARE protein family (reviewed in detail in [3, 12]). It is thought that this machinery is tightly controlled by members of the SM family. SNAREs form a large family of cytoplasmic oriented membrane proteins, with most of them anchored by a C-terminal transmembrane region (TMR). SNARE proteins are associated either with the vesicle membrane or with the target membrane. The defining feature of the family is a unique motif, the so-called SNARE motif. It is a stretch of about 60-70 amino acids (aa), arranged in heptad repeats 1.1-3. During the fusion process, SNAREs assemble into a tight four-helix bundle complex between opposing membranes. This association is thought to occur in a zipper-like fashion from the N-terminus towards the membrane anchors at the C-terminus, a process that is thought to pull the opposing membranes into close proximity (see fig. 1.1-2). In its interior, the SNARE complex is hold together by 16 layers (-7 to $+8$, see fig. 1.1-3) of mostly hydrophobic residues. The complex-forming domains are highly conserved, not only between different species, but also between different vesicle trafficking steps. Initially, SNAREs were named v- and t-SNARE, reflecting their membrane association (v stands for vesicle membrane and t for target membrane). However, a more detailed analysis of the main structural features of the SNARE complex allowed for a finer classification of SNARE

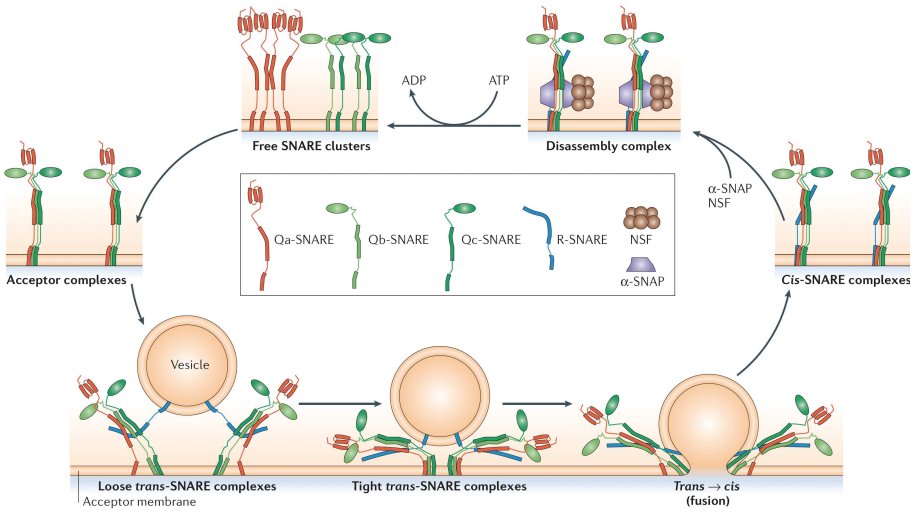

Figure 1.1-2: Initially, the three Q-SNARE motifs form the acceptor complex (Qabc). When the vesicle with the R-SNARE comes into close proximity, the formation of a four-helical trans-complex (SNAREs anchored on opposing membranes) is promoted. This zipper-like process starts at the N-terminnus and assembles towards the transmembrane anchors at the C-terminus. Finally, the fusion pore opens and the vesicle merges with the plasma membrane, transferring the SNAREs into a cis-complex-configuration (SNAREs anchored on the same membrane). To enable the SNAREs for consecutive rounds of fusion, they have to be disassembled again. This is managed by the AAA+ ATPase NSF together with its cofactor SNAP (soluble NSF attachment protein). (modified from [3])

proteins into four main groups (Qa, Qb, Qc, and R), reflecting their position in the four-helix bundle [13, 14, 15] (see fig. 1.1-3). In contrast to the mostly hydrophobic residues of the different layers in the SNARE complex, the 0-layer is special, as it contains hydrophilic residues. The reclassification of the SNAREs is based on the residues present at this position. While Q-SNAREs mostly contain a glutamine (1-letter code Q) at their 0-layer (see SNARE complex structure in fig. 1.1-3), R-SNAREs mostly contain an arginine (1-letter code R) at their central layer. SNAREs of different main groups vary in their N-terminal domains. Whereas most Q-SNAREs possess a three-helical Habc domain at their N-terminus, several R-SNAREs contain an N-terminal profilin/longin domain, and the members of the regulatory SNAREs contain two consecutive seven-bladed β-propeller domains

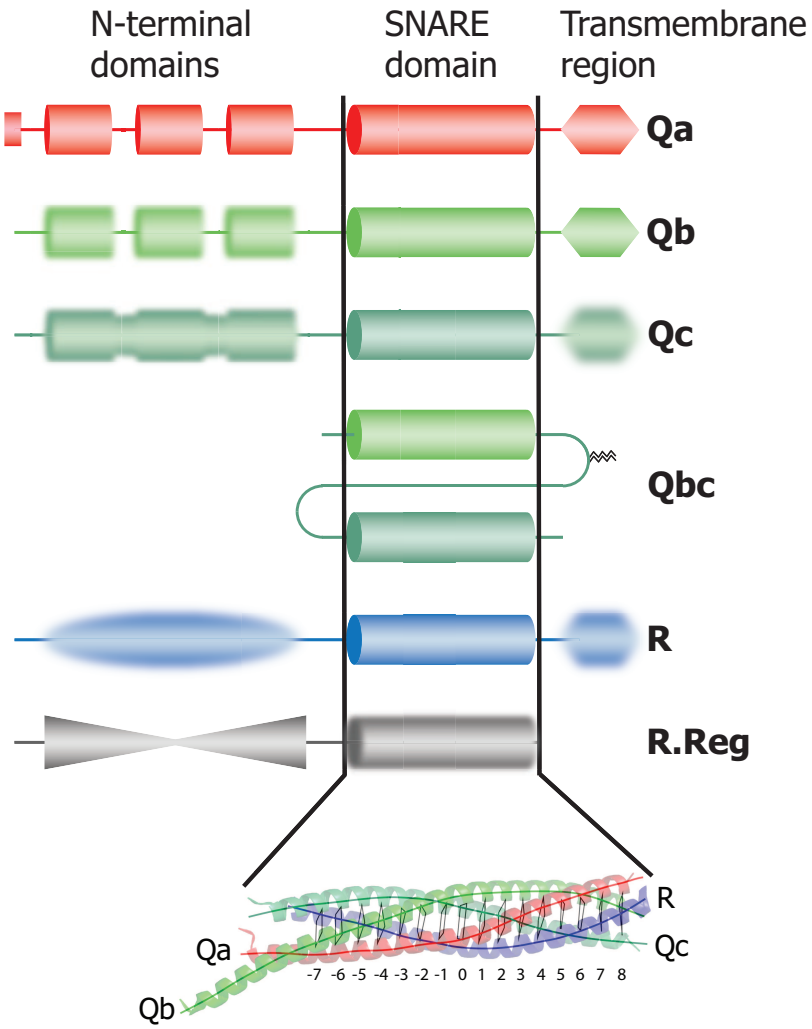

Figure 1.1-3: Domain compositions of Q/R SNAREs. Blurry domains cannot be found in all members of the specific subfamily. Qa SNAREs are known to possess a short N-terminal peptide and a three-helical domain (Habc domain). The same three-helical domain can also be found in several members of the Qb and Qc subfamily. A special Qc SNARE (Vam7) contains a N-terminal Phox homology (PX) domain, but lacks a transmembrane region (see section 3.3.1). Qbc members have an N-terminal Qb and an C-terminal Qc domain that are interconnected by a linker region. The linker often carries a cysteine stretch that is known to be palmitoylated (zig-zag lines) and serves as a membrane anchor [16]. Most R-SNAREs carry an N-terminal profilin/longin domain. Except Ykt6, all members of this subfamily possess a transmembrane anchor. Membrane association of Ykt6 is mediated by isoprenylation [17]. Regulatory SNAREs contain two consecutive seven-bladed β-propeller domains at their N-terminus. Some members of this subfamily lost their SNARE motif (see section 3.3.4). (adapted from [3])

(for details see fig. 1.1-3). Similar crystal structures from three different SNARE complexes [18, 19, 20] show that each main group contributes one SNARE motif to the formation of a four-helix bundle (QabcR-rule) [13]. In different intracellular trafficking steps, membrane fusion is mediated by specific combinations of SNAREs. However, it is still under intense debate in what manner SNAREs are to be assigned to different trafficking routes.

1.1.2 SNARE complex disassembly

After fusion of a vesicle with the plasma membrane, SNAREs are assembled and anchored on the same membrane (Cis-SNARE configuration, see fig. 1.1-1). For consecutive rounds of fusion, SNARE complexes need to be disassociated again, but spontaneous complex disassembly is very slow [21, 22]. Active SNARE disassembly is carried out by the AAA+ ATPase N-ethylmaleimide-sensitive fusion protein (NSF) together with its cofactor soluble NSF attachment protein (SNAP) [23]. Currently, it is assumed that at the beginning of disassembly process, the Cis-SNARE complex stands upright in the membrane. Since SNAREs do not possess a binding site for NSF, SNAP proteins (most likely three) envelope the complex and act as adaptors between the SNAREs and NSF [24, 25]. Subsequently, ring-shaped NSF hexamers hydrolyse ATP to disassemble the SNARE complex, beginning at the N-terminus [26, 27, 28, 29].

Higher eukaryotes have three isoforms of SNAPs: α-, β-, and γ-SNAP [25], whereas fungi only posses the single α-SNAP homolog Sec17. Unfortunately, it is entirely unclear how the different SNAPs are evolutionary related. Since most organisms contain only a single NSF, it is likely that SNAPs and NSF are able to disassemble all different SNARE complexes.

1.1.3 C2 domain proteins play an important role in neuronal exocytosis

As mentioned above, C2 domains can be found in various different factors involved in neuronal exocytosis. High conservation of sequence and structure suggests that all C2 domains belong to one family (see fig. 1.1-4). Originally, they were defined as the second conserved amino acid stretch in the α, β, and γ isoforms of Protein Kinase C (PKC) [30]. These members of the PKC family are involved in various signal transduction pathways, during which a stimulus (mechanical or chemical) is converted into a cellular response. They contain two consecutive C1 domains that serve as phorbol esters/diacylglycerol binding site, the aforementioned C2 domain, and a catalytic serine/threonine kinase domain. Several other C2 domain proteins play roles in signal transduction as well (e.g. Phospholipase C (PLC), cytosolic Phospholipase A2, Phosphatidylinositol 3-kinase). Interestingly, most of these factors possess only a single C2 domain. Various other proteins with C2 domains are involved in membrane trafficking (e.g. Syt, RP3, Doc2, Unc13). Often these proteins contain multiple C2 domains (e.g. Syt, RP3, and Doc2 possess two domains, Munc13 contains three C2 domains and Myoferlin, Dysferlin, and Otoferlin contain up to six C2 domains). Although it is not clear why multiple C2 domains, within one protein, emerged and how they function, it is quite imaginable that these domains might be able to function cooperatively. Many C2 domain proteins are cytosolic, but some factors (often related to intracellular trafficking) are membrane anchored by transmembrane regions (e.g. Syt, TCB).

A well-established function of C2 domains is Ca^{2+} dependent membrane lipid binding. C2 domains act like a switch, with increased lipid binding affinity upon Ca^{2+} binding. Higher intracellular Ca^{2+} concentrations can origin for example from opening Ca^{2+} channels in the synaptic endplate of a neuron or the Sarcoplasmic reticulum in muscle cells. Two different lipid

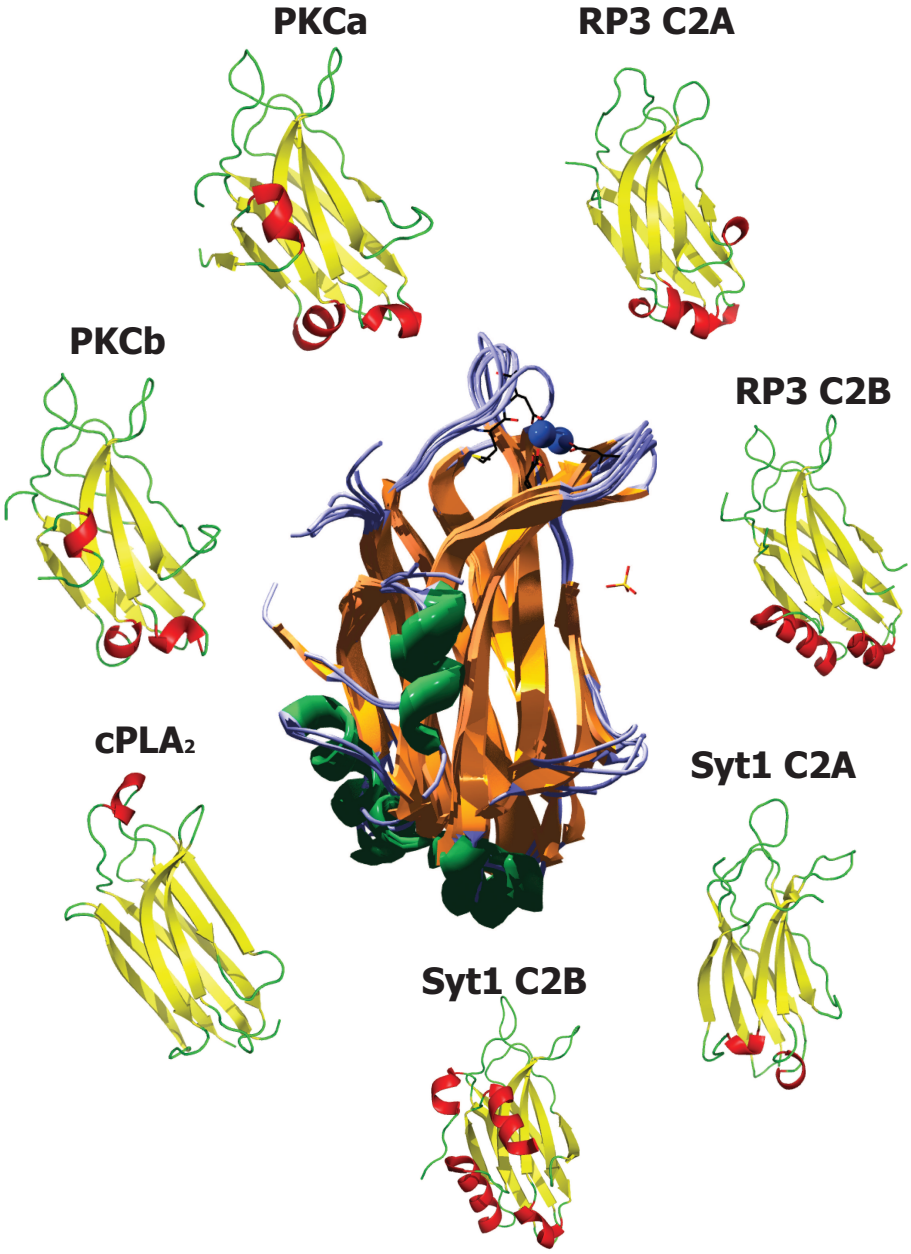

Figure 1.1-4: Peripherally shown are the C2 domain structures of Protein Kinase C α (PKCa), Protein Kinase C β (PKCb), cytosolic phospholipase A2 ($cPLA_2$), Synaptotagmin (Syt), and Rabphilin (RP3). Centrally shown is the overlay of the outer single structures. Two blue spheres at the top indicate two Ca^{2+} ions at the binding site.

binding modes can be distinguished. In the absence of Ca^{2+}, the regions around the Ca^{2+} binding loops exhibit a negative electrostatic potential. Upon binding of positively charged Ca^{2+} ions, these regions become either positive, which allows for binding of anionic membrane lipids (e.g. phosphatidylserine) or overall neutral, enabling binding of zwitterionic membrane lipids (e.g. phosphatidylcholine) [31]. Additionally, it has been shown that some C2 domains are involved in protein-protein interactions [32]. Still it seems that Ca^{2+} dependent membrane binding is the predominant function of C2 domains.

C2 domains have a variable length of about 130aa and they form an antiparallel eight-stranded β-sandwich with highly flexible loops on "top" and at the "bottom" (see structures on fig. 1.1-5). Two distinct topologies (termed type I and type II) can be observed in connecting the β-strands [33] (fig. 1.1-5). In type I, the red β-strand in figure 1.1-5 is at the N-terminus of the domain, whereas configuration of type II bears the β-strand at the C-terminus. The evolutionary background of this topology difference is completely unclear, but the question arises if this difference can be associated with a specific evolutionary event. It as postulated that type I C2 domains might have been originated by recombination of conserved terminal β-strands between neighboring type II domains in animal three calcium and lipid binding domains (TCB) [34]. However, the presented evidence and the line of argument does not prove this hypothesis without a doubt. It cannot be excluded that more than two different topologies exist or that the observed topology changes are the result of different events. To be able to shed more light into the evolutionary history of C2 domains, a more universal and profound analysis is necessary.

The original C2 domain function is not clear yet, but identifying the canonical C2 domain set of the eukaryotic ancestor might answer this question. As outlined before, C2 domains occur mostly as Ca^{2+} dependent membrane

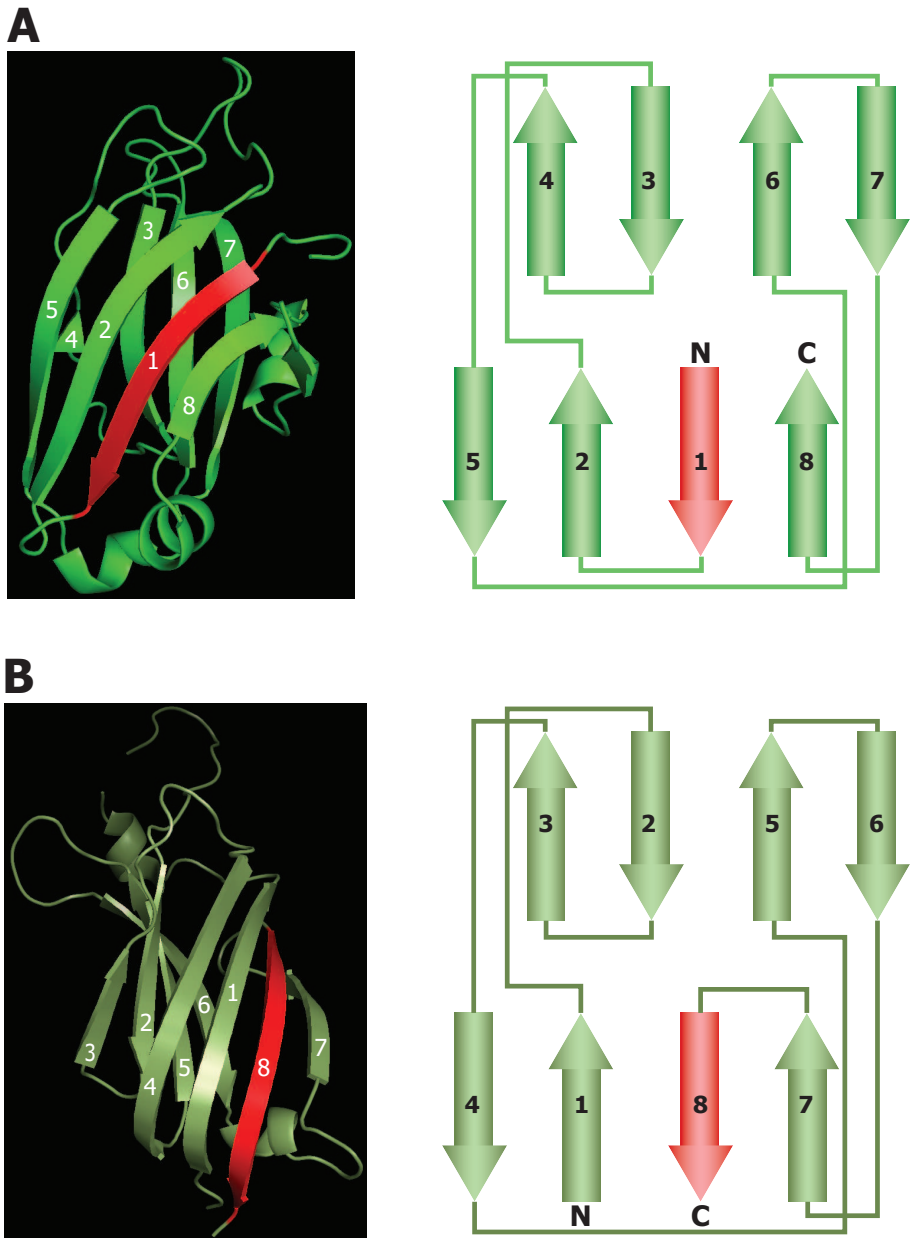

Figure 1.1-5: Two different structural topologies of C2 domains. Highlighted in red is the β-strand that is structurally equivalent in both topologies, but permuted in the primary sequence. (A) C2 type I topology with the red β-strand at the N-terminus. The structure shows the C2 domain of Protein Kinase C α (PKCa) from *Rattus norvegicus* (PDB code 1DSY). (B) C2 type II topology with the red β-strand at the C-terminus. The structure depicts the C2 domain of Protein Kinase C ϵ (PKCe) from *Rattus rattus* (PDB code 1GMY).

binding modules in various different proteins. This is often an auxiliary function that supports the protein in its actual role (e.g. kinase domain of PKC, catalytic domain of PLC). So far, it is unclear if a C2 domain of a specific protein can be used as a characteristic to identify this protein. Only by analyzing the evolutionary history of these domains throughly, it is possible to answer all these open questions.

1.2 Classification approach

As outlined above, several conserved protein families play important roles in vesicle trafficking. Little is currently known about the changes of individual factors during evolution and whether their functions are indeed conserved between fungi, plants, or animals. Moreover, several of these trafficking components only function in larger protein complexes. Hence, it is likely that individual factors of larger machineries co-evolved. Unravelling the evolutionary history of these vesicular fusion proteins would fill the gaps and provide more insight into variations of molecular events. To be able to accomplish this, a universal and thorough classification of the involved protein families is required. Several attempts based on standard bioinformatic approaches have been conducted to classify proteins involved in intracellular trafficking processes. However, these studies are limited either by the number of included species or by the number of included sequences [35, 6, 36, 37, 5, 13, 33, 15]. Additionally, such studies often use high throughput oriented methods, but speed at the cost of sensitivity can lead to false assumptions.

Several sophisticated and well established methods are available, which are able to detect many different domains (e.g. protein family database (Pfam) [38], simple modular architecture research tool (SMART) [39]). Such conventional approaches are usually based on only a few models and their main focus is to achieve a high degree of sensitivity (i.e. if a known domain is

present in a protein, it most likely will be detected). These methods usually work very well for the identification of domains, but existing models are rarely updated. Another limitation is that the specificity of the final result is often unknown. For example, SNARE motifs are often detected, but usually it is not possible to determine what kind of SNARE protein (e.g. Syntaxin, Synaptobrevin or Tomosyn) it might be. An additional problem of such methods is that their models usually are not generated to reflect the evolutionary development of a domain. This however is important to understand the evolutionary history of a domain.

To overcome the limitations of conventional approaches, our group developed a classification approach (see fig. 1.2-6) that aims to identify and distinguish all members of a given functional domain [7]. In the first step, a representative starting set of domain sequences is collected. Subsequently, an alignment of these sequences serves as the input for a basic phylogenetic reconstruction. The resulting hierarchy is then analyzed to define a set of groups and Hidden Markov Models (HMMs) are trained for each of these groups. In the next step, the models are utilized to gather more sequences by scanning different sources (e.g. sequence databases, genome projects). New sequences have to be verified to ensure only correct sequences are incorporated. The expanded set of sequences serves as the new basis for the refinement of the evolutionary conserved groups and models. This procedure is repeated iteratively until no new sequences can be identified or the quality of the models is sufficient.

Sequence alignments are the basis of the phylogenetic reconstruction and the HMMs. Hence, correct sequences and high quality alignments are very important elements in this classification approach. Incorporation of a verification process into the classification, in which newly gathered sequences are inspected by experts, ensures that only correct sequences are used in any further step. Although there are attempts to address redundancy and

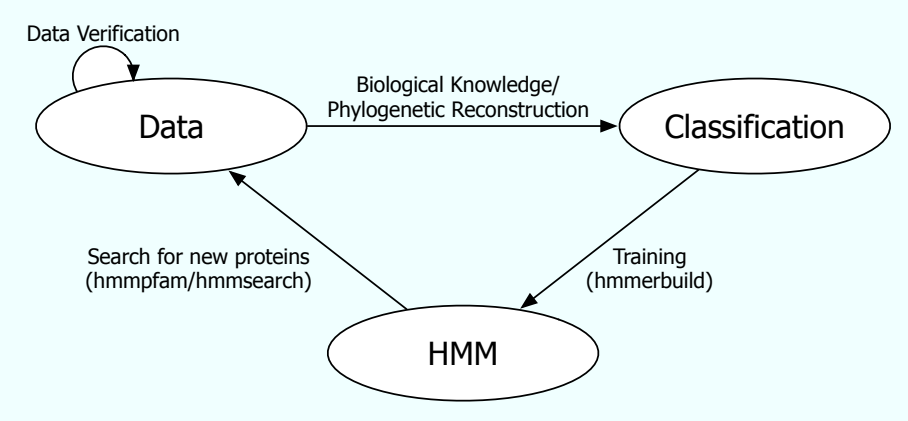

Figure 1.2-6: Schematic of the general classification system. The left circle (Data) is the starting point. Biological knowledge and phylogenetic reconstruction methods are applied to available data. This leads to a hierarchical classification for the underlying dataset. According to the resulting classification, Hidden Markov Models (HMMs) are trained for each distinct group. In the next step, these HMMs are used to scan several different sources (protein databases, genome projects) for new proteins. Newly predicted hits are not transferred directly into the working tables of the database, but go firstly into the verification tables. Entries in these tables have to be inspected by experts and then either deleted (wrong prediction or duplicate) or verified into the working tables.

accuracy in most common sequence databases, they still contain a variety of sequence variants (i.e. splice variants and sequences with low certainty). Collection of all these sequences without initial inspection, could negatively influence the balance and the quality of the underlying sequence dataset, but exactly this is mandatory to achieve high sensitivity in domain identification.

Amino acid sequences are prone to mutations and differences in the sequences reflect their evolutionary history. Homologous sequences can be divided into two different types, orthologs and paralogs. Orthologous sequences occur in different species, but originated for a common ancestor, whereas paralogous sequences originated by duplication in the same organism. Unraveling the evolutionary history of a domain by state-of-the-art phylogenetic reconstruction methods, allows for identification of orthologs and paralogs within a family of homologous domains. With this knowledge,

it is possible to generate a set of models that better reflect the evolutionary hierarchy of a domain family and this results in significantly improved specificity.
The developed classification approach, with a combination of machine learning methods, phylogenetic methods, and biological knowledge of functional domains, not only result in highly sensitive and specific predictors, but can also lead to the identification of unknown members within these domains and families.

1.3 *SNARE-Project* and Management System

The aim of the SNARE project was to establish a universal, hierarchical classification for SNARE proteins that reflects the specific evolutionary development of this family [7]. The previously introduced classification approach was utilized to achieve this goal (see section 1.2). Additionally, a strategy and a management system (database, Java database package, and web interface) was developed that ensured the efficiency of communication and knowledge transfer [40] (see fig. 1.3-7 and sections 1.3.2, 1.3.3, and 1.3.4). Biological knowledge and bioinformatic methods can be tightly combined to ensure high quality of the outcome.

1.3.1 SNARE classification

As introduced in section 1.1.1, SNARE proteins contain a SNARE motif with a length of about 60-70aa. However, to be able to use ungapped alignments, only the 53aa (layers -7 to $+8$) of the core SNARE motif were included into the classification. 150 well-known SNAREs (already classified into the main groups Qa, Qb, Qc, and R by Bock et al. [13]) served as a starting point for the analysis. Alignments of the motifs were used to train HMMs for each of the four SNARE main groups. Subsequently, a search of the non-redundant

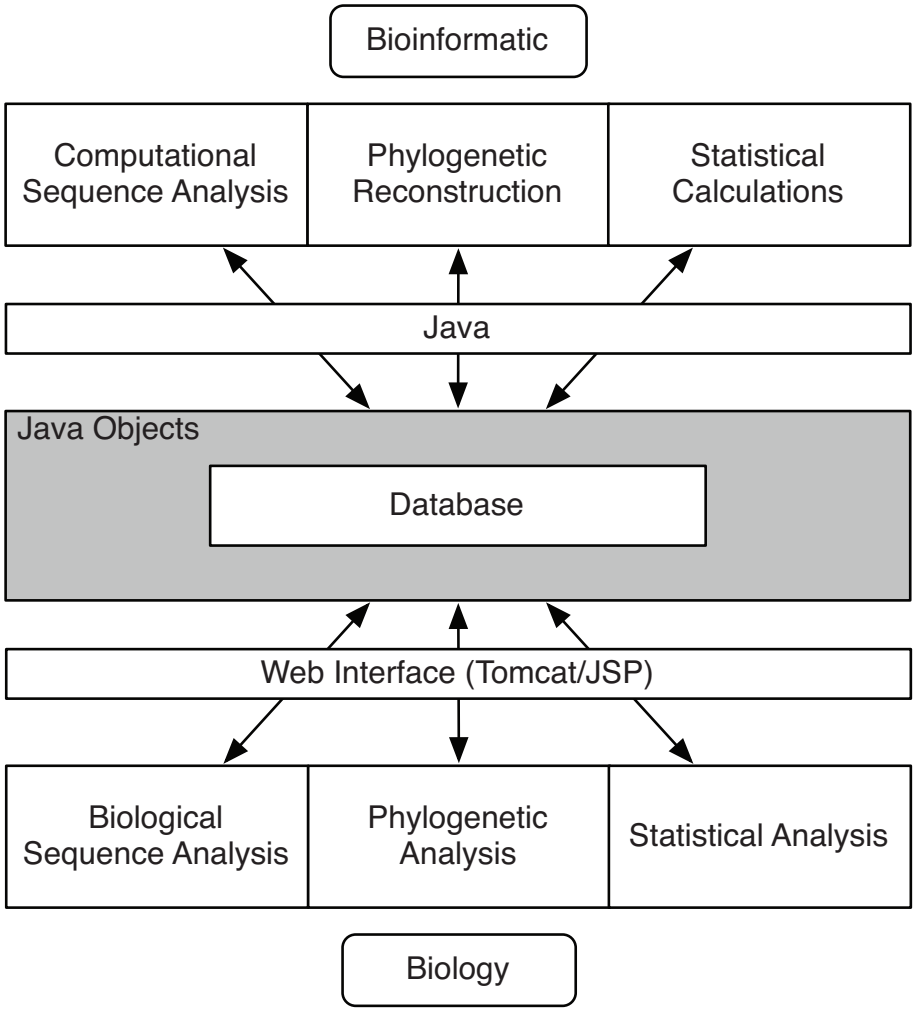

Figure 1.3-7: *SNARE-Project* interaction scheme between the bioinformatic (top) and the biology (bottom) side. All relevant information are stored in the database (white box in the middle). Each data table is wrapped by a belonging Java class (grey box) [41]. Stored information can be accessed either through direct usage of the Java wrapper classes or via a web interface , which also utilizes the respective classes.

(nr) database of the National Center for Biotechnology Information (NCBI, [42]) with the trained models resulted in about 800 SNARE proteins. Based on this dataset, a conducted phylogenetic reconstruction revealed 20 distinct conserved subgroups (see fig. 1.3-8). HMMs were trained for each of the 20 subgroups. After another round of searching in various sources (nr-, est-database, and several genome projects), followed by extensive sorting and selection, the final dataset comprised a total of 2165 SNAREs. The proteins were distributed over 154 different species, including 59 animals, 41 fungi, 18 plants, 25 protists, and two viruses. For about half of the species an almost complete SNARE set was present.

The 20 subgroups can be putatively assigned to different trafficking routes within the cell (see fig. 1.3-8). Five acceptor organelles can be assigned to basic intracellular transport: (I) the Endoplasmatic Reticulum, (II) the Golgi apparatus, (III.a) the trans Golgi network, (III.b) the endosomal compartments, and (IV) the plasma membrane. According to the QabcR-rule one SNARE of each main group needs to be present for the formation of a SNARE complex, therefore every distinct trafficking step requires a Qa, Qb, Qc, and R SNAREs from each main group. In the case of neuronal exocytosis these would be Syntaxin1a (Qa.IV), SNAP-25 (providing Qb.IV/SNAP.b, Qc.IV/SNAP.c), and Synaptobrevin2 (R.IV). Figure 1.3-8 depicts tentative assignments of SNAREs to the distinct intracellular trafficking steps. However, the way, in which SNAREs should be assigned to different trafficking routes, is still unclear. Additionally, some SNAREs are shown to have rather ambiguous interactions ([3, 44, 45]). Therefore, the SNARE groups in fig. 1.3-8 might represent the predominantly formed complexes, but participation of SNAREs in complexes of other fusion processes seems likely.

As mentioned before, the analysis revealed that the 20 subgroups are highly conserved within all species included. Several organisms seem to have an extended SNARE set (*Homo sapiens* 41, *Arabidopsis thaliana* 61). These

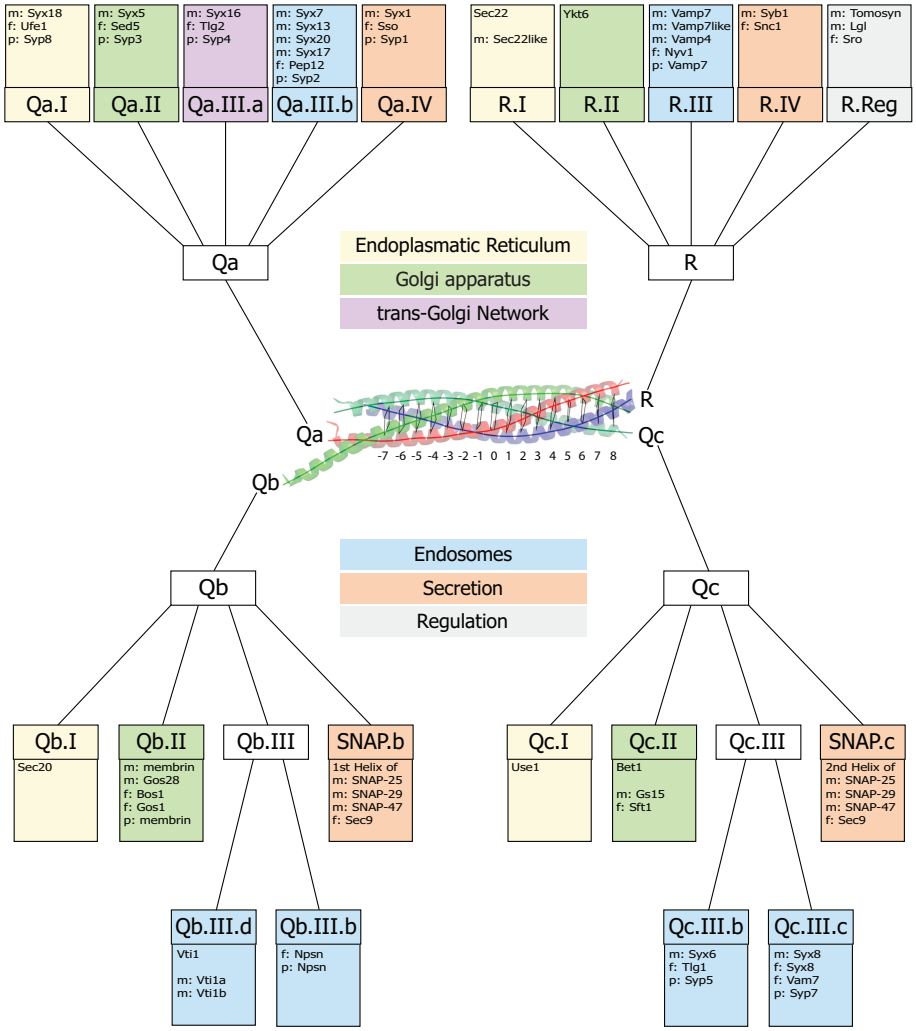

Figure 1.3-8: The four-helix bundle structure of the neuronal SNARE complex is shown as a ribbon diagram in the middle (blue, red, and green for synaptobrevin 2, syntaxin 1a, and Synaptosomal-associated protein 25a (SNAP-25a) respectively). The layers (-7 to $+8$) in the core of the bundle are indicated by virtual bonds between the corresponding Cα positions. SNARE proteins were divided into four main groups and 20 subgroups, using an extensive classication analysis. Genuine complexes are composed of four different SNARE motifs each belonging to one of the four main groups ('QabcR' composition). Putative SNARE units have been assigned to the basic transport steps. In addition to the fusogenic SNARE proteins, a regulatory R-SNARE without a membrane anchor, tomosyn, exists. The most commonly used names for the different SNARE types are given. For historical reasons, the names used for homologous SNAREs are often different in the different eukaryotic kingdoms. Different SNARE names used for metazoa (m), fungi (f), and plants (p) are listed. The names syntaxin and synaptobrevin (the secretory R-SNARE of metazoa that is also referred to as VAMP, vesicle-associated membrane protein) are abbreviated by Syx and Syb respectively. Several plant Q-SNAREs have been named syntaxin of plants (Syp). Moreover, several more special names of the markedly increased SNARE repertoire of vertebrates are not listed. [43, 7]

changes are usually the results of multiplication and diversification of the basic 20 subgroups. In the contrary, some lineages only comprise a very simple SNARE repertoire (fungi, green algae), containing only the basic 20 SNAREs. Hence, the 20 SNARE sub groups, shown in fig. 1.3-8, might be the original SNARE repertoire of an assumed proto-eukaryoitc ancestor.

1.3.2 SNARE database

To be able to store the information of the SNARE family, an appropriate database structure is necessary. The SNARE database [40] (see fig. 1.3-7) is build up by tables for the protein sequences, the motifs (*working tables*), the species, the groups, and the families. Moreover, tables exist for the protein sequences and motifs that still have to be verified (*verification tables*). Tables and table relations of the database are shown in fig. 1.3-9. The database is realized in the *MySQL* open source software [46].

1.3.3 SNARE Java database package

For each table in the database (see fig. 1.3-7) a software package with Java classes [41] was developed [40]. An instance of such a class represents one dataset (row) of the associated table. Each attribute of the tables are mirrored onto a member variable in the according Java class. Additional attributes simplify the handling of datasets. All classes contain a variety of member function that are able to alter/process the dataset and the database in various helpful ways. Every class implements the interface *dataset* (see fig. 1.3-10).

1.3.4 SNARE web interface

A web interface was developed that provides easy access to all stored information [40]. This web interface allows to search the database for specific data, to insert new data, and to verify predicted data. The web interface also

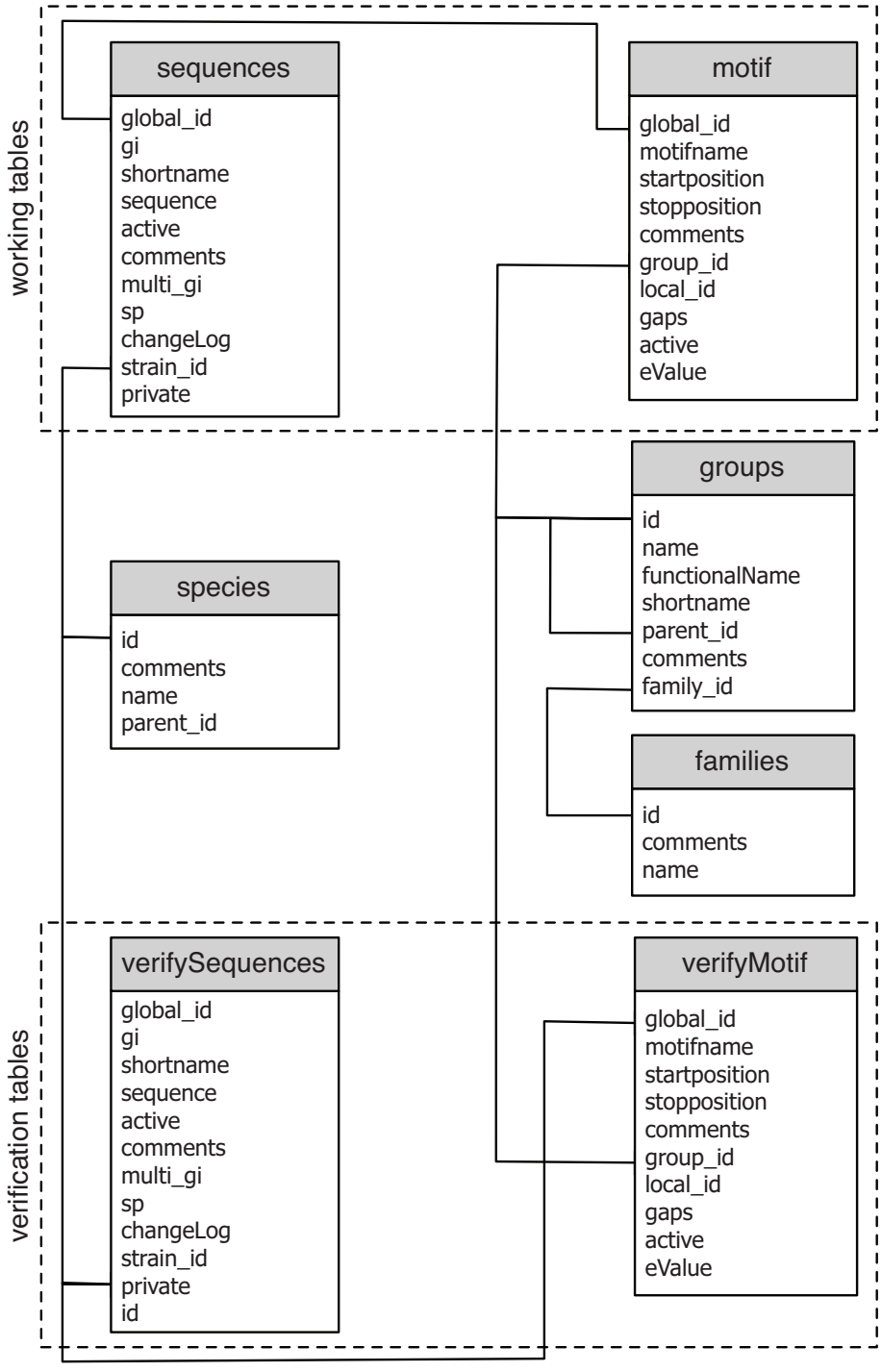

Figure 1.3-9: A schematic of the table relations of the *SNARE-Project* database. The tables sequences and motif build the so called working tables that contain the verified sequences/motif. The verification tables (verifySequences/verifyMotif) hold predicted hits, which were not inspected yet. Additionally, tables for families, groups (within families), and species complement the database.

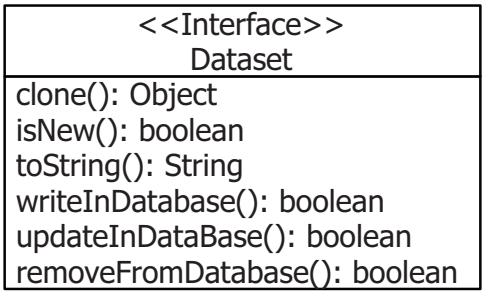

Figure 1.3-10: UML-like diagram of the interface *dataset*

implements a rights management system with three different ranks, to be able to grant different privileges to different user. For detailed information see section 1.3.2.

1.4 Aim of this Work

Previously, our group has analyzed the evolutionary history of the SNARE protein family. For this, a management system had been developed that included a database, Java database package, and a web interface. This study, which focused on the core membrane fusion machinery, was the starting point for analyzing the evolutionary history of the entire protein machinery involved in the docking and fusion of transport vesicles with the acceptor compartment. It turned out, however, that the management system designed for the SNARE protein family is not flexible enough to incorporate other protein families. The main aim of this study was to develop a new management system that would allow for incorporating various additional protein families. Since several protein families involved in vesicle trafficking contain various different domains and sometimes even multiple copies of the same domains, special attention had to be paid to manage proteins with complicated domain architecture. To indicate the domain composition of such proteins, a layout system needed to be developed. In addition, easier and more automatic data handling and data maintenance of the stored se-

quence data should be aimed for.

Furthermore, comprehensive tests of the novel data management system should be conducted by analyzing different additional protein families. These tests not only serve to find possible weak points of the management system but also to improve the pipelines for data handling. Based on the established classification of the SNARE protein family, the evolutionary history of the SNARE protein family in the fungi lineage should be investigated as a starting point. Since the genomic data available for fungi expanded drastically in the last few years, this would require to incorporate a large set of additional SNARE sequences from various fungi species. A novel protein family to incorporate into the new database were the SNAPs, which serve as cofactors for the SNARE disassembly ATPase NSF. According to the literature, the SNAP family could be expected to be rather small, usually only present in few copies in each eukaryotic genome. Therefore, this protein family was chosen to be analyzed in all eukaryotes in order to reconstruct its evolutionary history. In contrast, C2 domain proteins appear to be much more abundant, in particular in animals. This large protein family was chosen to be incorporated into the database and to eventually reconstruct its evolutionary history. In contrast to the SNARE and SNAP protein families this is a more challenging task, because this protein family is very diverse as the C2 domain can be contained as module for Ca^{2+} mediated membrane binding in different types of proteins. Owing to its complex modular architecture, C2 domain proteins would therefore serve as a touchstone for the novel data management system.

2 Material & Methods

2.1 MUSCLE

Alignment reconstruction tools are used in many different situations. For example, as an initial step of the classification and for constructing alignments for the phylogenetic analysis. Experience with different alignment construction tools (e.g. T-Coffee [47], PROBCONS [48], MAFFT [49]), manifested MUSCLE as the tool of choice. T-Coffee and PROBCONS calculate alignments with high accuracy, but high computational cost and high memory usage limits the number of sequences (<100) for both tools [50]. Additionally, the PROBCONS implementation was not very stable, this resulted in various program crashes. MAFFT and MUSCLE are comparable in terms of accuracy, speed, and memory usage [50]. However, MAFFT had considerable problems to construct alignments for a large number of very short sequences. MUSCLE proved to be very fast, accurate, and robust, therefore it was used as the standard tool.

MUSCLE is an abbreviation and stands for **MU**ltiple **S**equence **C**omparison by **L**og-**E**xpectation [51]. It is an example of an iterative alignment algorithm for constructing Multiple Sequence Alignments (MSAs). The algorithm of MUSCLE can be divided into three parts (see fig. 2.1-1):

- Initial Progressive Alignment
- Improvement of the Alignment

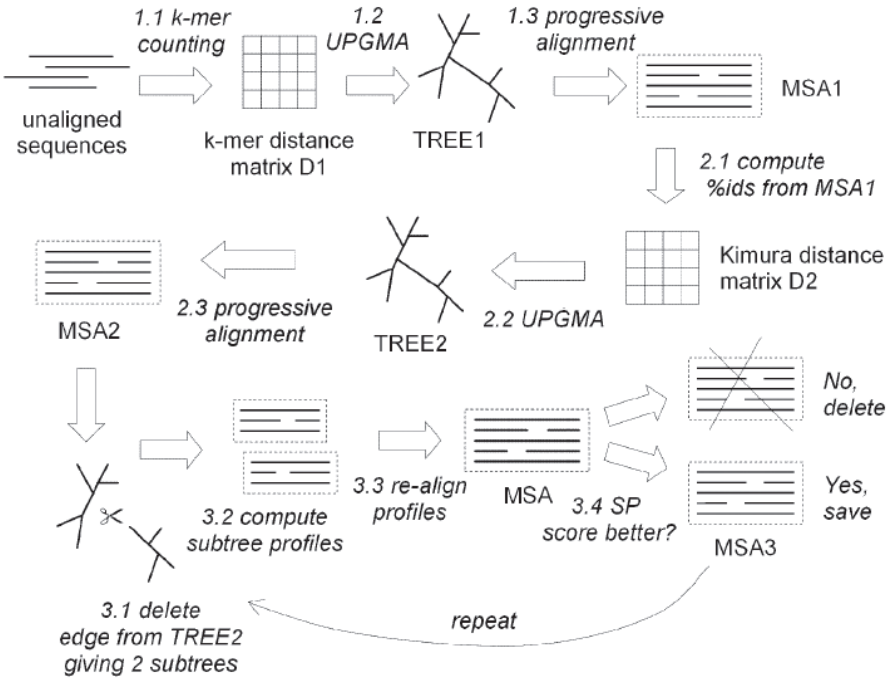

Figure 2.1-1: MUSCLE algorithm [51]

- Iterative Refinement of the Alignment

Initial Progressive Alignment

The focus during the initial alignment is speed and not accuracy. First, a distance matrix is build up with the $k-mer$ (word of length k) distances for each pair of input sequences. Assuming that two related sequences X,Y have a higher number of common $k-mers$ than two random sequences, the frequencies $n_X(\tau)$, $n_Y(\tau)$ of $k-mer$ τ in X, Y can be counted and together with their length L_X, L_Y a similarity measure between sequences X and Y can be defined as:

$$F_{X,Y} = \sum_{\tau} \frac{min[n_X(\tau), n_Y(\tau)]}{min[L_X, L_Y] - k + 1}$$

The k-mer distance measures are defined as:

$$d_{k-mer} = 1 - F_{X,Y}$$

The $k-mer$ distances (d_{k-mer}) are used to reconstruct a distance matrix, which in turn is used to reconstruct a guide tree. MUSCLE uses by default Unweighted Pair Group Method with Arithmetic mean (UPGMA) for the construction of the guide tree. One key advantage of UPGMA is its good runtime ($O(N^2)$) compared to other distance based reconstruction methods [51].

Another characteristic of MUSCLE is the profile scoring function for the pairwise alignment. MUSCLE introduces a new function called *log expectation* (LE) *score*:

$$LE^{xy} = (1 - f_G^x)(1 - f_G^y) \log(\sum_i \sum_j \frac{f_i^x f_j^y p_{ij}}{p_i p_j})$$

i and j are symbols from an alphabet (e.g. amino acids), f_i^x is the observed frequency of i in column x of the first profile and f_G^x the observed frequency of gaps in that column at position x. p_i is the background probability of i and p_{ij} the joint probability of i and j being aligned to each other, taken from the PAM-VTML-240 matrix [51, 52]. The factor $(1 - f_G^x)(1 - f_G^y)$ represents the occupancy of columns x, y and reduces the score for columns with a majority of gaps, leading to an significant improvement of the MSAs accuracy.

Combining the UPGMA guide tree with the LE score assures fast construction of an initial progressive alignment (see fig. 2.1-1 step $1.1 - 1.3$).

Improvement of the Alignment

Phase two utilizes the initially constructed MSA to calculate an improved distance matrix. Using the new matrix for the construction of a refined guide

tree, which in turn should lead to an improved MSA. Phase two is similar to phase one, with the improved distance matrix as the only difference (see fig. 2.1-1 steps $2.1 - 2.3$). MUSCLE uses the Kimura distance [53] to obtain the new distance matrix from the initial MSA. For two sequences with partial overlap D, the distance is defined as:

$$d_{Kimura} = -\log(1 - D - \frac{D^2}{5})$$

Iterative Refinement of the Alignment

Phase three is the iteration step. It can be divided into four parts (see fig. 2.1-1 steps $3.1 - 3.4$):

- Delete an edge to get two subtrees
- Compute subtree profiles
- Realign the profiles
- Compare the scores

Although, the resulting alignment of phase two is reasonably accurate, it is still biased towards the construction of the initial guide tree. In the first step of the third phase, two subtrees will be obtained by removing the edge with the greatest distance to the root of the guide tree. The profiles of the two subtrees are then realigned, this leads to a new MSA. Finally, the score of the new alignment is calculated and compared to the last best alignment obtained so far. If its score is higher, the new alignment will be used as the starting point in the next step, otherwise the last best one. This four steps will be repeated until the calculation converges or a user defined threshold is satisfied.

Usage

Default usage:

muscle *-in <inputfile> -out <ouptutfile>*

Usage for large alignments:

muscle *-maxiters 1000 -maxmb 2 -in <inputfile> -out <ouptutfile>*

2.2 HMMER

Profile HMMs can be utilized to model position-specific information about an MSA (e.g. the degree of conservation or the amino acid distribution of a column, possible inserts, and gaps) and to use this model to search for sequences or parts of sequences that show similar characteristics as the MSA. A profile HMM consists of an emission alphabet Σ of symbols (e.g. the 1-letter symbols for amino acids), a set of states π (π_1, π_2...), a matrix $P = \{p_{kl}\}$ of transition probabilities $p_{kl}, \forall\, k, l \in \pi$ (sums up to 1) and emission probabilities $e_k(b), \forall\, k \in \pi\, and \forall\, b \in \Sigma$ (sums up to 1) [54]. An MSA can be used to train a HMM, thereby training P and e with the given *training set*. The HMMER package is an implementation of profile HMMs for biological sequence analysis [55]. It provides programs for training a HMM (*hmmbulild*, *hmmcalibrate*) and for conducting sequence searches (*hmmsearch*, *hmmpfam*). The results of a HMM search depends on the sequences used during training. On the one hand side, if the chosen sequences are to similar, the method may not be able to find distantly related sequences (HMM is to specific). On the other hand, if the training set is to diverse, the chance of randomly associating sequences with the profile (false positives) increases significantly. HMMER offers a good tradeoff between sensitivity and specificity. For example, if a protein family contains three subgroups, HMMs can be trained for each of the subgroups and for the whole family. Such a

strategy provides options of high sensitivity (general family HMM) and high specificity (subgroup HMMs).

For example, HMMER can be much more sensitive and specific than classical sequence similarity search methods (Basic Local Alignment Search Tool (BLAST) [56] or FASTA [57]), since a flexible number of sequence can be used to train the HMM that builds the basis of the search. HMMER can also more flexible than Position-Specific Iterative BLAST (PSI-BLAST) [58]. PSI-BLAST runs an initial BLAST search for a sequence, using a standard substitution matrix (e.g. BLOSUM62). The resulting sequences are then used to construct a Position-Specifc Scoring Matrix (PSSM). In the next search iteration, this PSSM replaces the initial matrix, which can increase the search sensitivity. These last two steps can be repeated, until PSI-BLAST converges (no new sequence hit included in the last iteration). Crucial point of this method, is to choose the correct sequences for the construction of the PSSM. In this, PSI-BLAST and HMMER are quite similar, but the usage of HMMER is much more flexible and therefore much more practical. For example, the sequences for HMM training can be collected independently of the database a search is focusing on, whereas PSI-BLAST is directly dependent on the sequences in the database and thus the PSSM might be biased. Additional, a very practical feature of HMMER is the possibility to quickly check, whether a sequence belongs to a certain family. This can also be done with the PSI-BLAST, but it involves an indirect and intricate procedure.

2.2.1 Architecture

Fig. 2.2-2 shows an example for the the so called *Plan 7 profile HMM* of HMMER. The depicted architecture in the example represents a MSA with four columns, each is modeled by a match state (M_x). Every match state contains emission probabilities for emitting a character of the emis-

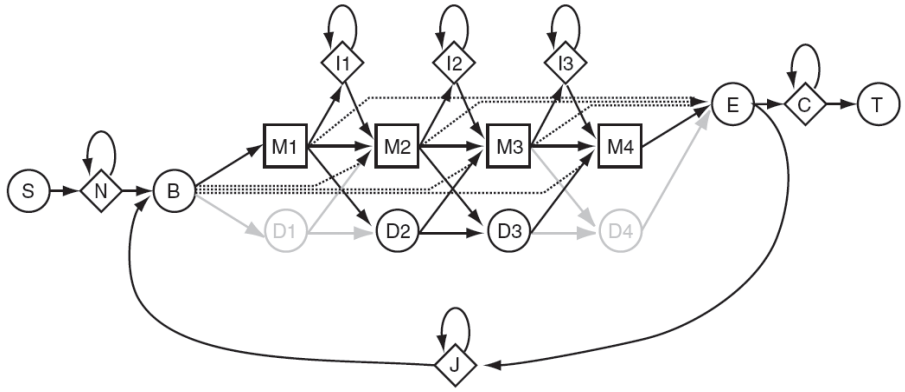

Figure 2.2-2: *Plan 7 profile HMM* architecture [55].

sion alphabet. In the case of protein sequence alignments, this would be 20 emission probabilities, one for each amino acid. Deletion states (D_x) are non-emitters and they model a region of gaps in an MSA. Possible insertions in an MSA are modeled by insertion states (I_x). These states contain emission probabilities that represent specific regions of inserts. HMMER calls the group $M/D/I$ at the same position a node [55]. The arrows in the architecture represent the transition probabilities to switch from one state into another. Except for $I \rightarrow D$ and $D \rightarrow I$, all transitions between $M/D/I$ are possible. Additionally, the architecture contains non-emitting states for entering (B) and exiting (E) the main model. $B/M/D/I/E$ builds the core of the *Plan 7 profile HMM* architecture.

$S/N/C/T/J$ are special states that are important for local or multi hit alignments. Initially, the model begins in the non-emitting start state (S). Regions of unaligned N-terminal/C-terminal sequence would be modeled by the N/C state. The J state represents possible joining segments of unaligned sequence between the hits. This comes in handy, if the the system is used to search for certain motifs in a sequence that could contain multiple copies of that specific motif. $N/J/C$ all emit on transition with specific emission probabilities. Finally, the non-emitting E state represents the exit of the model.

2.2.2 HMMER Programs

The HMMER package contains several different programs that are able to accomplish various task. Programs that are mainly used in this thesis are described in more detail, giving more insight into the application and some useful parameters.

hmmbuild

A program to build a HMM from an MSA.
Basic Usage:

hmmbuild [*-options*] <*hmmfile output*> <*alignment file*>

Options:

-n <s> : name; name this (first) HMM <s>
-o <f> : re-save annotated alignment to <f>
-A : append; append this HMM to <hmmfile>
-F : force; allow overwriting of <hmmfile>
–hand : this allows the specification of the model architecture by hand
–amino : force the sequence alignment to be interpreted as amino acids
–nucleic: force the sequence alignment to be interpreted as nucleic acids

Additionally, there are several options concerning weighting schemes for sequences (see table 2.2.1 [55]).

Principal usage for constrcuting HMMs:

hmmbuild *-F –hand –amino* <*hmmfile output*> <*alignment file*>

Option	Weighting Method	Reference
–wblosum	Henikoff simple filter weights	[59]
–wgsc	GSC tree weights (default)	[60]
–wme	maximum entropy (ME)	[61]
–wpb	Henikoff position-based weights	[62]
–wvoronoi	Sibbald/Argos Voronoi weights	[63]
–wnone	dont do any weighting	

Table 2.2.1: Different weighting schemes options of *hmmbuild*. [55]

hmmcalibrate

If a model is not calibrated, it uses an analytic upper bound calculation (extremely conservative) to determine the expectation value scores (E-value) for a search hit. By calibrating the model, *hmmcalibrate* empirically determines parameters for an HMM that make searches more sensitive. The calibration can take some time, but this eventually results in more accurate E-values and therefore higher sensitivity.
Usage:

hmmcalibrate <*hmmfile*>

hmmsearch

This program searches a sequence database for matches to a specific HMM.
Usage:

hmmsearch [*-options*] <*hmmfile*> <*sequence file/database*>

Options:

–cpu <n> : sets the number of CPUs to <n>
-E <x> : sets E value cutoff (globE) to ≤ x
-T <x> : sets T bit threshold (globT) to ≥ x
-Z <n> : sets Z (# seqs) for E-value calculation
-A <n> : limit of the alignment output is <n> best

domain alignments

Principal usage for sequence searches:

hmmsearch *-Z 100000 <hmmfile> <sequence file/database>*

hmmpfam

A program that searches an HMM database for matches to a query sequence. The query sequence can be one among many in a sequence database. All matches for the sequences are listed in the output.

Usage:

hmmpfam [*-options*] *<hmm database> <sequence file/database>*

Options:

–cpu <n> : sets the number of CPUs to <n>
-n : nucleic acid models/sequence (default protein)
-E <x> : sets E value cutoff (globE) to $\leq$ x
-T <x> : sets T bit threshold (globT) to $\geq$ x
-Z <n> : sets Z (# seqs) for E-value calculation
-A <n> : limit of the alignment output is <n> best domain alignments

Principal usage for sequence searches:

hmmpfam *-Z 100000 <hmm database> <sequence file/database>*

2.3 Phylogeny

Phylogeny is the reconstruction of evolutionary relations between different organisms, based on molecular data (DNA, RNA, or amino acid sequences). The most commonly-used methods can be divided into three different basic types: distance methods, maximum parsimony (MP), and maximum likelihood estimation (MLE).

For a given set of molecular data, distance based methods first compute a distance matrix and then try to find a tree that represents these distances as closely as possible. These methods are very fast, even for large datasets. However, their accuracy is heavily debated and they seem to perform not as good as methods from the other two categories. Additionally, quality estimation of such reconstructed trees can be quite time consuming.

Based on a statistical model for evolution, MLE methods try to find a phylogenetic tree that maximizes the likelihood of generating the given sequences at the leaves of the tree. MLE provides a systematic frame-work for explicitly incorporating assumptions and knowledge about the process that resulted in the given data. Any model of evolution is only a rough estimation of real biological evolution, but fortunately MLE is quite robust to violations of the model assumptions. MLE methods can statistically evaluate different tree topologies and use all available sequence information. On small data sets (about 20 sequences) MLE methods work excellent, but for larger datasets heuristics have to be utilized. This is due to the extensive tree space search, which is necessary to find the maximum likelihood.

MP methods take a set of aligned sequences and try to find a tree and a labeling of its internal nodes by auxiliary sequences with the intention of minimizing the number of mutations along the tree. In contrast to distance methods and MLE, MP methods are cladistics, as they try to reconstruct the pathways of evolution. MP methods do not provide information on branch length and are prone to long branch attraction artifacts. Long branch attraction is a phenomenon of phylogenetic reconstruction methods that infers two or more long branches as related, independent of their true evolutionary relation.

2.3.1 IQPNNI

Important Quartet Puzzling and Nearest Neighbor Interchange (IQPNNI) [64] is a very efficient MLE method for tree reconstruction and can be divided into four steps. In the initial step (1), the method calculates a tree using BIONJ [65] and then Nearest Neighbor Interchange is applied to this tree, until no further improvement of the likelihood function can be found [66]. To optimize the tree (2), first each leave with a probability below a certain threshold is deleted. These leaves are then reinserted by usage of Important Quartet Puzzling (IQP), followed by optimization using Nearest Neighbor Interchange. The resulting tree is tested (3) and, if the log-likelihood is better compared to the current best tree, the new tree is kept. The method stops if the number of optimization steps are above a pre-defined threshold (4), otherwise it returns to step (2).

2.3.2 Likelihood-Mapping

Likelihood-Mapping is a graphical method to visualize phylogenetic content of a set of aligned sequences [67]. Each quartet of sequences can infer three fully resolved tree topologies. Analysis of the maximum likelihoods of these topologies builds the basis of the method. A equilateral triangle with the three topologies as vertices represents the likelihoods as points inside (see fig. 2.3-3). Fig. 2.3-3 (A) shows a simplified model, in which the tree preferences are indicated by three basins. Data that represents real-world evolution is not as simple and sometimes it is not possible to resolve the phylogenetic relation of four sequences. Being not able to resolve this is either due to short sequences (noise) or a potential star-like evolution. Fig. 2.3-3 (B) depicts a model for which this possibility was included. With this model, it is possible to visualize phylogenetic content and it shows whether data are suitable for phylogenetic reconstructions. Additionally, the analysis is also able to look at specific clusters (disjoint groups of sequences), instead

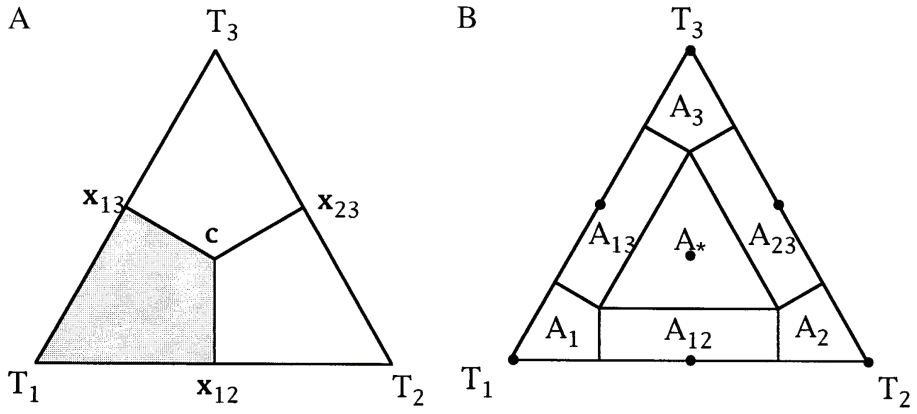

Figure 2.3-3: Equilateral triangles that visualizes the Likelihood-Mapping. (A) Simplified model with three basins. Point c shows equal preference for all three trees, whereas the x points show equal preference for two trees and none for the third. (B) Refined model for improved real-world evolution representation. A_* shows the area for star-like phylogeny. A_1, A_2, and A_3 show tendencies for tree-like regions. For the rest of the areas, the tendency for either topology is not clear. Modified from [67].

of looking at all quartets. Such an Likelihood-Mapping analysis can be used to exhibit support values of internal branches for a given tree topology.

2.3.3 Bootstrapping

Bootstrapping is a commonly used method for evaluating the reliability of an inferred phylogenetic tree. It starts by resampling the original MSA with replacement of the columns, this is analogous to cutting the MSA into individual columns and throwing them into a hat. Subsequently, columns are drawn from the hat and they become the columns of the new MSA. After a column is drawn, it is put back in the hat, the hat is shaken and another column is drawn. In the next step, a phylogenetic tree is inferred based on the new MSA. Resampling and tree reconstruction are repeated for a sufficient amount of times (between 100 and 1000 iterations). Branches of the original predicted tree are considered to be significant, if the same branches are also frequent (>85%) in the resampled trees. Bootstrapping can be applied by using the phylip package [68].

Bootstrapping is very useful in practice, but it is biased [69, 70, 71, 72]. The approximately unbiased (AU) test [73] aims on reducing the bias of the bootstrap probability. Changes in the sequence length lead to several sets of bootstrap replicates. To obtain the bootstrap probability values for different sequence lengths, the method counts the number of times the hypothesis is supported by the replicates for each set. Using the change in the bootstrap probability value along the changing sequence length, the AU test calculates the approximately unbiased probability value. The test provides an unbiased procedure for assessing the confidence of the inner edges of a tree.

2.3.4 Calculation & evaluation of phylogenetic trees

Each phylogenetic reconstruction started with the calculation of a IQPNNI tree. The Jones, Taylor, and Thornton matrix [74] was used as a distance matrix and a gamma distribution with four categories (accounts for rate-heterogenity across sites) served as the model of evolution. The stopping rule of IQPNNI was used, but the reconstruction had to run at least the suggested number of iterations. Likelihood-Mapping [67] was applied to each edge of the tree to estimate the accuracy of the topology. As a second source of confidence, the bootstrap analysis of the phylip package [68] was applied with a 1000 replicates. Standard settings for seqboot, again the Jones, Taylor, and Thornton distance matrix and a gamma distribution (with parameter approximation from tree-puzzle) for protdist, and standard options for neighbour were used. A random seed was set to nine if required. To correct for the systematically biased bootstrap values, consel [75] was used to perform the AU test [73]. A modified version phyml calculated [66] the site-wise log-likelihoods needed for the AU test. The initially obtained IQPNNI tree served as a starting point to join the results of the confidence calculation and its inner edges were labeled with Likelihood-Mapping and the corrected bootstrap values.

2.4 SNARE web interface

With the simplification of data handling in mind, a web interface was developed [40] for the *SNARE-Project* (see section 1.3). This web interface provides easy access to all information in the database (see fig. 1.3-7). The dynamic content of the website is realized in JavaServer Pages [41]. JavaServer Pages are a Java based technology and therefore allows usage of the software package, introduced in section (1.3.3). The website implements a rights management system with three different ranks. The first rank is "user", with this rank it is only possible to look at the protein sequences. No changes of existing protein sequences or entries of new protein sequences can be made. The second rank is "submit". In addition to the first rank, users with "submit" rights can change protein sequences or insert new protein sequences into the database, but each change or insertion needs to be verified by a user with the highest rank, "verify". Only these users have the rights to verify changed or inserted protein sequences. This ranking system is very important, as it ensures data integrity. Currently, the actual website is for internal use only, but a public version of the website (without submitting and verification possibilities) is located at `http://bioinformatics.mpibpc.mpg.de/snare/`.

To obtain an overview of the already known and classified protein sequences, the website provides the possibility to search the database with different attributes. Search attributes are the short name of a protein sequence, the species of a protein sequence, the group of a motif that belongs to a protein sequence and whether a protein sequence is active or inactive. Two ways can be used for the search for protein sequences by their short names, species or groups. Firstly, by typing the exact short name of the protein sequence into a edit field. The result is just this protein sequence. Secondly, a wild card (%) can be used to do a fuzzy search in the database. Each protein sequence, for which the parameter is part of the short name, is re-

Shortname-parameter: *ArTh%;%ho%*
Group-parameter: *Qb.I;R%*
Result: All datasets, which have "ArTh" at the beginning or "ho" somewhere in the short name
and have motifs with the groupname "Qb.I"
or with a groupname that begins with "R"

Shortname-parameter: *DaRe_Syx8*
Species-parameter: *Homo sapiens*
Result: There will be no result, because no dataset exists that has a short name that includes "DaRe",
and a species that is "Homo sapiens"

Figure 2.4-4: Example for a query at the SNARE website

turned. If the edit field contains several search parameters, they must be separated (";"). Additionally, boxes with all available species and groups are presented. Multiple entries of these boxes can be chosen and they are included into the search criteria. Several search parameters for one attribute are logically-or connected, attributes themselves are logically-and connected (see example 2.4-4).

Sometimes it is necessary to modify existing protein sequence entries if new information is available. The website provides functionality the for altering of a dataset and it distinguishes between two cases. Firstly, if the user who changed the data has "verify" rights and the attribute "sequence" was not changed, the protein sequence is simply updated in the *sequences* table of the database. Secondly, if the attribute "sequence" was changed or the dataset was changed by an user with "submit" rights or if an user with "verify" rights marks the dataset to be veried, the protein sequence is transferred to the *verifySequences* table. In addition, a search for conserved motifs of the protein sequence is conducted (*hmmsearch*, 2.2.2). The program searches through the amino acid chain according to the trained HMMs and writes the results in a file. Afterwards, the motifs are extracted and saved into the *verifyMotif* table. A connection between the altered protein sequence entry

in the *verifySequences* table and the new motifs in the *verifyMotif* table is established.

Insertions of new protein sequence entries into the database is restricted to user with "submit" or "verify" rights. All fields on the insertion mask can be editable freely, except for the species field. In this case, a drop-down box contains all the species that are stored in the database. If the species of the protein sequence is not present in the database, it has to be added first. The new protein sequence is first inserted into the *verifySequence* table. Additionally a search (*hmmsearch*, 2.2.2) for the motifs is conducted. Resulting motifs are inserted into the *verifyMotif* table. Like in the procedure after changing a protein sequence a connection between the new protein sequence in the *verifySequences* table and the new motifs in the *verifyMotif* table is established.

Verification of newly predicted protein sequences can also be carried out via the website. This part is only accessible to user with "verify" rights. The verification search mask provides the same search options as already described before. Several helpful tools are offered to a user that verifies protein sequences. Predictions for a sequence can be accepted, which transfers the sequences, along with its motif(s) into the *working tables*. By choosing the option "Keep in verification table" changes can be made, but the protein sequence and its motifs remain in the *verication tables*. Additionally, the website offers the possibility to recalculate the motifs for a specific protein sequence. It is in inevitable to recalculate motifs, if a earlier conducted prediction was incorrect or if the amino acid chain of a protein had to be altered. Furthermore, occasional updates of the HMM models might lead to the necessity of motif recalculation. In all these case the old motifs are deleted and the new motifs are inserted into the *verifyMotif* table and linked to the actual protein sequence. Moreover, possible *false positives* (predicted motifs that are incorrect hits), together with its motifs, can be deleted on

the website.

Predicted motifs, together with all relevant information, are presented apart from the actual sequence. One of the most important reference for the quality of a motif prediction by HMM methods is the so called E-value. It is a significance criteria for the reliability of a predicted motif and it represents the probability for a motif to occur in the amino acid chain by chance [54]. Along with the motif sequence, the HMM consensus and the similarity are presented. The amino acids of the HMM consensus string are the ones with the highest probability according to the HMM used. Capital letters stand for highly conserved residues. Additionally, insert errors in the HMM consensus are highlighted by dots (”.”). Similarity shows the correlation between the HMM consensus and the actual found motif. Letters (1-Letter code for amino acids) indicate exact matches, whereas not exact, but conserved matches are shown by a ”+” [54]. Additional options can be utilized to further simplify the decision finding process for the verifying user. Sometimes it can be helpful to be able to check similar, already verified motifs. Therefore a link for each motif exists that displays all verified motifs that belong to the same group and occur in the same species. It is possible to align the actual motif with one of the already verified motifs, to compare the similarity. This might not be sufficient, if a species is completely new. In this case, no verified motifs are present in the database. Therefore, it can be helpful to be able to check homologous species. The website also provides a link to compare verified motifs from homologous species that belong to the same group. After evaluating all information on the predicted motifs, a verifying user has to mark the correct ones for transfer to the *working tables* and the wrong ones for deletion. This can be done for every motif via radio buttons.

Furthermore, the websites provides sections for insertion, view, and change of species and groups.

3 Results & Discussion

3.1 Aligning sequences

Sequence alignments play a crucial role in the classification of domains. They serve as the basis of the phylogenetic reconstructions, as well as training sets for HMMs. Therefore, the quality of sequence alignments is of the utmost importance. Of all alignment construction tools tested, MUSCLE [51, 52] showed the best performance, especially in the case of large datasets. Unfortunately, even the heuristic approach of MUSCLE (see section 2.1) seemed to be error-prone in some cases (see fig. 3.1-1 A). Hence, a strategy was developed that aims on reducing the complexity of sequence alignment construction, by combining different methods. HMM predictions (motifs) are used to align conserved regions (see section 3.1.1). If the HMM is of high quality, consequently the prediction is of high quality and the conserved regions are well aligned. Motifs can contain not well conserved inserts that do not fit to the conserved regions represented by the HMM. These inserts can additionally be aligned by MUSCLE. The alignments are further refined, by applying an iterative block strategy that scans for conserved blocks and realigns the regions in between (see section 3.1.2). Additionally, a conservation filter was developed to ensure that all alignments only contain significant information (see section 3.1.3).

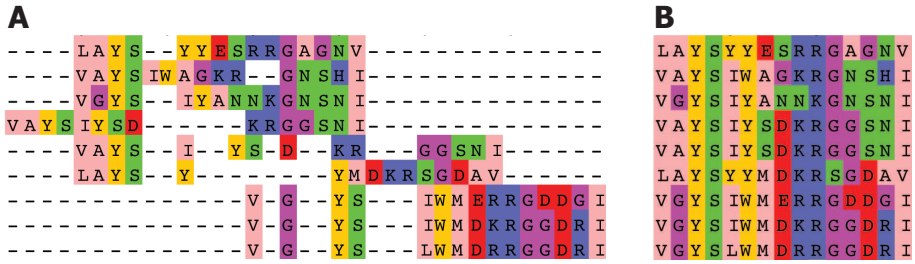

Figure 3.1-1: Small example of (A) mis-aligned sequences and (B) well-aligned sequences after refinement.

3.1.1 Motif Aligner

At the beginning of a classification, initial alignments are constructed using MUSCLE, followed by manual curation according to secondary structure elements, available 3D structures and TMR predictions. Afterwards, these high quality alignments serve as training sets for the first HMMs (see hmmbuild and hmmcalibrate in section 2.2.2). Subsequently, these models can be used to gather sequences with homologous domains form various sources (see hmmsearch and hmmpfam in section 2.2.2). All hits come with the HMM consensus string that indicates conserved positions (including gaps) and possible insert errors (see section 2.4). Since all domains predicted by the same specific model must have the same number of conserved positions, the consensus can be used to align these domains. An algorithm was implemented that goes through all consensus strings and aligns the residues at the conserved positions of the domains. Insert errors are independent of the HMM (various length) and therefore have to be aligned with a alignment construction tool (see fig. 3.1-2). Combining HMM motif predictions for conserved regions, together with MUSCLE for not well conserved insertions enhances the quality of the alignments significantly and thus improves the classification.

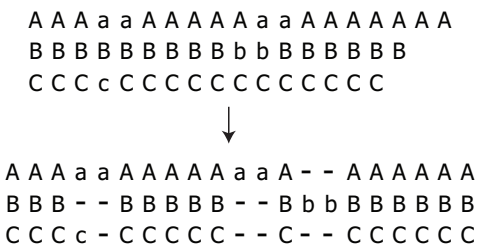

Figure 3.1-2: Example for an alignment, using the HMM consensus string. On top are the unaligned sequences. Capital letters show conserved positions, whereas small letters indicate insert errors. The aligned sequences at the bottom show blocks of conserved residues with blocks of aligned insert errors in between.

3.1.2 Alignment Refiner

Unfortunately, even the motif aligner approach sometimes results in mis-aligned sequences. The HMM package (hmmer, see section 2.2) applies the so called *Plan 7 profile HMM* architecture to construct the HMMs (see section 2.2.1). This architecture does not allow transitions from an insert to a deletion state and vice versa. If a training set alignment would result in such a transition, the programs of the HMM package adjust the alignment to correct for this before the HMMs are constructed. This policy could be one cause for mis-aligned sequences, when using the HMM consensus.
To improve the alignments further, an algorithm was developed that refines a given alignment using an iterative block strategy. It searches for conserved blocks within an alignment and aligns unconserved blocks in between, using alignment construction tools. For newly aligned unconserved blocks, it recursively refines them further, applying the same strategy. This approach significantly improves the alignment quality (see fig. 3.1-1 B) and therefore the quality of the classification.

3.1.3 Conservation Filter

As mentioned before, sequence alignments are the basis of phylogenetic reconstructions and HMMs. To ensure the quality of the results, it is impor-

tant to filter out alignment columns with insufficient information content. The information content of a column can simply be defined by the number of gaps (a column with 500 residues has a higher information content than a column with 200 residues and 300 gaps), but also by entropy [76]. Additionally, sequences can be incomplete (e.g. incomplete Expressed Sequence Tag (EST) sequences or misassembled sequences). Such sequences might possess only little information content and need to be filtered out as well. To address both issues, an algorithm was developed that traverses an alignment and removes columns with information content below a specific threshold and rows below a specific number of amino acids. This ensures that only the most significant information of an alignment are used in any further steps.

3.2 *Tracey*

During the work with the SNARE proteins it became clear that the *SNARE-Project* management system (database, Java database package and web interface, see section 1.3) was not flexible enough for the further analyses. Core principles of the system remained unchanged, but established components were adapted and complemented with novel additions (see fig. 3.2-3). The subsequent sections give an overview about innovations made to the management system and how these improve data handling. A major aim of the project was to shed more light into the evolutionary history of the protein families that are part of the intracellular vesicle fusion machinery. This might also provide the possibility to infer conclusions on the functionality of specific factors. Therefore, the novel management system was named *Tracey*, adopted from in**trace**llular vesicle fusion machiner**y**. Major objectives of this new system are to be generic, to be flexible, and to yield high-performance.

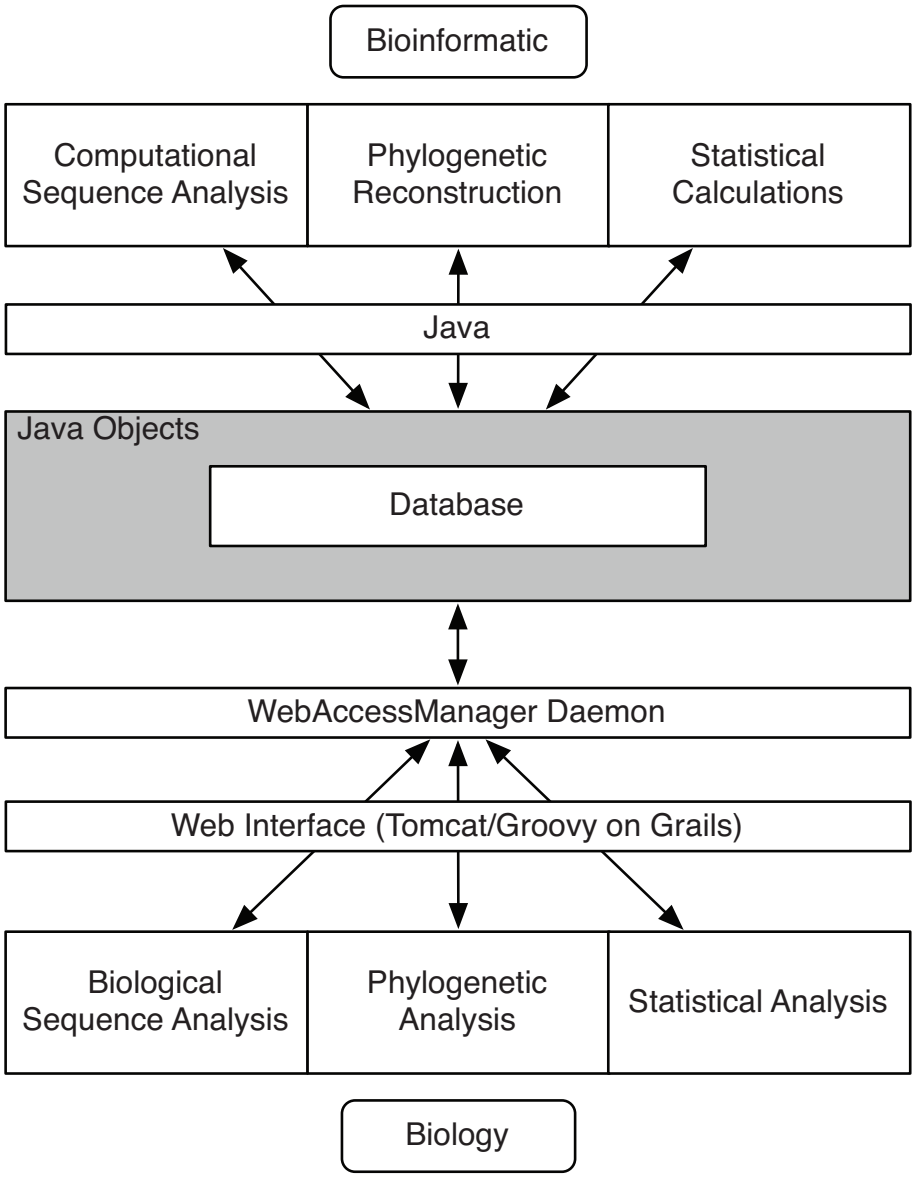

Figure 3.2-3: Novel interaction scheme between the bioinformatic (top) and the biology (bottom). All relevant information are stored in the database (white box in the middle). Each data table is wrapped by a belonging Java class (grey box) [41]. Stored information can be accessed either through direct usage of the Java wrapper classes or via a web interface. Contrary to the *SNARE-Project*, this novel interaction scheme contains an additional layer that mediates data exchange between the web interface and the database.

3.2.1 *Tracey* database

Although the *SNARE-Project* database (see section 1.3.2) served as the basis for the novel *Tracey* database, only little remained unchanged. The tables *sequences*, *motifs* (*motif*), and *verifyMotifs* (*verifyMotif*) were already present in the *SNARE-Project* database, but their configuration was modified fundamentally. Additionally, *species*, *groups*, and *families* also underwent an extensive redesign and were renamed to *taxonomy*, *domainGroups*, and *domains*. The table *verifySequences* was removed from the database. Figure 3.2-4 shows a schematic overview of the *Tracey* database and the relations between tables. Some of the NCBI databases are used as a reference for tables in this database. The database is embedded in the *MySQL* open source software (version 5.1.34) [46].

3.2.1.1 *sequences & genes*

The table *sequences* builds the core of the database. Fig. 3.2-5 (A) shows the column names and the associated column types. Each sequence is stored in this table. Its primary key is "sequence_id", which is an unsigned integer value, unique for every dataset. All protein sequence sources provide their sequences with a short description (usually as the FastA header). These information can be stored in the field "foreignAnnotation". A "shortName" can be given to every sequence in the table. These short names are usually build up by the "taxonomyShortname" of the organism (see section 3.2.1.2), followed by an underscore and the abbreviation of the protein name. For example, the SNARE Synaptobrevin 1 from *Homo sapiens* has the short name HoSa_Syb1. The field "annotation" is reserved for custom annotations. "sequence" can hold either the 1-letter code of an amino acid chain, the 1-letter code of an RNA chain, or the 1-letter code of an DNA chain. It can be very helpful to be able to rank sequences with different statuses. Values for these are in style of NCBI and can be "live" (alive and ready to use), "crystal

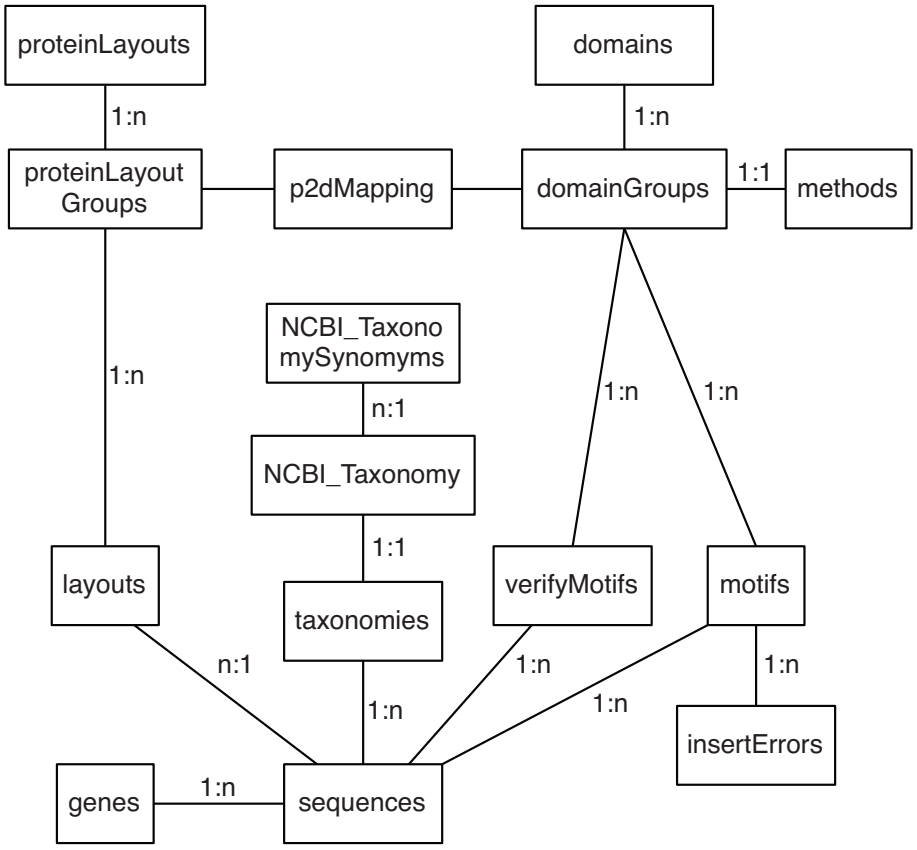

Figure 3.2-4: *Tracey* database scheme. The central table of the database is *sequences*. Every other table is directly or indirectly linked to this one. The table *genes* helps to keep track of all sequences belonging to the same gene. All necessary information about organisms can be stored in tables *taxonomies*, *NCBI_Taxonomy*, and *NCBI_TaxonomySynonyms*. Predicted motifs are stored in *motifs*/*verifMotifs*, with links to *insertErrors*, *domainGroups*, *domains*, and *methods*. Layouts are hold in the *layouts* table with links to *proteinLayoutGroups*, and *proteinLayouts*. Relations between *domainGroups* and *proteinLayoutGroups* are realized by *p2dMapping*.

structure" (represents a crystal structure), "suppressed" (sequence is being suppressed from the analysis, maybe incorrect), "replaced NCBI" (replaced at the NCBI database and should no longer be used), "replaced" (replaced with a sequence that is not from NCBI), "withdrawn" (withdrawn from the source database and no longer avaiable), "dead" (not used), "ignore" (sequence is ignored) and "unknown" (unknown status). Comments about the entry can be saved in the field "sequenceComments". References to the original sequence, can be stored in the fields "dbxref" (identifier of the original entry within the source database) and "sourceDatabase" (the name of the source database). Every entry in the *sequences* table contain a reference ("taxonomy_id") to a species in the *taxonomies* table (see section 3.2.1.2). *sequences* contains published and unpublished data simultaneously, therefore "private" can be used to mark them accordingly. Unforunately, protein sequence nomenclature is not unique. Homologous sequences from different organism can have varying names. For example, the most prominent Qa.I SNARE is called Syntaxin 1 (Syx1) in most organisms, but in the nematode *Caenorhabditis elegans* it is called Unc64, based on the screen in which it was discovered. Sometimes, even the same proteins from the same organism have different names (Synaptobrevin/Vamp). It is important to include as many names as possible of a specific protein. For exactly this purpose the field "aliases" can be used. Some sequences may be updated over time. If this happens, their status is set to "replaced". The new sequence contains a link to the replaced one in the field "replacedBy". As mentioned before, the "sequence" field can hold either a protein, DNA, or RNA sequence. The type of the sequence is indicated by the values of the field "sequenceType". "gene_id" is a foreign key that links the sequence to an entry in the *genes* table.

A gene can result in several different sequences (DNA, RNA or protein), according to possible splice variants. Data handling can be simplified a lot,

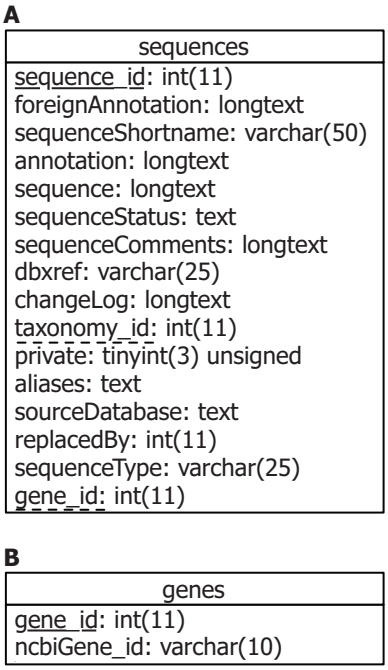

Figure 3.2-5: An UML-like schematic of the tables (A) *sequences* and (B) *genes*. Primary keys of each table are indicated by underlines, foreign keys by dashed underlines.

by knowing whether similar sequences are products of the same gene. To be able to store genes, the *Tracey* database contains the *genes* table (see fig. 3.2-5 (B)). Besides the primary key "gene_id", the table only contains an additional field for the unique identifier ("ncbiGene_id") of the NCBI genes database, to be able to link genes to this table [42]. The *genes* table is still quite rudimentary, but will be extended considerably in the future.

3.2.1.2 *taxonomies*

The main function of the taxonomy tables is to keep all sequences organized from a taxonomical perspective. This allows for quick extraction of all sequences belonging to a specific organism or a specific taxonomical branch (e.g. metazoa or plants). Each sequence entry is linked to a specific organism. NCBI provides an established system to structure taxonomical information. This comprehensive NCBI Taxonomy database [42] is used as

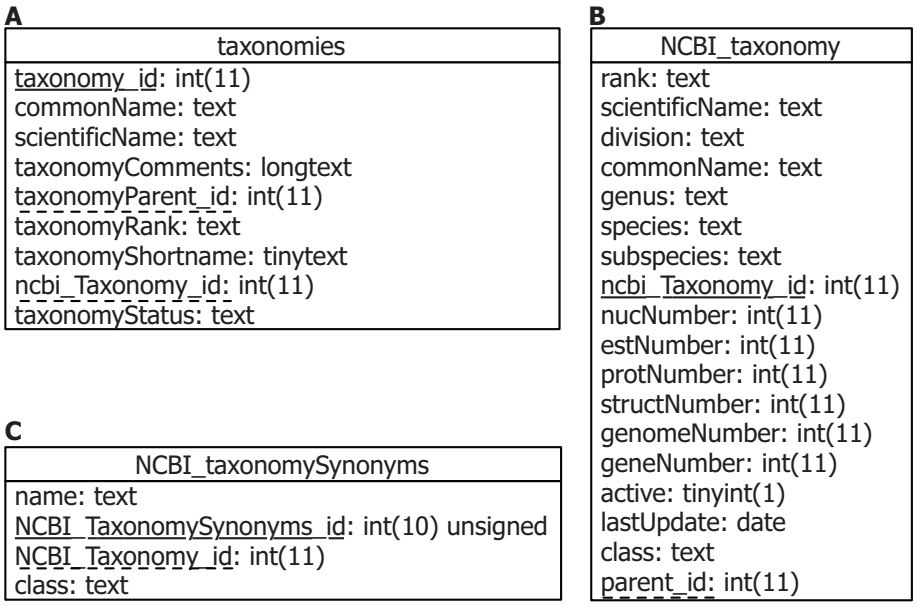

Figure 3.2-6: An UML-like schematic of the tables (A) *taxonomies*, (B) *NCBI_taxonomy*, and (C) *NCBI_taxonomySynonyms*. Primary keys of each table are indicated by underlines, foreign keys by dashed underlines.

a reference for the *Tracey* table *taxonomies* (see fig. 3.2-6 (A)). In fact, the tables *NCBI_taxonomy* (see fig. 3.2-6 (B)) and *NCBI_taxonomySynomyns* (see fig. 3.2-6 (C)) are kept synchronized with the NCBI taxonomy database, which ensures the integrity of the *taxonomies* table. The *taxonomies* table, as its name suggests, not only contains single organisms, but also their hierarchical taxonomic classification. Primary key of *taxonomies* table is "taxonomy_id", this unsigned integer values is unique for each dataset. The field "taxonomyParent_id" links an entry back to "taxnomy_id" of the parent. The table provides two names fields, "commonName" (e.g. human, fruit fly) and "scientificName" (e.g. Homo sapiens, Drosophila melanogaster). It is possible to add comments ("taxonomyComments") to every dataset. The ranks ("taxonomyRank") of taxonomy entries are directly taken from *NCBI_taxonomy* and reflect the phylogenetic rank of an entry (e.g. kingdom, phylum, family, genus, species, strain). Each taxonomy entry has a unique "taxonomyShortname". If not already in use, it is build by the initial

two letters of the species, followed the initial two letters of the strain (e.g. Homo sapiens, HoSa), otherwise a similar short name is chosen. A direct link to the *NCBI_taxonomy* table is stored in the field "ncbi_taxonomy_id". Sometimes, several species of the same genus are available (e.g. 14 species of *Drosophila*). Associated sequence are often similar and thus would not provide additional information, but could bias the analysis. For better discrimination each dataset in the *taxonomies* table has a specific status. Available statuses are, in descending order, "main reference" (species that are include in a standard analysis), "secondary reference" (can be used as a secondary reference), "additional" (additional species without new information but are included for completeness), and "unknown" (species without known status).

3.2.1.3 *motifs*

Motif predictions can result in multiple hits for the same position. Therefore, an expert has to control the predicted motifs and choose the correct ones, this is called "verification process". Newly predicted motifs (verifyMotifs) go into the *verifyMotifs* table (see fig. 3.2-7 (B)). Upon verification, the verified motifs get transferred into the *motifs* table (see fig. 3.2-7 (A)). The principal design of the tables *verifyMotifs* and *motifs* is identical. Primary keys of the tables are the fields "motif_id" and "verifyMotif_id", respectively. In order to connect a motif to a specific sequence, the "sequence_id" can be saved as a foreign key. All motifs that belong to the same protein family (domain) also possess an identical "motifname" (e.g. SNARE, C2). It is essential to know where a motif occurs within a sequence and if it bears any gaps. For this purpose, *motifs* and *verifyMotifs* offer the fields "startposition", "stopposition", and "gaps". The position fields contain simple integer values, but the gaps field contains a combination of the start position of the gap, followed by a colon and the gap length (e.g. 34:2 would be a gap of length 2 starting at sequence position 34). Specific comments can be saved

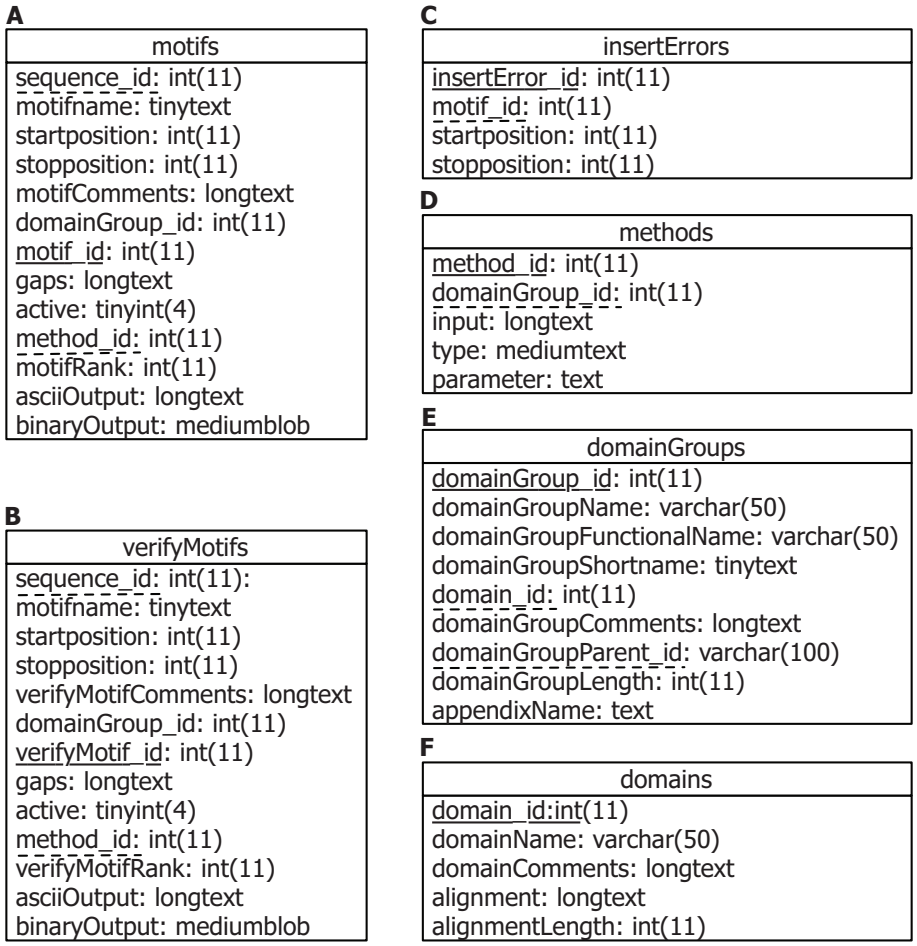

Figure 3.2-7: An UML-like schematic of the tables (A) *motifs*, (B) *verifyMotifs*, (C) *insertErrors*, (D) *methods*, (E) *domainGroups*, and (F) *domains*. Primary keys of each table are indicated by underlines, foreign keys by dashed underlines.

in "motifComments"/"verifyMotifComments". Both tables contain a "domainGroup_id" field to connect the motifs to a specific domain group in the table *domainGroups* (see fig. 3.2-7 (E)). The "active" field is specifically important for the *verifyMotifs* table. Initially, all predicted verifyMotifs possess an active value of "0". During the verification process, a reviewer assigns an active value of "1" to the correct verifyMotifs, whereas the other verifyMotifs get an active value of "-1". All motifs/verifyMotifs are connected to the *methods* table (see fig. 3.2-7 (D)), via the field "method_id". A "rank" is assigned to each motif/verifyMotif, to be able to determine the significance. The fields "asciiOutput" and "binaryOutput" contain method specific information. All American Standard Code for Information Interchange (ASCII) encoded information are stored in the "asciiOutput" field. In the case of an hmmsearch/hmmpfam prediction, this field would contain the E-value, the bitscore, the consensus string, the similarity, and the actual predicted motif in Extensible Markup Language (XML) format [77]. Binary information (e.g. images) are stored in the "binaryOutput".

Predicted motifs may contain insert errors, which are indicated by the prediction method. Upon verification, all insert errors are transferred into a separate table called *insertErrors* (see fig. 3.2-7 (C)). It is a small table that contains a primary key ("insertError_id"), a foreign key ("motif_id") that links entries back to their motif, and positions for the start ("startposition"), and the end ("stopposition") of the insert error. In the database of the *SNARE-Project*, insert errors were part of the *motif* table, but this turned out to be impractical.

The tables *motifs* and *verifyMotifs* have links to the methods table (see fig. 3.2-7 (D)). This table has the primary key "method_id" and is designed to associate domain groups ("domainGroup_id") with methods. The "type" field contains the actual method name (e.g. hmm). "input" and "parameter" can be used to specify optional parameters.

Several tables have connections to *domainGroups* (see fig. 3.2-7 (E)). Potential hierarchical relations of a classification are represented by domain groups. For example, the SNARE hierarchy (see fig. 1.3-8) shows the four main types Qa, Qb, Qc, and R. These basic types are stored in the database as the basic domain groups of the SNARE family. Additionally, each of the main types could be further refined into subtypes. These more specific domain groups are also stored in the database (e.g. Qa.I is involved in transport to the Endoplasmatic Reticulum, whereas Qa.IV is involved in secretion). The *domainGroup* table contains all domain groups within a specific domain (indicated by the field "domain_id"). Each entry has a unique unsigned integer field "domainGroup_id" as primary key. Three columns ("domainGroupName", "domainGroupFunctionalName", and "domainGroupShortname") contain different annotations for the domain groups, plus one column ("domainGroupComments") for possible comments. As mentioned before, domain groups might exhibit hierarchical relations. Therefore, domain groups have the field "domainGroupParent_id" that points to the "domainGroup_id" of their direct ancestor. The "domainGroupLength" for each domain group dataset can be stored in the respective field. Resulting from the protein domain architecture, SNARE domain groups all have the same length, whereas C2 domain groups can vary in their length. All members of the SNAP domain group have either a "B" or a "C" as their appendix, depending whether they represent the Qb or the Qc motif of the protein. The field "appendix" provides the possibility to store these appendices into the database.

All domains are organized in the table *domains*. Each entry has a unique primary key "domain_id" and unique "domainName" (e.g. SNARE, C2). Additionally, comments about the domain can be saved in "domainComments". The basis for every domain classification is a MSA. *domains* provides fields to store the "alignment" and the "alignmentLength".

3.2.1.4 *layouts*

Another new feature of *Tracey*, is the possibility to combine motifs into layouts. A layout contains the domains, a protein is composed of (domain compositions). For example, the SNAP-25 layout contains the motif of the domain group SNAP-25.b, followed by the motif of domain group SNAP-25.c. To be able to save all necessary information, a special architecture was designed. Entries in the table *layouts* (see fig. 3.2-8 (A)) are used in a similar fashion as motifs. Each one has a unique primary key "layout_id" and a connection to *sequences* via the foreign key "sequence_id". Comments about the layout can be saved in the "layoutComments" field. Each layout belongs to a protein layout group, linked through the field "proteinLayoutGroup_id". They have "layoutRanks" that are calculated by combining the significance values of the underlying motifs. Different statuses ("layoutStatus") can be given to the layouts. Currently, only two statuses can be assigend, "live" and "unknown". All layouts possess a "layoutString". This string describes how the layout is composed. A general layout string looks like

startposition1#domainGroup_id1,motif_id1#stopposition1;
startposition2#domainGroup_id2,motif_id2#stopposition2;
...
startpositionN#domainGroup_idN,motif_idN#stoppositionN;

with N equals the number of motifs the layout is composed of. Sometimes, a sequence is incomplete, but its domain composition is obvious. In this case the domain group id would be set but the "motif_id" would be substituted by a dash (−). Additionally, the position parameters would be set to either the start position of the following motif or the end of the sequence. Due to performance issues, the layout string is at the moment static. Eventually, this string has to be calculated dynamically, whenever related information change.

Similar to motifs that are hierarchically organized through domain groups, layouts are hierarchically organized via protein layout groups. The table *proteinLayoutGroups* can store these elements (see fig. 3.2-8 (B)). Protein layout groups are specifically designed to describe general domain composition of a group of protein sequences. For example, the aforementioned SNAP-25 layout is a protein layout group and its parent is the protein layout group doubleSNARE. Such an organization allows for the hierarchical representation of domain compositions, similar as for single domains via a domain groups. Protein layout groups have unique primary keys ("proteinLayoutGroup_id") and contain three fields ("proteinLayoutGroupName", "proteinLayoutGroupFunctionalName", and "proteinLayoutGroupShortname") for annotation purposes. Comments about the protein layout group can be stored in "proteinLayoutComments". The hierarchical organization of the protein layout groups requires a connection to a direct ancestor. This is realized through the field "proteinLayoutGroupParent_id". Each protein layout group is connected to a dataset in the table *proteinLayouts* via the field "proteinLayout_id".

Entries in the table *proteinLayouts* (see fig. 3.2-8 (C)) are used to the generally define a protein layout composition. Protein layout groups are specializations of this general composition. Each *proteinLayout* dataset possesses a unique primary key "proteinLayout_id" and a unique "proteinLayoutName". Comments can be stored in "proteinLayoutComments". The general composition is contained in the field "layout" and contains a string with domain identifiers (see 3.2.1.3), separated by semi-colons.

3.2.1.5 p2dMapping

Protein layouts are used to define the domain composition and more specific domain group compositions are organized via protein layout groups. It is indispensable to be able to assign domain groups in correct order to their

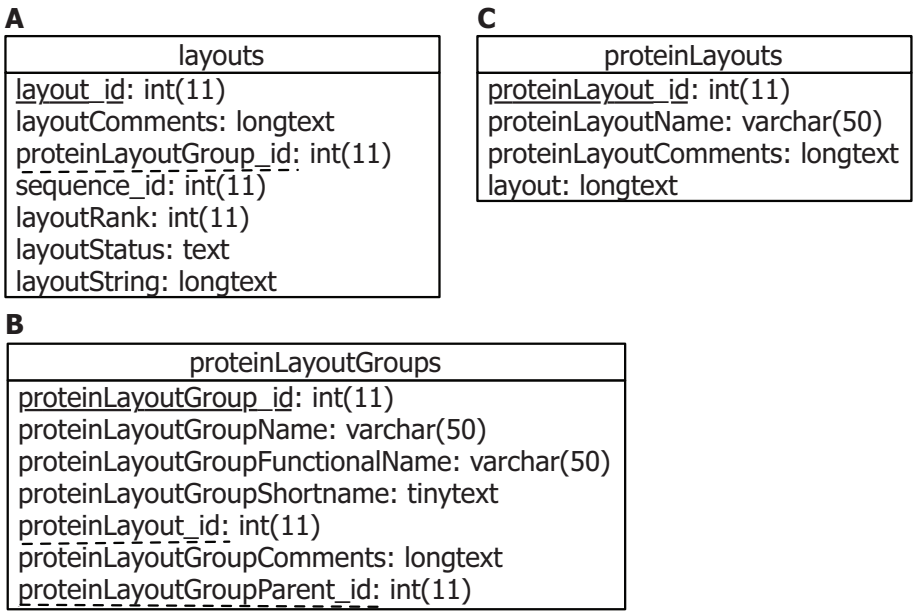

Figure 3.2-8: An UML-like schematic of the tables (A) *layouts*, (B) *proteinLayoutGroups*, and (C) *proteinLayouts*. Primary keys of each table are indicated by underlines, foreign keys by dashed underlines.

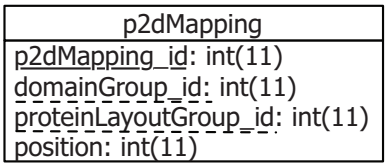

Figure 3.2-9: An UML-like schematic of the *p2dMapping* tables. The primary key of each table are indicated by underlines, foreign keys by dashed underlines.

protein layout groups. The table *p2dMapping* (see fig. 3.2-9) serves exactly this purpose. It has a unique primary key "p2dMapping_id" and fields for "domainGroup_id" and "proteinLayoutGroup_id". With this table it is possible to assign several domain groups to a specific protein layout group. The order of the domain groups can be determined by the field "position". It holds indices/positions for the domain groups, starting from zero.

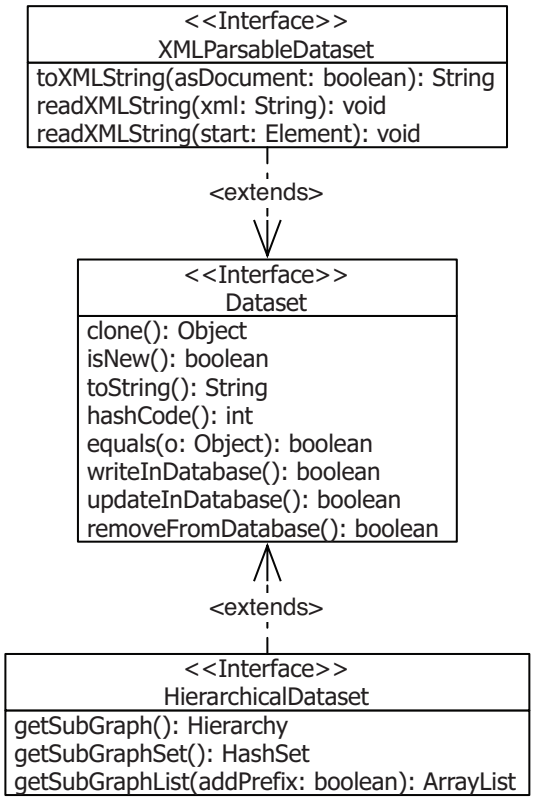

Figure 3.2-10: Interfaces of the *Tracey* Java database package. *Dataset* is designed as the central interface of the package. The interfaces *XMLParsableDataset* and *HierarchicalDataset* extend the basic interface with additional functionality.

3.2.2 *Tracey* Java database package

With the development of the *Tracey* database, the system contains a persistent data storage mechanism. Subsequently, a system was needed that can take care of data handling in a efficient manner (e.g. input/output, processing). Good experience made, during the development and work with the *SNARE-Project* Java database package (see section 1.3.3), influenced the decision to realize the data handling system for *Tracey* in a similar fashion. The Java database package for *Tracey* comprises classes for every table in the database (see fig. 3.2-4). Basic functionality of all classes is assured through implementation of specific Java interfaces.

Interfaces

The basic interface *Dataset* (see fig. 3.2-10) defines the fundamental functionality every class in the database package should implement. It includes functions for writing a dataset into the database (writeInDatabase()), updating a dataset that is already in the database (updateInDatabase()), deleting a dataset from the database (removeFromDatabase()), and checking if a dataset is already present in the database (isNew()). Additionally, the interface contains functions for cloning an object (clone()), string representation of an object (toString()), comparison of objects within a list (equals()), and hash code calculation (hashCode()).

Some of the tables, introduced in section 3.2.1, are designed to be able to contain hierarchical datasets (e.g. domain groups, taxonomies). Hence, the respective classes should reflect the hierarchically organization. Fundamental functionality of such classes are defined by the interface *HierarchicalDataset*. It contains functions that return the hierarchical data as a customized Hierarchy object (getSubGraph()) or as a Java HashSet (getSubgraphSet()). Additionally, the function getSubGraphList(addPrefix: boolean) returns the hierarchical data as a Java ArrayList. If the "addPrefix" parameter is true, the returning entries will have dash prefixes (-, - -, - - -,...) in front of their names. The prefix starts containing no dashes and adds one for every descending step within the hierarchy. This simplifies the differentiation of hierarchy levels (see group box for domain groups at the upper right side on fig. 3.2-14).

It is important to be able to extract datasets from the database into a document or to import dataset from a document into the database. A standardized way to do this is to encode data in hierarchical XML format [77]. To support this format, all classes implement the *XMLParsableDataset* interface. It includes functions to read in data in XML format either in full encoding (readXMLString(xml: String)) or from a specific element in the XML-

tree (readXMLString(Element: start)). Furthermore, the interface defines a function to encode datasets into XML format (toXMLString(asDocument: boolean)). If "asDocument" is true, the dataset is transformed in a full XML document, otherwise it is written as an XML tree element.

Database Classes

Each table in the *Tracey* database has an associated Java class in the database package and this class has member variables to be able to map all information of the associated table. For example, the class that corresponds to the table *domains* is called *DomainData* and it contains the member variables "domain_id" (int), "name" (String), "comments" (String), "alignment" (String), and "alignmentLength" (int), according to the columns of the table (see fig. 3.2-7 (F)). An object instance of a class represents a row of the associated table. Every class contains a set of constructors that can be distinguished into two distinct types. Firstly, the constructors that instantiate an object with all information contained as parameters. Secondly, constructors that retrieve all information directly from the database. In the latter case, the constructor contains a parameter that represents a primary key of a table (e.g. "domain_id" or "name" for the table *domain*). If such a constructor is invoked, it connects to the database and transfers the information, associated with the primary key, in the newly instantiated object. Each class contains several additional functions that aid in data handling. For example the class *DomainData* possess functions that can retrieve datasets from other tables that have a connection to a specific domain (e.g. *domainGroups*, *motifs* or *sequences*). The results are returned as a list, containing the respective objects (e.g. *DomainGroupData*, *MotifData*, or *SequenceData*). Establishing a database connection for a single dataset is very fast, but retrieving a large list of datasets can take a substantial amount of time, if the database connection has to be opened and closed for every entry. To

address this problem, most classes provide functions that open a database connection and transfer sets of entries of a table into a Java map data container (e.g. HashMap, TreeMap). Keys of such maps are either the primary keys or the unique name, which point to the respective objects (e.g. domain identifier point to *DomainData* objects or domain names point to *DomainData* objects). A database connection has to be established only once and an object can be retrieved immediately, if only the identifier or the name is available.
The flexible and comprehensive *Tracey* Java database package enables high performance database interaction and data handling for all upcoming tasks.

3.2.3 *Tracey* web interface

With the introduction of the *Tracey* database (section 3.2.1) and the Java database package (section 3.2.2), the bioinformatic part of *Tracey* (see fig. 3.2) is covered. Data handling should be carried out in a simple fashion and the data should be available from various locations. Hence, the best solution for data access is web based. Main goals of the *Tracey* web interface are performance and flexibility, not only for new add-ons, but also in the case of data presentation. Since the *Tracey* Java database package was already implemented, it was favorable to use a Java based web application framework. It was decided to use Grails, formerly known as "Groovy on Rails" (Groovy is a object-oriented programming language for the Java platform, often also used as a Java scripting language). Grails is an open source web application framework, based on the Java platform, that follows the Model-View-Controller (MVC) paradigm (see fig. 3.2-11). In the case of the *Tracey* web interface, the aforementioned classes of the Java database package, together with the tasks of the Web Access Manager (see section 3.2.4), build the Model layer. All components of the Controller and the View layer are implemented within the Grails framework, as either Groovy

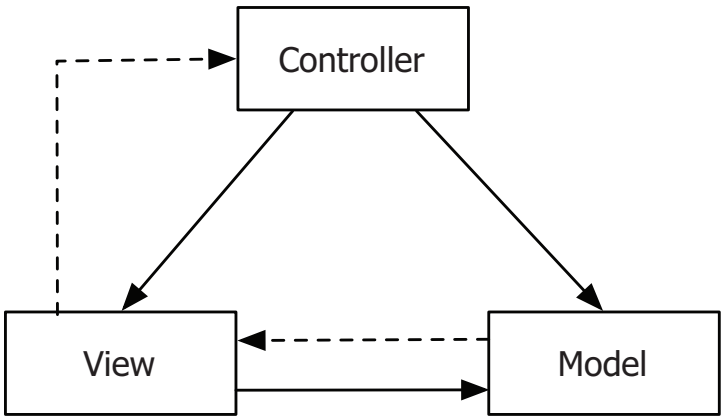

Figure 3.2-11: Schematic of the MVC paradigm. Model represents the respective data upon which the application operates. The underlying data can be simple information or it can be a persistent storage mechanism such as a database. View renders the model in a suitable form. In a web application, this is usually HyperText Markup Language or Extensible Hypertext Markup Language. The controller is responsible for coordination and interaction. It handles user actions, interacts with the model and delegates to a view.

classes or Groovy Server Pages (GSP) respectively.

All views of the web interface are implemented as GSPs. One major goal of the *Tracey* web interface is flexibility, therefore all views are build up in a modular fashion and each modul/panel can easily be replaced if necessary. Grails offers an elegant way to serve this purpose, called dynamic tag libraries. Customized functions (actions), that correspond to a tag, can be bundled in Groovy classes. These Groovy actions can be invoked from a GSP by adding the respective tag (see fig. 3.2-12). All panels in the following sections correspond to a customized tag. The results can easily altered by either changing the logic behind a tag or by exchanging the tag in the view.

To access any information, a user has to login to the system with the proper authentication. Similar to the web interface of the *SNARE-Project* (see section 2.4), the *Tracey* web interface also implements a rights management system with different ranks. The principal differentiation into "user", "submit", and "verify" was adopted from the SNARE web interface (see section

```
GSP Tag:
      <g:sequenceData ... />

Tag Library Action:
      def sequenceData = { attrs, body ->
            ...
      }
```

Figure 3.2-12: Code snippet example for the usage of customized dynamic tag libraries. The tag has to be invoked in a GSP, but the actual logic is implemented as a action in a Groovy class. Possible parameters in the tag call are indicated by three dots. All parameters are available through the *attrs* variable in the Groovy class.

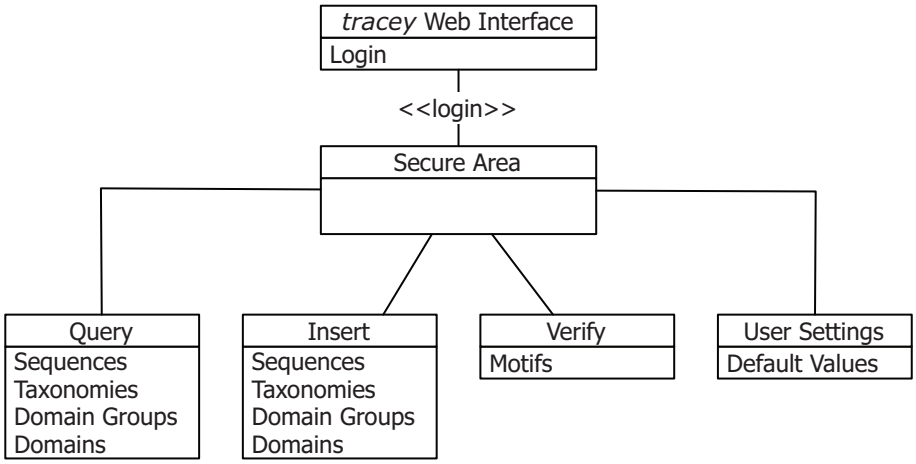

Figure 3.2-13: An UML-like schematic of the *Tracey* web interface. A user has to login with the proper authentication. The secure area can be sectioned into four parts, *Query*, *Insert*, *Verify*, and *User Settings*. Each of the main sections contains further subsections for specific tasks.

2.4), but it is possible to further refine these ranks. Additionally, in *Tracey* it is possible to grant ranks with respect to domains and protein layouts. For example, a user can have "verify" rank for a SNAREs, but only "user" rank for SNAPs. Once logged into the secure area, the web interface offers four main sections (see fig. 3.2-13). At the moment, the *Tracey* web interface is intended for internal use only, but ultimately it will also contain a public section.

3.2.3.1 Query

The main purpose of the query section is to access verified information. These information can be sequences, together with their predicted motifs, taxonomies, domain groups, and domains. Every user that falls into the "user" rank category is granted access to areas of this part of the web interface.

Sequences

Fig. 3.2-14 shows the search mask for sequences. The tabs at the top show all protein layouts contained in the database (SNARE is selected). Various search parameter can be specified. The upper panel of the search mask holds options for short name, taxonomy, domain groups, sequence_id, and motif_id. Unlike for the query example of the *SNARE-Project* web interface (see 2.4-4), a fuzzy search does not require a wild card anymore. A user can just type in all distinct parameter (separated by a ";") into a edit field and the system takes care of the rest. Additionally, multiple choices can be made in the group boxes for taxonomies (by clicking the check boxes) and domain groups (by marking the lines). The second panel holds search options for status, private, public, and protein layout groups. All edit fields and group boxes behave the same way as for the first panel. The options for domain groups and protein layout groups change with the selected protein layout tab. Sort options (short name, taxonomy, status, and private) can be chosen in the third panel, all results are sorted accordingly. A user can chose to view the results either as a list or a detailed view. Pressing one of the buttons in the bottom panel either resets the view or submits all parameter and starts the database search. The specified parameter ("sec9;sso" in the short name field and clicked checkbox for *Saccharomyces cerevisiae*) in fig. 3.2-14 results in the *Saccharomyces cerevisiae* sequences *SaCe_Sec9*, *SaCe_Sso1*, and *SaCe_Sso2*.

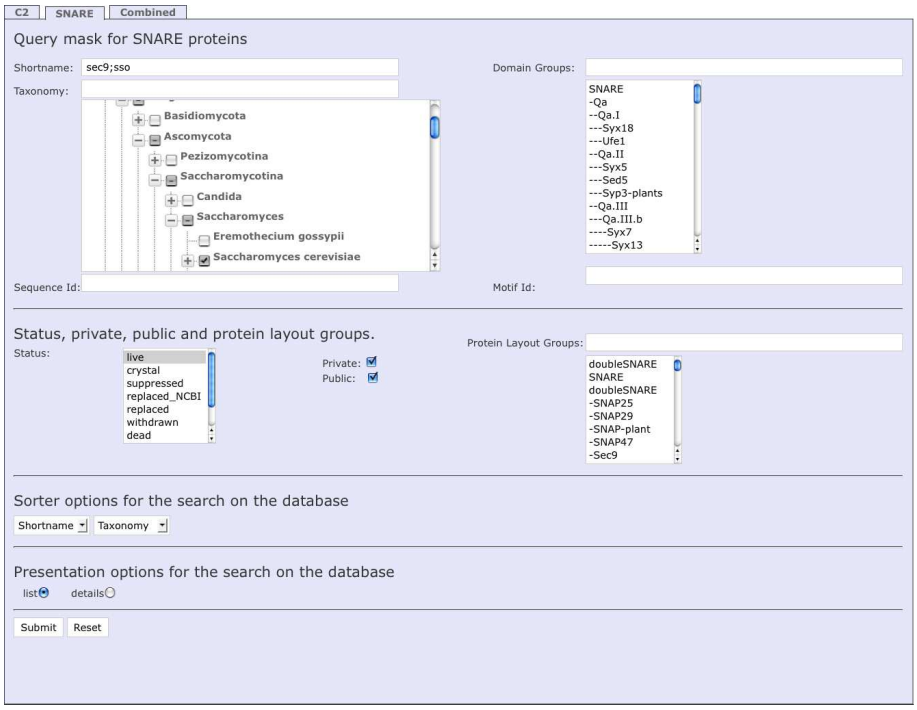

Figure 3.2-14: Sequence query mask of the *Tracey* web interface.

The list view (see fig 3.2-15) simply arranges the resulting sequences, with basic information (i.e. short name, taxonomy, aliases, status, foreign annotation, comments, and sequence), under each other. A user can obtain detailed information, either by clicking the belonging check boxes and pressing the change button at the bottom of the list or by directly clicking at the short name of a sequence of interest. Additionally, it is possible to extract the results in FastA format. Clicking the "Fasta File" link on the lower right side opens a pop-up window with the respective information.

Fig. 3.2-16 shows the detailed view of the protein *SaCe_Sec9*. The first panel shows all sequence related information (i.e. short name, aliases, taxonomy, status, comments, db reference, source database, gene identifier, foreign annotation, and sequence). A user can conduct a BLAST search [58] against all sequences with similar domains, by either clicking the links "BLAST" or "BLAST All". The latter performs a direct search against all sequences with

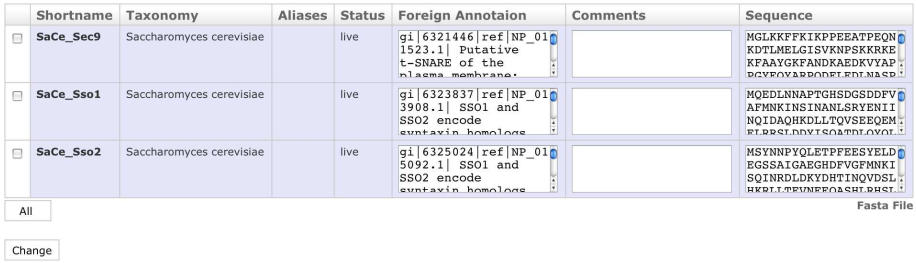

Figure 3.2-15: Sequence list view of the *Tracey* web interface.

similar domains, whereas the "BLAST" link firstly opens a pop-up window with an edit field for short name parameters. This field can be used to specify short names of the sequences that should be incorporated into the search (similar to the sequence query mask on fig. 3.2-14). The *SNARE-Project* web interface did not provide an option to perform BLAST searches. With this possibility, annotation of sequences can be accomplished much more efficient. The link "Recalculate Motifs" enables a user to recalculate the motifs of the actual sequence. Clicking on the link opens a pop-up window with various options. A user can specify, which motif should be recalculated and for which domain the search should be conducted. Additionally, it is possible to change some search parameters. If a motif is chosen for recalculation, the system deletes the actual motif and performs an HMM search (see section 2.2) of the actual sequence. All results are transferred into the *verifyMotifs* table. Similar as for the list view, it is possible to extract the results in FastA format (see "Fasta File" link in the upper right corner of the sequence panel).

The second panel in fig. 3.2-16 involves all layout related information. These are the protein layout group name, the status, the rank, and the domain composition. The domain composition is depicted in a graphical fashion and it shows all motifs of the sequence lined up. Motifs of different domains are shown in distinct combinations of shapes and colors. For example, SNARE motifs are shown as blue cylinders (3.2-16), whereas C2 motifs are shown as

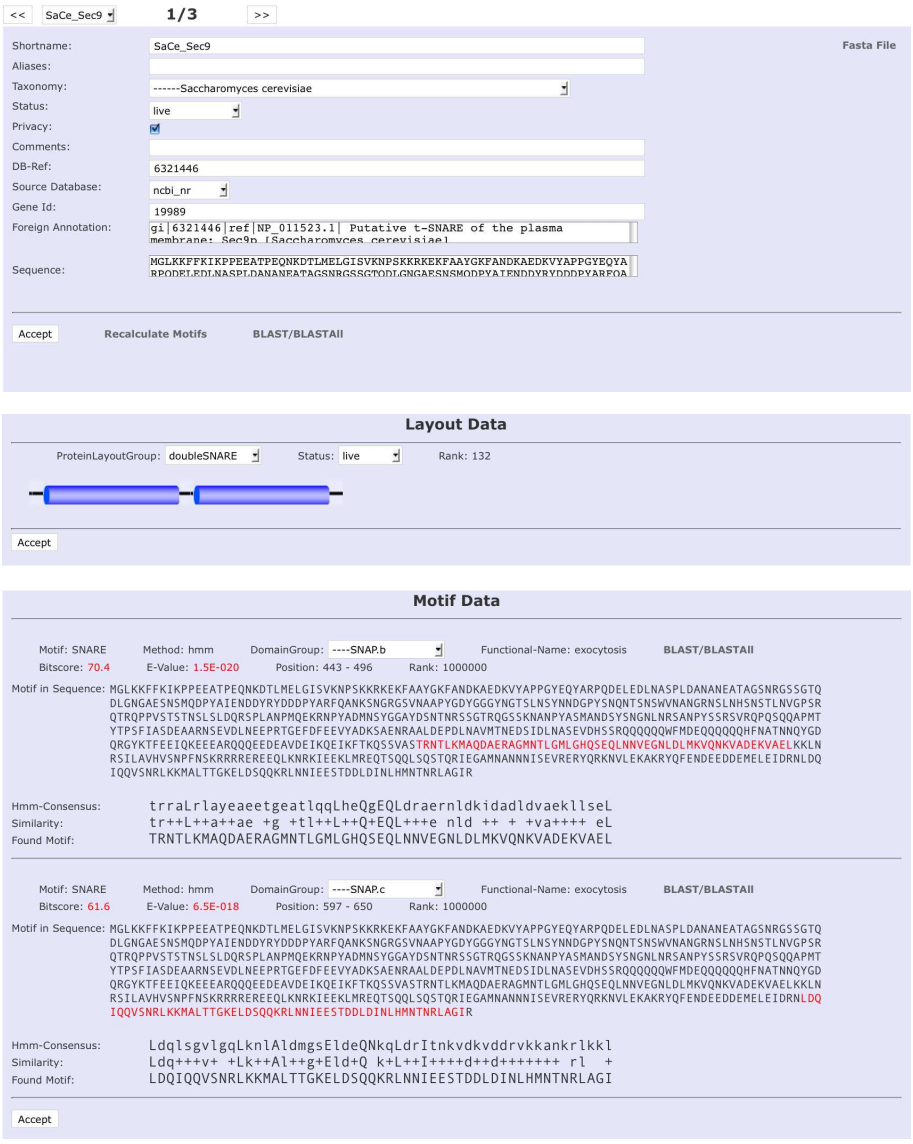

Figure 3.2-16: Detailed sequence view of the *Tracey* web interface. This view also includes to the sequence belonging layouts and motifs.

yellow ellipses. Hovering the mouse over one of these motifs shows a hint box with the domain group name and the position of the respective motif. For example, hovering the mouse over the first motif in fig. 3.2-16 would show that the motif belongs to the SNAP.b domain group and that it ranges from start position 443 until stop position 496 in the sequence.
All motif related information are listed in the third panel of fig. 3.2-16. These are the motif name, the method, the domain group, the functional name of the domain group, the bitscore, the E-value, the position within the sequence, and the rank. Additionally, the sequence, with the position of the motif highlighted by red amino acids, is shown. Furthermore, the panel shows an alignment of the HMM consensus string, the similarity, and the motif. This indicates the degree of conservation of specific positions within the motifs. The HMM consensus string depicts the amino acids with the highest probability according to the used model (positions with capital letters are higher conserved) and possible insert errors (”.”). Correlations between the HMM consensus string and the predicted motif are shown as the similarity. Letters show exact matches, whereas not exact, but conserved matches are indicated by a ”+” [54]. Every motif offers a BLAST search [58] option (see ”BLAST/BLAST All” links on the upper right of each motif). Both links behave similar to the ones of sequence panel, with the only difference that the searches only involve motif sequences.

Taxonomies, Domain Groups & Domains

In addition to sequences, the query section also provides a user with the possibility to browse through taxonomies, domain groups, and domains. Despite little differences, querying taxonomies, domain groups, and domains is done in the same way as discussed for sequences. The ”user” rights for this part of the web interface are independent of the rights for sequences and have to be specified for each area. The views are also build up in a modular

fashion and make use of the customized tag library.

3.2.3.2 Insert

The web interface allows for insertion of data into the database. It is mandatory for an user, who wants to insert data, to possess the rank "submit" for the specific category. Data can be inserted into the *sequences*, *domainGroups*, *domains*, and *taxonomies* table. Fig. 3.2-17 shows the insert mask for the aforementioned categories.

Every mask contains edit or drop-down fields for all necessary information. Inserting a sequence involves more than just the transfer of the specified sequence parameters into the *sequences* database. Upon submission, the system performs an HMM search for all available domains. Newly predicted motifs are written into the *verifyMotifs* table. To be able to verify those, a user needs to have access to the verify section of the web interface (see section 3.2.3.3).

The domain group insert mask (see fig. 3.2-17 (B)) contains a drop-down field for a domain. Choosing a domain entails possible choices of the drop-down field for parents. Without a choice for domain, the parents drop-down field contains all domain groups, a user has rights for. If a domain is chosen, only the domain groups that belong to this domain are contained in the drop-down field. Additionally, a domain group can have more than one parent. Clicking the "+" link next to the parent label adds another domain group drop-down field. Every additional field is also affected by the domain choice.

The domain insert mask (see fig. 3.2-17 (C)) contains a special field for alignment. Pressing the "Browse" button allows a user to choose a local alignment file (FastA format). Upon submit, the system reads in the alignment of the specified file and transfers it into the *domains* table. Along with this, the system also determines the respective alignment length.

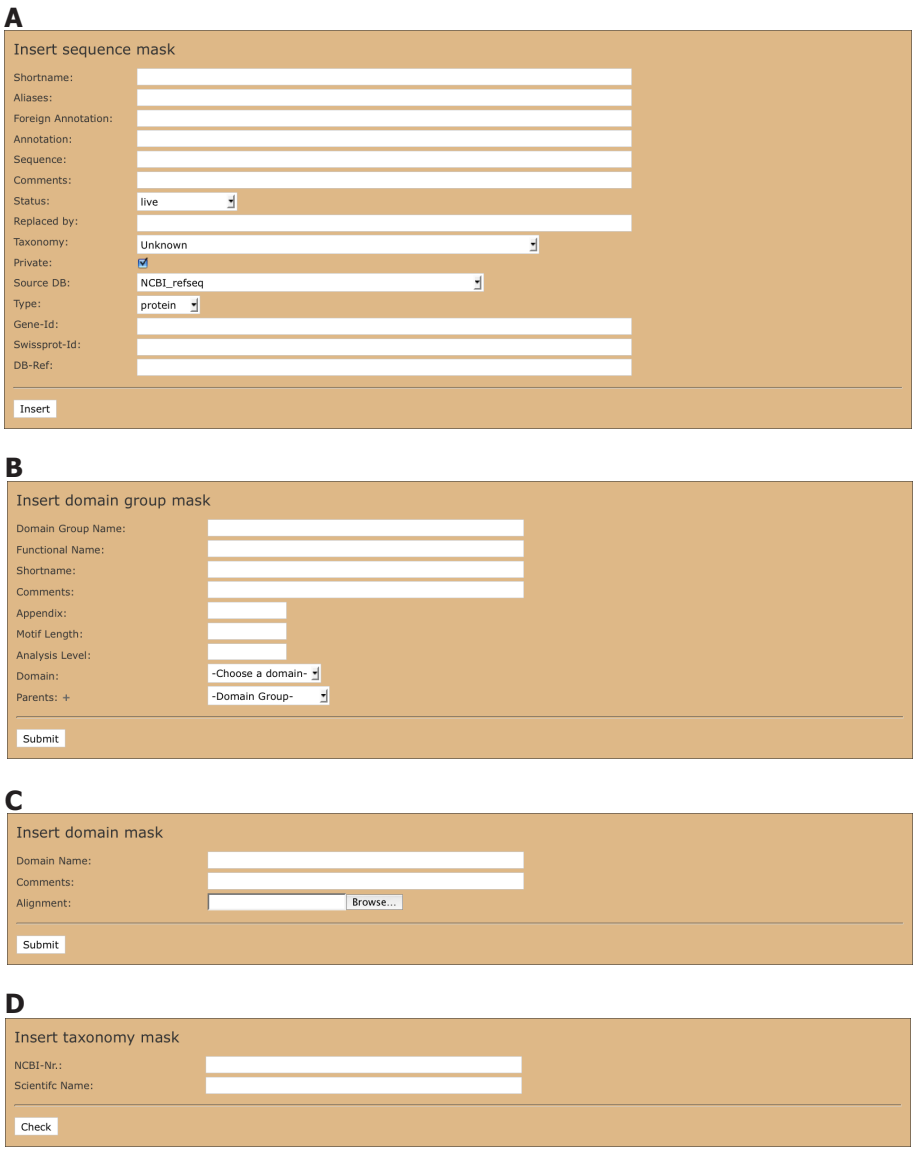

Figure 3.2-17: The insert masks for (A) sequences, (B) domain groups, (C) domains, and (D) taxonomies of the *Tracey* web interface.

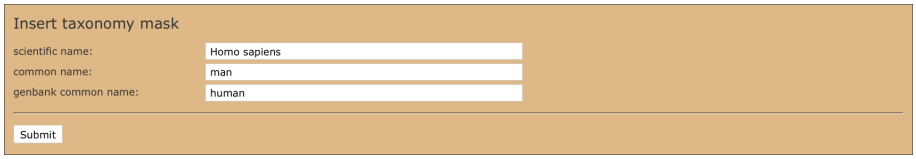

Figure 3.2-18: NCBI taxonomy entry for scientific name, common name, and genbank name of *Homo sapiens*. This is the result for entering the NCBI taxonomy number for *Homo sapiens* (9606) into the "NCBI-Nr" field on the insert taxonomy mask (see fig. 3.2-17 (D)).

Inserting a new taxonomy is somewhat special. The *Tracey taxonomies* table (see section 3.2.1.2) follows the NCBI Taxonomy database [42] quite thorough. A new taxonomy cannot just be inserted as submitted, but has to be transfered from the NCBI Taxonomy database [42]. Hence, the taxonomy insert mask (see fig. 3.2-17 (D)) contains only the fields "NCBI-Nr" and "Scientific Name". The system takes the information of either field and tries to find the respective entry in a taxonomy dump file. The latest version can be found on NCBI's File Transfer Protocol server [78]. After the entry is located, the scientific name, the common name, and the genbank common name of the respective taxonomy are presented to the user (see fig. 3.2-18). The user has to check the results and decide if this is the correct taxonomy. If this holds true, pressing on the "Submit" button will insert the dataset into the *Tracey taxonomies* table.

Some tables of the database contain columns that are marked as unique (e.g. sequence short name, foreign annotation, or domain name). If a user tries to insert information that is already contained in one of these columns, the system will return an error message. After the insertion is carried out, all new components are directly ready for use.

3.2.3.3 Verify

To ensure high quality data, newly predicted motifs initially go into the *verifyMotifs* table. The *Tracey* web interface provides a section to browse through these motifs and to mark them for verification or deletion. Browsing

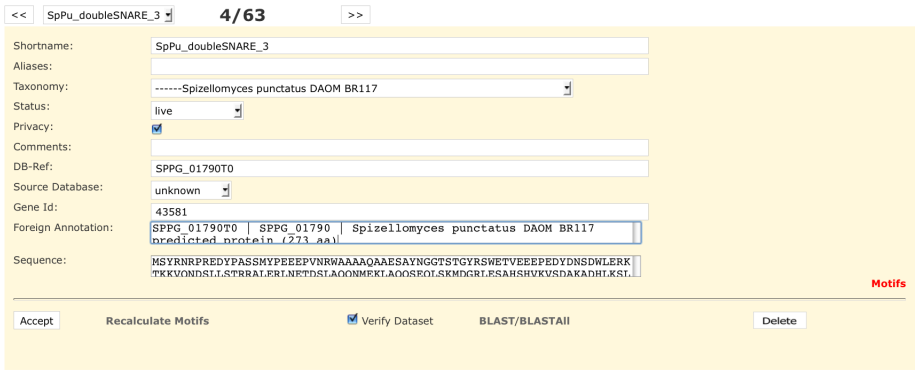

Figure 3.2-19: Detailed sequence panel of the verify section.

and verifying motifs is restricted to users with adequate knowledge, those will be assigned with the "verify" rank. The verify section contains a query mask, which is quite similar to the one of the query section (see fig. 3.2-14). Variations are the color (yellow for the verify section and blue for the query section), no preselection of the status and instead of a protein layout group field, it involves a taxonomy status field.

As shown on fig. 3.2-19, the detailed sequence panel of the verify section is very similar to the sequence panel of the query section (see fig. 3.2-16). One important difference is the red "motif" link on the lower right side of the panel. This link only appears, if the respective sequence has already verified motifs. Clicking on it opens a pop-up window with all motif relevant information. Another variation is the "Delete" button. If this button is clicked, the system deletes all motifs in the *verifyMotifs* table that are associated with the actual sequence. After that, it checks if the actual sequence has more information associated with it (e.g. additional motifs or layouts). If this is not the case, the system also erases the sequence from the database. Additionally, this panel contains a check box ("Verify Dataset") at the bottom. It has to be checked, if a motif (marked as verified) is supposed to be transferred from the *verifyMotifs* to the *motifs* table. In this case, it is not allowed for any motif to still be marked as "Unknown". Fig.

3.2-20 shows the motif panels of the verify section. The lower detailed motif panel is similar to the ones used in the query section (see fig. 3.2-16). One major difference are the three radio buttons ("Delete", "Unknown", "Verify") below each motif. If a user wants the motif to be verified, all motifs at the same position have to be marked either for deletion or verification. For verification of motifs with the *SNARE-Project* web interface, a user had to go through all motifs and click the respective radio buttons. This could be quite time consuming, because sequences can contain a lot more than two predicted motifs for a specific position. Furthermore, a sequence might be comprised of motifs at multiple positions. The *Tracey* web interface provides an effective solution to address this problem (see fig. 3.2-20 upper panel). All motifs are displayed in a graphical fashion, with different motif positions horizontally and multiple predictions for a specific position vertically. The example on fig. 3.2-20 shows two distinct motif positions ($78 - 132$ and $216 - 270$). Predictions resulted in two motifs for each position, SNAP.b/Qb.III for the first one and SNAP.c/Qc.III for the second one. Multiple motifs for one position are presorted by their E-value. The graphical depiction also includes the significance criteria (bitscore in blue and E-value in red) for the respective motif. These can be utilized to decide upon which motif to mark for verification. Pressing the "Accept" link marks the topmost motifs of each position. It is possible to alter the vertical sorting by dragging the respective motif to another position. Marking motifs in this panel has a direct effect on the radio buttons of the detailed motif panels. Fig. 3.2-21 shows the same example with marked motifs. In this case the "Accept" link was used to mark the topmost motifs. By doing this, the system automatically switches the radio buttons of the marked motifs to "Verify", whereas all other radio buttons are switched to "Delete". Occasionally, significance criteria alone might not be sufficient to decide upon verification. It is possible to compare each motif to similar, already verified

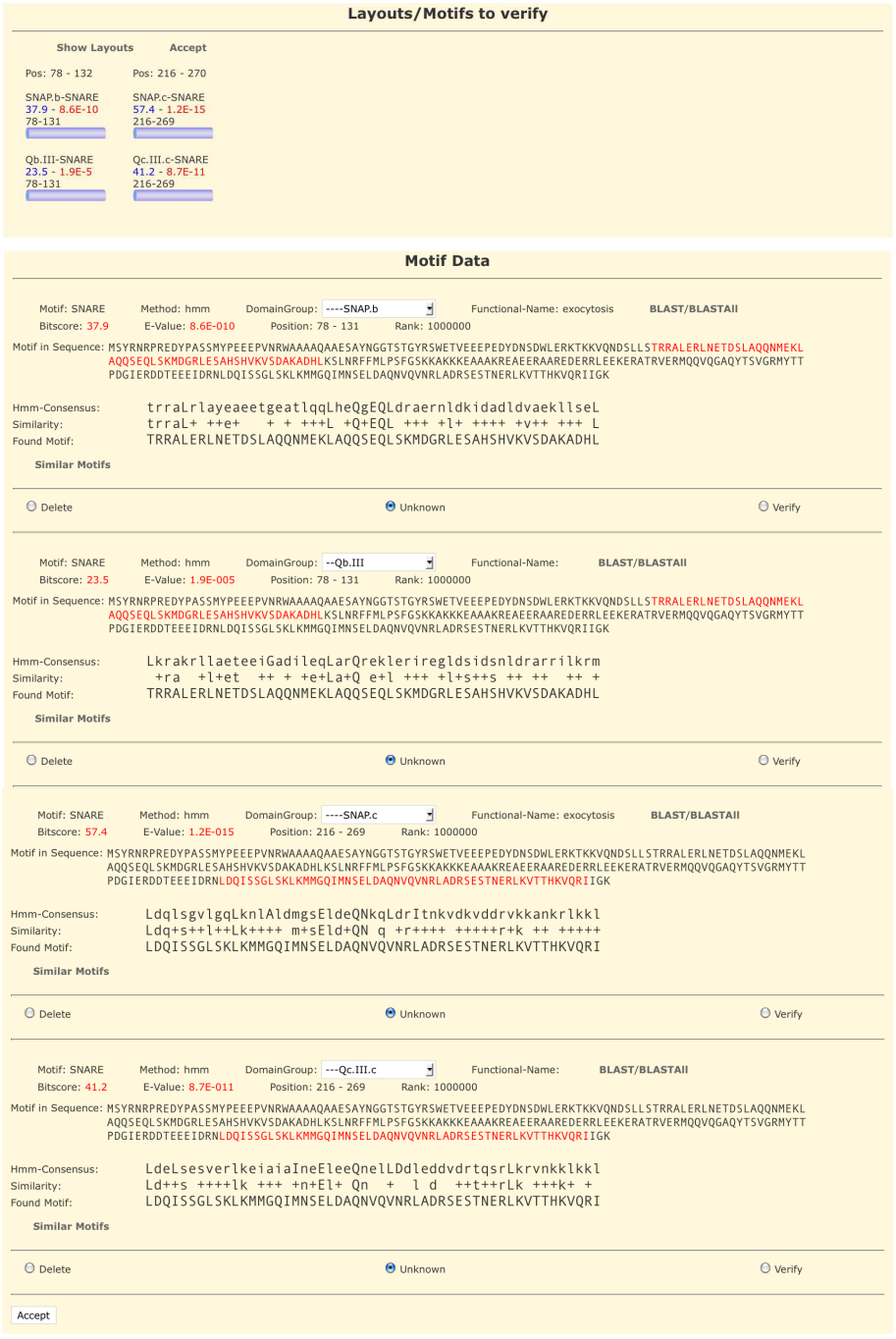

Figure 3.2-20: Motif display of the *Tracey* web interface verification section. Nothing is verified yet. All motif cylinders in the upper panel are displayed in the same color and all radio buttons are set to "unknown".

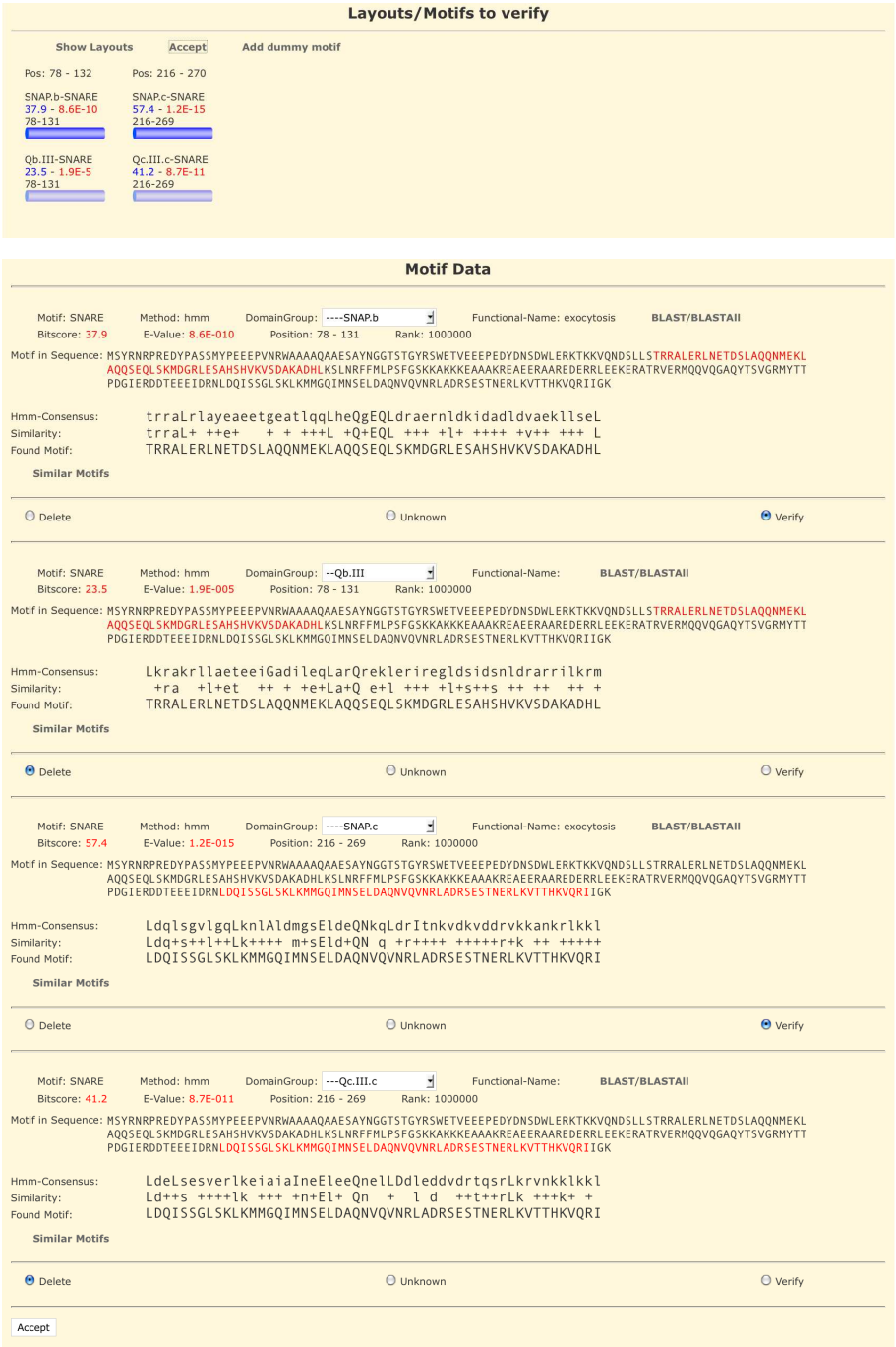

Figure 3.2-21: Motif display of the *Tracey* web interface verification section. The motifs are marked for verification/deletion. The upper two motif cylinders are displayed in darker blue and all radio buttons are set either to "'Verify" or "Delete".

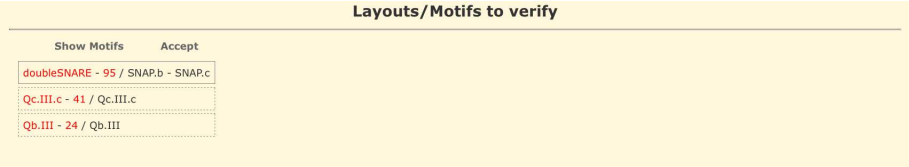

Figure 3.2-22: Layout verification panel of the *Tracey* web interface. Marked layouts are indicated by a continuous frame line. The numbers in red are the layout ranks, which are calculated by summing bitscores of the underlying motifs.

motifs. Clicking the "Similar Motifs" link starts a search for verified motifs that occur in the same species group and belong to a similar domain group (parents or children of the actual one).

Besides the motif verification, the system also allows verification via layouts. Clicking on the "Show Layouts" link at the upper panel on fig. 3.2-21 switches this panel to display layouts (see fig. 3.2-22). The depicted putative layouts are not stored in the database, instead the system calculates all possible layouts from the predicted motifs on-the-fly. Significance criteria for the layouts are the ranks. In the example on fig. 3.2-22 "doubleSNARE" has rank 95, "Qc.III.c" has rank 41, and "Qb.III" has rank 24. The ranks are the added bitscores of the underlying motifs. After the marking of all motifs/layouts the system transfers motifs, marked as "Verify", together with the layout into *motifs* and *layouts* respectively. If the verification was made through the layout mode (see fig. 3.2-22), the chosen layout and the belonging motifs are transferred. If it was done via the motif mode, the layout is calculated according to the chosen motifs and then together with the motifs transferred.

The *Tracey* web interface verification system is very efficient and allows for verification of motifs from up to 500 sequences per hour.

3.2.3.4 UserSettings

To further increase work efficiency, the web interface allows users to specify certain default values. Currently, these are "Presentation Mode", "Preferred

Protein Layout", "Verification Mode", "Direct Verification", and HMM search related parameters. Values for "Presentation Mode" can be "list" or "detail". Upon this choice, the results of the query and the verify section get presented in list or detailed mode. All protein layouts, a user has rights for, are possible choices of "Preferred Protein Layout". This parameter directly influences which protein layout is preselected at the search sequence masks of the verify and the query section (see tabs on top of fig. 3.2-14, SNARE is preselected). As introduced in section 3.2.3.3, verification can be done either by marking motifs (see upper panel of fig. 3.2-21) or by marking layouts (see fig. 3.2-22). The setting of parameter "Verification Mode" determines which mode is active. "Direct Verification" can be used to preselect the "Verify Dataset" checkbox (see fig. 3.2-19 at the bottom). Sometimes a user might want to work on predicted motifs, but does not want them to get transferred into the *motifs* table. In this case, this option can be set to "false" and the checkbox is not preselected. As already pointed out before, the link "Recalculate Motifs" on the detailed sequence panel of the query (see fig. 3.2-16) and the verify section (see fig. 3.2-19) allows for motif recalculation of a sequence. A user can specify certain HMM search specific parameters (e.g. E-value cutoff). Default values for these parameters can also be defined in the user settings section.

3.2.4 WebAccessManager

The WebAccessManager (WAM) (see fig. 3.2-3) is a tool that solves several problems that arose during the development of *Tracey*. Firstly, there is always the security issue. As published and unpublished data are stored in one database, it would be preferable to hide the database completely from access, even from the server hosting the *Tracey* web interface. A second issue is performance. The system needs to be able to conduct various calculations and any results should be presented over the web interface. It is

possible to separate the performance issue into two smaller problems, the running time should be minimal and it should be possible for any number of calculations to run simultaneously. Preserving data coherence is the third problem. Since multiple user might change information in the database at the same time, the system needs to be able to secure data integrity.
The WAM has been designed to solve the aforementioned problems. Java Remote Method Invocation technology builds the core of the WAM. This allows a devoloper to create distributed, remote Java objects. Such objects can be invoked on a Java virtual machine (JVM) and then sent through a network layer to another JVM. Possible results can be returned back to the original JVM. Using the apache commons package [79], a daemon service, called AccessManagerServer (AMS), was implemented. The AMS starts the AccessManager (AM) and takes care of keeping the service alive, restarting it upon possible crashes and error handling. For example, if the AM crashes for an unknown reason, the AMS immediately starts the AM again and thus downtime for the service is minimal. The AM is the central management facility of the WAM and its main functionality is the organization of different "Tasks". Any kind of request from the web interface to the AM has to be sent as a "Task". Additionally, each task needs to implement one of four currently available basic TaskType interfaces. These are ReadTask, ModifyTask, WriteTask, and LongRunningTask. These interfaces all extend the basic Task interface. It defines the basic operations that any task must be able to conduct (see fig. 3.2-23 (A)). The method "execute()" starts the operation of a task and the outcome can be obtained with "getResult()". Furthermore, the interface defines several maintenance methods: "getUserId()" returns the user identification of the user that requested the task, getStackTrace() returns possible error traces, getUserMessage() returns potential feedback to the user beyond the result, "getAgeInSeconds()" returns the time a task has been running for, and "getTaskType()" returns

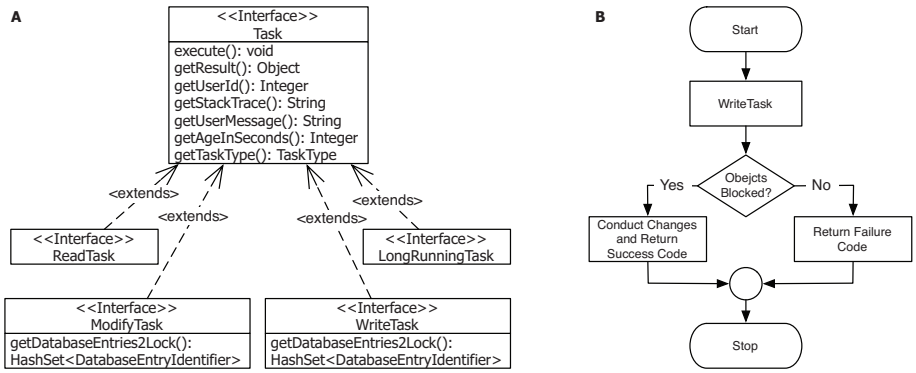

Figure 3.2-23: (A) Interfaces for the different tasks included in the WAM. On top is the basic interface that all tasks have to implement. ModifyTask and WriteTask have an additional method that provides access to identifiers of blocked elements during data modification. (B) Flow chart for submission of a WriteTask.

the type of the task.

ReadTask represents a request to extract data from the database. A ModifyTask also extracts data, but additionally indicates that the user, who requested the data, has the intention of modifying it. By doing so, the system stores the identifiers of the requested objects in a list for a given amount of time (specified by a program parameter). The ModifyTask interface defines the additional method "getDatabaseEntries2Lock()", which returns the identifiers of the blocked elements. To change the requested data, a user has to sent a WriteTask to the server. This task also implements the method "getDatabaseEntries2Lock()", which returns the object identifiers the task intends to change. If the objects are still blocked, write access is granted and changes can be made (see fig. 3.2-23 (B)). In this case, the elements are removed from the block list and the execute() method is invoked. However, if the elements are not blocked, either because someone else changed them beforehand or the time between the modify request and the write request expired, the write request will be denied. The final task type is called LongRunningTask. It is mainly used to run, possibly long-lasting, complex computations (i.e. alignment calculation).

To be able to use multiple processors or cores in parallel, Java provides the

ExecutorService (ES) technology. Basically, all ReadTasks are grouped together into one ES, all Write- and ModifyTasks into another ES, and all LongRunningTasks into a third service. The Write- and ModifyTask ES is restricted to a single core and thus serializes database access. ReadTasks ES and LongRunngingTask ES are not restricted to a single core, but LongRunngingTask ES runs on a separate set of cores, since it might conduct long-lasting, complex operations and could therefore overload the WAM hardware. Each task submitted to the WAM is enclosed in a TaskRunner object that implements the Runnable interface and these objects are redirected to the according ES.

The WAM has been proven to be a very powerful tool. For example, using the WAM allows for the removal of all database references from the web interface. In fact, since the web interface is only dependent on the Java objects, introduced in section 3.2.2, for database interaction, it could easily be moved to a different storage system. Additionally, the WAM enables the usage of modern chip design, by allowing calculations to run on multiple cores. Furthermore, it is possible to adapt the technology to be able to use a platform like GridGain [80] to run complex tasks on a computer cluster. In order to accomplish this, only the TaskRunner and the ES have to be adapted.

3.3 SNARE proteins in fungi

The underlying dataset for the classification of the fungi SNAREs comprises an overall set of more than 1500 SNARE proteins from 123 fungi species. Out of this, 70 species contain almost complete SNARE sets. Most of the sequences originate from the nr-database at *NCBI* [42], various genome projects (DOE Joint Genome Institute [81], Baylor College of Medicine [82], J. Craig Venter Institute [83], and Broad Institute [84]), and several EST databases (NCBI Expressed Sequence Tags database [42], Fungal Genomics

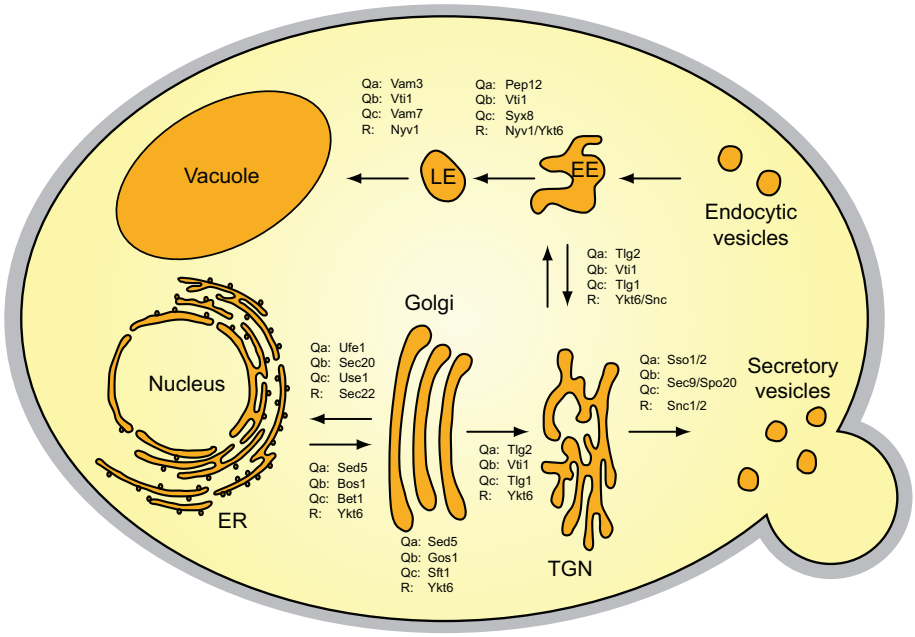

Figure 3.3-24: Schematic outline of the vesicle trafficking pathways and tentative assignement of the involved sets of SNARE proteins of baker's yeast. It should be kept in mind, however, that the assignment of some SNAREs to certain trafficking steps, in particular of the R-SNAREs, is still debated. Note that baker's yeast has two endosomal syntaxins, Pep12, and Vam3, that are thought to be involved in consecutive trafficking steps towards the vacuole, whereas other fungi only have one. [87]

Project [85], and Taxonomical Broad EST Database [86]). All sequences and species used in this study can also be found in the SNARE database at `http://bioinformatics.mpibpc.mpg.de/snare`.

For the classification of the fungi SNAREs, previously established HMM models were used [7] . The 53aa of the highly conserved SNARE core motif build the basis for these models. Fig. 3.3-24 shows a schematic of a yeast cell with its different inner compartments. As already elaborated by Kloepper et al. [7], fungi SNAREs can generally be divided into 20 distinct subgroups. Each subgroup can be associated with a distinct conserved trafficking step (see fig. 1.3-8). Compared to the assumed SNARE set of the proto-eukaryotic ancestor, the SNARE sets in fungi remained largely unchanged. In most organisms with a complete genome, each SNARE sub-

group is comprised only of one member. Thus only little over twenty different SNAREs were found in most fungi species. In other eukaryotic kingdoms (eg.g metazoa or plants), SNARE sets with 30 or more members are not unusual. Increase in SNAREs might have occurred during the transition from a unicellular to a multicellular lifestyle [88]. Together with other studies ([9, 89, 90]), the idea was promoted that an expansion of the SNARE set is generally linked to the rise of multicellularity. Interestingly, fungi developed multicellularity independently, but in contrast to plants and metazoa, this change was not accompanied by an expansion of the SNARE set.

Despite the very basic repertoire of SNARE proteins in fungi, some lineages underwent duplication events. Mostly secretory and endosomal/vacuolar trafficking pathways were affected by such events, whereas SNAREs of the more ancient trafficking routes between the Endoplasmatic Reticulum, the Golgi apparatus, and the trans Golgi network were basically unaffected.

A closer look revealed that the fungi SNAREs are comparable to the set of the single-cell choanoflagellate *Monosiga brevicollis*. This organism is thought to be closely related to animals. Unlike *Monosiga brevicollis*, basal animals possess a somewhat increased number of SNARE proteins [88].

Remarkably, the fungi *Batrachochytrium dendrobatidis* and *Blastocladiella emersonii* both contain a sequence, classified as Qb.III.d (novel plant SNARE (Npsn,)) [91], which is absent in *Monosiga brevicollis*. Initially, Npsn was thought to be only present in plants [91] and protists [7], but not in choanoflagellates or animals [88]. *Batrachochytrium dendrobatidis* and *Blastocladiella emersonii* both belong to the chytrid/Chytridiomycota division of the fungi kingdom. This lineage is very basal and its members are considered to be the most primitive fungi [92]. In agreement with this, phylogenetic reconstructions usually places the SNARE protein sequences of this species closely to the root. Taking all this into account, it seems that Npsn was part of the assumed SNARE repertoire of the proto-eukaryotic ancestor, but was later

lost in choanoflagellates, metazoa, and more derived fungi.

3.3.1 Vam7 is an apomorphy of the fungi lineage

Generally, fungi possess three Qc.III-type SNAREs, Tlg1, syntaxin 8 (Syx8), and Vam7 [93, 94, 95, 96], whereas basal metazoa and the choanoflagellate *Monosiga brevicollis* possess only two Qc.III homologs (Syx6 and Syx8) [88]. The latter two SNAREs are homologs of Tlg1 and Syx8, respectively. Strikingly, Vam7 cannot be found in any other eukaryote, but was found in all fungi with a completely sequenced genome, even in the basal chytrid *Batrachochytrium dendrobatidis*. This was already reported before by Yoshizawa and colleagues [36]. However, their analysis had a number of incorrect biological conclusions, due to a inferior cluster approach that led to wrongly assigned relationships. Additionally, several well established SNAREs were also missing in their analysis. Despite the similarity of the Vam7 SNARE motif to the other members of the Qc.III group, it cannot be judged clearly, whether Vam7 is a descendent of either Tlg1 or Syx8.

Vam7 is the only SNARE with an N-terminal PX domain, but without TMR. The PX domain can interact with the phospholipid phosphatidylinositol 3-phosphate [97], which is specific for membranes of the endosomal and vacuolar pathways. This suggests a role of Vam7, together with its cognate SNAREs Vam3 (Qa.III.b), Vti1 (Qb.III.b), and Nyv1 (R.III), in endosomal trafficking or homotypic vacuole fusion [98, 99]. Furthermore, the acquisition of this novel membrane binding domain by Vam7 possibly compensated for the loss of the C-terminal TMR. This protein is a unique invention of the fungi kingdom and is the only SNARE with this specific domain structure. Thus, Vam7 is a defining feature (apomorphy) of the fungi lineage and can therefore be used as a criterion for the recognition of fungal species.

3.3.2 SNARE changes in the endosomal/vacuolar pathways of the *Saccharomycotina*

As mentioned before, only a few SNARE repertoire changes could be observed in the fungi lineages. One occurred in the *Saccharomycotina* (sometimes referred to as *hemiascomycetes*), this is a subphylum of the *Ascomycota* phylum. Two well studied organisms of this subphylum are *Candida albicans* (human pathogen) and *Saccharomyces cerevisiae* (baker's yeast). The Qa.III.b SNARE proteins of this subphylum underwent a duplication, giving rise to Pep12 and Vam3. It is believed that Pep12 and Vam3 are involved in trafficking to late endosomes and vacuoles, respectively. While Pep12 also is present in other fungi lineages, Vam3 seems to be a duplication specific to the *Saccharomycotina* (see fig. 3.3-25). This observation is also in agreement with previous studies [100]. Interestingly, there are no clear homologs of Vam3 in other multicellular eukaryotes ([91], [88]), this indicates that the duplication of the Qa.III.b SNARE into Pep12 and Vam3 occurred independently to the metazoan duplications of early endosomal syntaxins that resulted in syntaxin 7, syntaxin 13, syntaxin 17, and syntaxin 20 [88]. Another possibility is that the unique duplication of Qa.III.b-type SNAREs into Pep12 and Vam3 in *hemiascomycetes*, might be the result of of a specialized homotypic vacuole fusion machinery that originated in this lineage. This process occurs during the asexual reproduction of a yeast cell, in which the vacuoles of the mother cell get fragmented and the resulting multiple small vacuoles subsequently segregate into the budding daughter cell. These fragments fuse again and become the new vacuoles of the daughter cell [101, 102, 103]. Previous studies used homotypic vacuole fusion to analyze the role of SNAREs and other factors in the fusion process [99, 98].
The R-SNARE Nyv1 seems to interact specifically with Vam3 and not with Pep12, which is thought to interact preferentially with the R-SNARE Ykt6. It has been debated before, whether Nyv1 is really homologous to the meta-

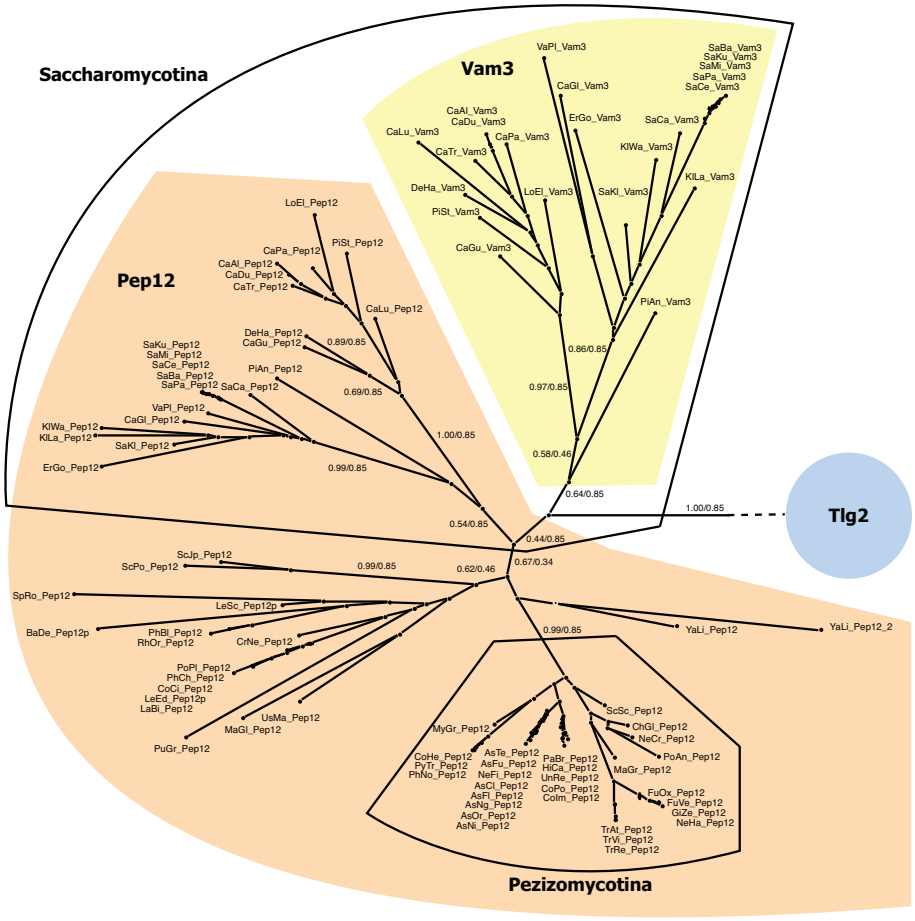

Figure 3.3-25: Unrooted Qa.III.b tree of fungi SNAREs highlighting the events of gene duplication and diversification of the Qa.III.b SNARE group in fungi. The endosomal Qa-SNAREs (Qa.III.b-type) of fungi split into two major branches, Pep12 and Vam3, within the Saccharomycotina. The syntaxin involved in trafficking towards the trans Golgi network, Tlg2 (Qa.III.a-type) is shown as outgroup. The labels on the major branches represent the Likelihood Mapping (left) and AU support values (right). [87]

zoan R-SNARE Vamp7 and previous work indicated that the SNARE motif of Nyv1 from baker's yeast is only distantly related to Vamp7 in animals [104, 105]. In contrast to this, the results of the conducted analysis, showed that Nyv1 clearly falls into the R.III group, the same group Vamp7 belongs to as well. A recently published structure promotes this further, as it shows that Nyv1 indeed contains a canonical profilin-like N-terminal extension [106], often also referred to as longin domain [104, 105]. Previous

analyses were able to find this domain in other R-SNAREs Sec22 (R.I) [107] and Ykt6 (R.II) [108]), but could not confirm its existence in Nyv1. Generally, a homolog of Nyv1 can be found in all fungi, except the members of the *Schizosaccharomyces* (*Schizosaccharomyces pombe* and *Schizosaccharomyces japonicus*). In comparison to the other fungi lineage, Nyv1 of *Saccharomycetales* seems to be quite derived and it is therefore not surprising that it has been difficult before this analysis to acknowledge Nyv1 as a bonafide endosomal R-SNARE.

3.3.3 A whole genome duplication resulted in an increased set of secretory SNAREs in *Saccharomyces cerevisiae*

The conducted analysis revealed that of all inspected fungi, *Saccharomyces cerevisiae* possesses the largest set of SNARE proteins. Its repertoire comprises 26 different SNAREs (including the regulatory proteins Sro7 and Sro77). This is surprising, because so far baker's yeast was considered to be a reduced organism. Besides the previously mentioned fungi specific Vam7 and the *Saccharomycotina* specific Vam3, it also contains four closely related pairs of secretory SNARE proteins, Sso1/Sso2 (Qa.IV), Snc1/Snc2 (R.IV), Sec9/Spo20 (Qbc.IV), and Sro7/Sro77 (R.Reg). This increase of secretory SNAREs probably arose during a well established whole genome duplication (WGD) in this organism [109, 110]. Similar duplications are also present in closely related species (*Saccharomyces paradoxus*, *Saccharomyces mikatae*, *Saccharomyces bayanus*, and *Saccharomyces kudriavzevii*). Together, these species form the *Saccharomyces sensu stricto* group (see fig. 3.3-26). A WGD is mostly followed by a loss of duplicated genes. In the case of baker's yeast, only 14% of all duplicated genes were kept. Interestingly, all duplicated SNAREs that were retained are involved in the secretory process, suggesting that this provided the organism with a selective advantage. Out of the three homologous secretory pairs, only the Qbc-SNAREs (Sec9/Spo20)

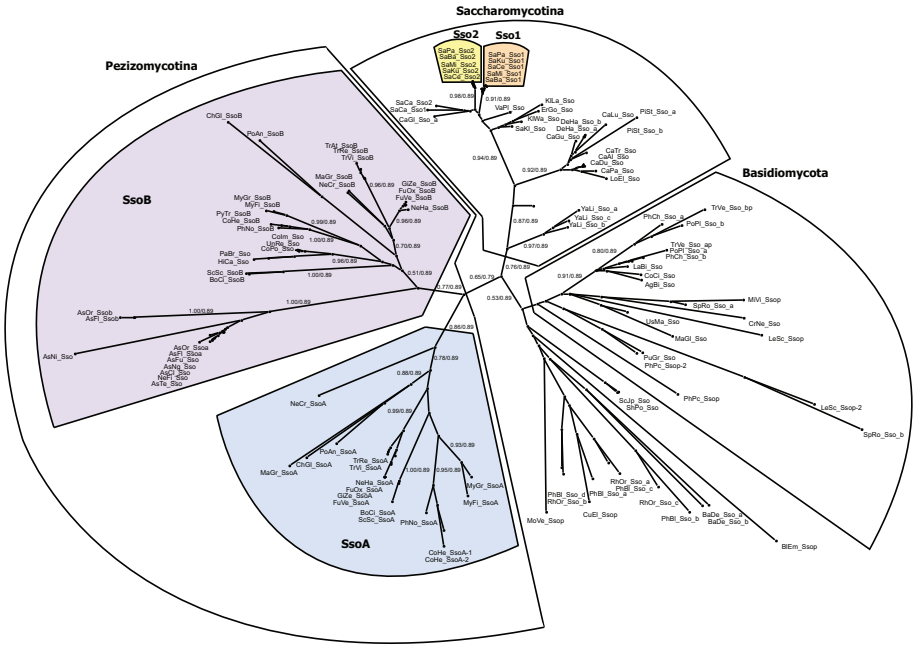

Figure 3.3-26: Unrooted Qa.IV tree of fungi SNAREs highlighting the events of gene duplication and diversification of the Qa.IV SNARE group in fungi.The phylogenetic tree of secre- tory syntaxins (Qa.IV-type) reveals independent expansions in Pezizomycotina and in the Saccharomyces sensu stricto clade. The latter expansion probably occurred during a whole-genome duplication, during which the other secretory SNAREs were duplicated as well. Note that independent duplications of Sso genes occurred in other lineages as well. In each tree, the diverged SNARE types are shown by different colors. The labels on the major branches represent the Likelihood Mapping (left) and AU support values (right). [87]

diverged quite significantly (approximately 37% identity), while the secretory syntaxins (Sso1/Sso2 with approximately 79% identity) and R-SNAREs (Snc1/Snc2 with approximately 74% identity) are much more similar. Besides sequence similarity differences between Sec9 and Spo20, Spo20 is also shorter and it lacks a rather long N-terminal extension [111], which is typically for this SNARE type in *Saccharomycotina* lineage. As mentioned before, both proteins are involved in secretory processes, but they are specialized to different developmental stages in baker's yeast. Whereas Sec9 interacts with both secretory syntaxins and synaptobrevins in secretion during vegetative growth, the more deviated Spo20 is required only for sporulation. During sporulation in *Saccharomyces cerevisiae*, the prospore mem-

brane, which envelops each daughter nucleus during meiosis, is generated de novo adjacent to the spindle pole bodies by the secretory SNARE machinery [112]. Sec9 and Spo20 show partial functional redundancy during sporulation, because only the Spo20/Sec9 double mutants exhibit a complete loss of prospore membranes. While Spo20 mutants show mild sporulation phenotyp, Sec9 mutants have no sporulation phenotype [112, 113, 111]. Additionally, even though the two secretory syntaxins Sso1/Sso2 share a high sequence similarity, they might have some functional differences, since only Sso1 seems to be necessary during sporulation [114, 115].

3.3.4 The tomosyn SNARE motif in *Saccharomycotina* is degenerated

It has been established before that tomosyn (R.Reg), a regulatory R-SNARE without a membrane anchor, is a member of the assumed SNARE repertoire of the proto-eukaryotic cell [7]. The structure of the tomosyn yeast homolog Sro7 revealed that the protein possesses two consecutive seven-bladed β-propeller domains at its N-terminus [116]. Absence of a transmembrane anchor at the C-terminus makes tomosyn unable to participate in the fusion process [14, 117, 118, 119]. Instead, it is believed to function as a regulator by controlling the accessibility of the Q-SNAREs acceptor complexes in polarized secretion [116]. *Saccharomyces cerevisiae* comprises two homologs of tomosyn, Sro7 and Sro77 (55 % identity) [120]. Both proteins do not possess a canonical R-SNARE motif and are therefore very often confused with the Lethal giant larvae (Lgl) factor, which supposedly has a function in establishing epithelial cell polarity in animals [121, 122]. Previous studies established that Lgl derived independently from tomosyn in animals [123, 117], where it lost its SNARE domain [88]. The fact that the R-SNARE motifs of tomosyn homologs are present in other fungi lineages ([123, 117]) points to the conclusion that the motifs of Sro7 and Sro77 degenerated independently

in the *Saccharomycetales* clade.

3.3.5 The *Pezizomycotina* lineage contains clearly diverged secretory syntaxins

Duplications of SNAREs involved in secretory processes, particularly the secretory syntaxin (Sso, Qa.IV), occurred in several fungi lineages. Mostly, the duplicates did not diverge much and so far no clear pattern between duplications and lifestyle or taxonomical grouping could be detected. Previous studies ([100, 124]) established that *Neurospora crassa* (*sodariomycete*) possesses two markedly diverged secretory syntaxins (nsyn1 and nsyn2). The same holds true for the *Pezizomycotina Trichoderma reesei* [125]. *Pezizomycotina* (older name *Euascomycota*) are filamentous fungi that reproduce by fission rather than budding. Together with the other major subphylum, *Saccharomycotina*, they comprise most of the fungi phylum *Ascomycota*. This phylum, commonly known as "sac fungi", is the largest in fungi and covers more than $64,000$ subspecies. *Ascomycota* are defined by the so called ascus, a microscopic sexual structure that surrounds the meiotic spores. During vegetative growth, most *Pezizomycotina* grow hyphae at their tips, a process that involves highly polarized secretion of cell wall material. The two syntaxins of *Trichoderma reesei*, Sso1 and Sso2, were found to be involved in different secretory processes. Whereas Sso2 was found in the apical compartments of actively growing hyphae, Sso1 was found in older, non-growing hyphae. It has been shown before by *in vivo* fluorescence resonance energy transfer studies that the two proteins exert their secretory function at spatially segregated areas of the plasma membrane [125]. Fig. 3.3-26 clearly shows that the two diverged secretory syntaxins in *Trichoderma reesei* and *Neurospora crassa* represent a split that arose within the lineage of *Pezizomycotina*.

Interestingly, not all species within the *Pezizomycotina* show a duplication

of the Qa.IV SNARE group. Some species of the *Eurotiomycetes*, including the genus *Aspergillus*, only contain one copy of the Sso gene. This points to the conclusion that the group branched off before the duplication occurred [126]. Within the *Aspergillus* genus, some species (e.g. *Aspergillus oryzae*, *Aspergillus flavus*) actually contain two different secretory syntaxins, but it seems that this duplication occurred independently within the *Eurotiomycetes*. To clarify the difference between the duplicated secretory syntaxins in *Saccharomyces cerevisiae* (Sso1 and Sso2) and the ones in *Pezizomycotina*, the latter are named SsoA and SsoB.

The duplication in *Pezizomycotina* did not affect the cognate SNARE (i.e. Snc (R-group) and Sec9 (Qbc-group)) partners of the secretory syntaxins. Subsequently, these protein should be able to interact with both SsoA and SsoB. For *Trichoderma reesei* this was shown by Valkonen et al. [125]. Of notice to mention is that in a few *Sodariomycetes* (i.e. *Magnaporthe grisea*, *Neurospora crassa*, *Podospora anserina* and *Chaetomium globosum*) Sec9 is markedly diverged.

3.3.6 Fungi phylogeny is recapitulate by SNAREs

Recent phylogenetic analyses, based on multi-genic information, showed that the phylogenetic relationships within the fungi kingdom might differ from the established opinion. Furthermore, studies like [126, 127, 128, 129, 130, 92, 109, 110] introduced a new view on certain fungal clades that differed clearly from previous classifications of yeasts, which were usually defined mostly by morphological analysis and growth responses. Individual gene trees based on orthologous SNAREs concur with the newer classification. Together, this points to the conclusion that SNARE proteins have diversified only slowly during evolution, but fast enough to reflect species evolution within the fungi kingdom. For a better glimpse at the resolution of fungal species evolution, a phylogenetic tree based on the concatenated sequences

of all singleton SNAREs was reconstructed. This tree reflects the recently refined classification of the major fungi lineages (see fig. 3.3-27). The major fungi phyla *Basidiomycotina* and *Ascomycotina* are clearly distinguished in the tree. A closer look into the *Ascomycotina* reveals the major subphyla, *Pezizomycotina* and *Saccharomycotina*. The third subphylum, *Schizosaccharomyces*, clearly sits on the outside of the *Ascomycotina*. Within the *Pezizomycotina*, the fungi of the *Eurotiomycetidae* split first. The difference between the *Eurotiomycetidae* (only one Sso) and the other members of the *Pezizomycotina* (SsoA and SsoB) was previously discussed in section 3.3.5 and fig. 3.3-27 (*Split Sso*) clearly shows that this characteristic emerged after the split of the *Eurotiomycetidae*.

Two major clusters can be distinguished in the *Saccharomycotina* phylum. One is the *Candida* clade. Members of this clade translate the codon CTG as serine instead of leucin [133, 132]. The other major clade of the *Saccharomycotina* is the *Saccharomyces* complex. These two distinct clusters are clearly separated in all individual trees built from orthologous SNAREs. Interestingly, *Yarrowia lipolytica*, the single member of a third cluster of the *Saccharomycotina* splits earlier. The top of the tree in fig. 3.3-27 shows the evolutionary relation of some more basal fungi. Concatenated sequences of two *Mucoromycotina*, *Rhizopus oryzae* and *Phycomyces blakesleeanus*, and the chytrid fungus *Batrachochytrium dendrobatidis* were incorporated into the tree reconstruction. It can be seen that these species clearly differ from the *Basidiomycotina*.

3.4 SNAP proteins

As introduced in section 1.1.2, members of the SNAP family aid in SNARE disassembly by mediating NSF binding to the assembled SNARE complex. The SNAP family was already discovered more than 30 years ago [25]. Three isoforms (α-, β-, and γ-SNAP) were purified from bovine brain cytosol. Since

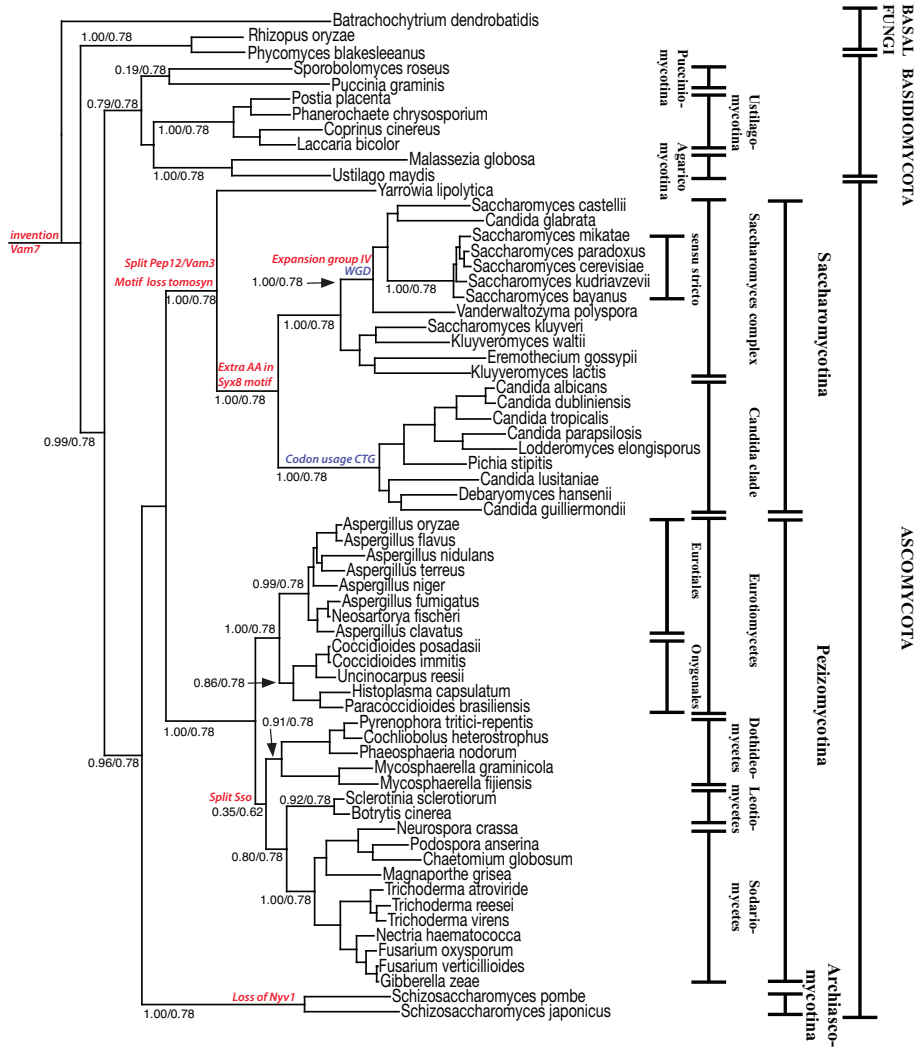

Figure 3.3-27: Phylogenetic relationships based on concatenated SNARE sequences from 66 sequenced fungal genomes.The major clades are named. Species that share the WGD [109, 110] and those with the different code usage (CTG) [131, 132] are indicated in blue, SNARE protein-derived synapomorphies are indicated in red. Support of the clades are represented by likelihood mapping values (left) and AU support values (right). The best AU support value returned by the bootstrap analysis was 0.78. [87]

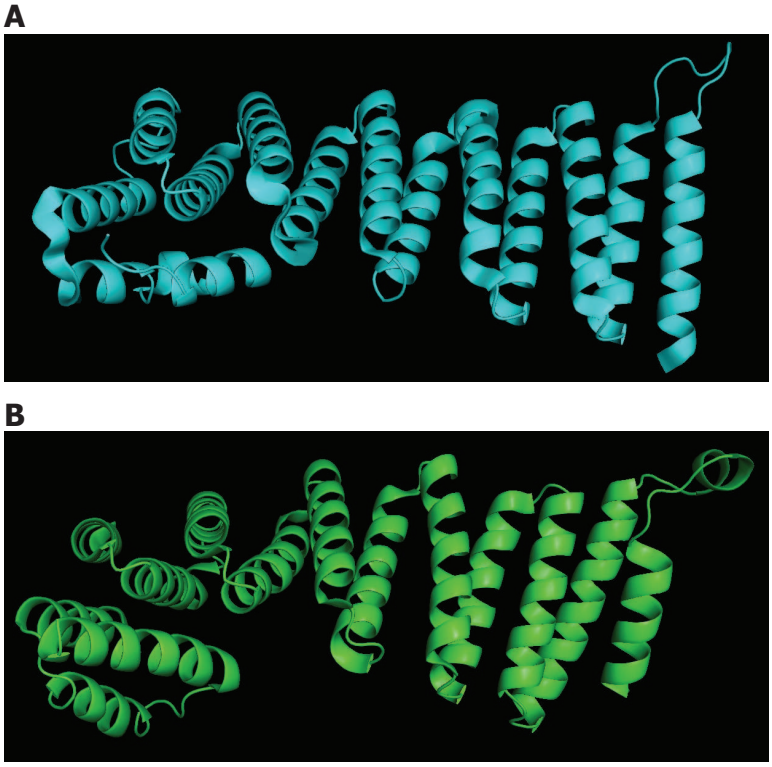

Figure 3.4-28: Structures of (A) Saccharomyces cerevisiae α-SNAP (Sec17, PDB code 1QQE) [135] and (B) monomer B of the γ-SNAP structure of *Brachydanio rerio* (PDB code 2IFU) [136].

then, only little progress has been made analyzing the evolutionary history of the SNAP family. This is especially surprising, considering the importance of SNAPs in all intracellular trafficking step. For instance, severe growth defects have been shown for a temperature sensitive α-SNAP mutant (Sec17 mutant) in yeast [134, 25]. To gain a deeper insight into the evolutionary development of the SNAPs, a detailed classification of the family was conducted in this thesis.

Structures of baker's yeast α-SNAP (Sec17) and zebra fish γ-SNAP show a high degree of structural similarity (see fig. 3.4-28). Both structures reveal an elongated all-helix protein, containing 14 α-helices in α-SNAP [135] and 15 α-helices in γ-SNAP [136]. For an initial SNAP alignment, the two structures were used as reference points. 41 α-, β-, and γ-SNAPs from var-

ious species were aligned. This led to a first SNAP HMM. Subsequently, this model was used to scan the Reference Sequence (RefSeq) database from NCBI. This resulted altogether in about 300 SNAP sequences. Three HMMs were trained for the three SNAP subtypes (α-SNAP with length of 275aa, β-SNAP with 273aa, and γ-SNAP with 286aa) and additionally a general HMM with all SNAPs combined (SNAP with 273aa). Another scan of the RefSeq database, the est database, and several genome projects, followed by removal of duplicates and misassembled sequences led to an overall set of 434 SNAP sequences. In detail, the dataset included SNAPs from 69 animals, 79 fungi, 19 plants, 47 protists, and 2 viruses.

A phylogenetic analysis revealed two distinct, basic SNAP groups, one consisting of γ-SNAP and one consisting of α- and β-SNAP (see fig. 3.4-29). Only α- and γ-SNAP could be found in all eukaryotic lineages. This suggests that the duplication into α-SNAP and γ-SNAP must be very ancient and that these two constitute the set of SNAP genes present in the assumed proto-eukaryotic ancestor. Whether this duplication occurred synchronously to the event that gave rise to the twenty basic SNARE subtypes remains yet elusive.

In the branch of α- and β-SNAP, β-SNAP probably arose by duplication of α-SNAP in vertebrates (see fig. 3.4-29). In fact, it has been reported that two consecutive rounds of WGD in vertebrates occurred. This resulted for example in an enlarged secretory SNARE repertoire [88]. Furthermore, previous studies have shown that α-SNAP is ubiquitously expressed in a wide range of tissue, whereas β-SNAP is a brain-specific isoform [137]. Together, this indicates that this additional SNAP must have provided a selective advantage in vertebrates. A high degree in sequence identity between α-SNAP and β-SNAP in vertebrates further supports that the two proteins are closely related. For example, α-SNAP of *Homo sapiens* is quite similar to other vertebrate α-SNAPs (e.g. *Xenopus laevis* 90%, *Danio rerio*

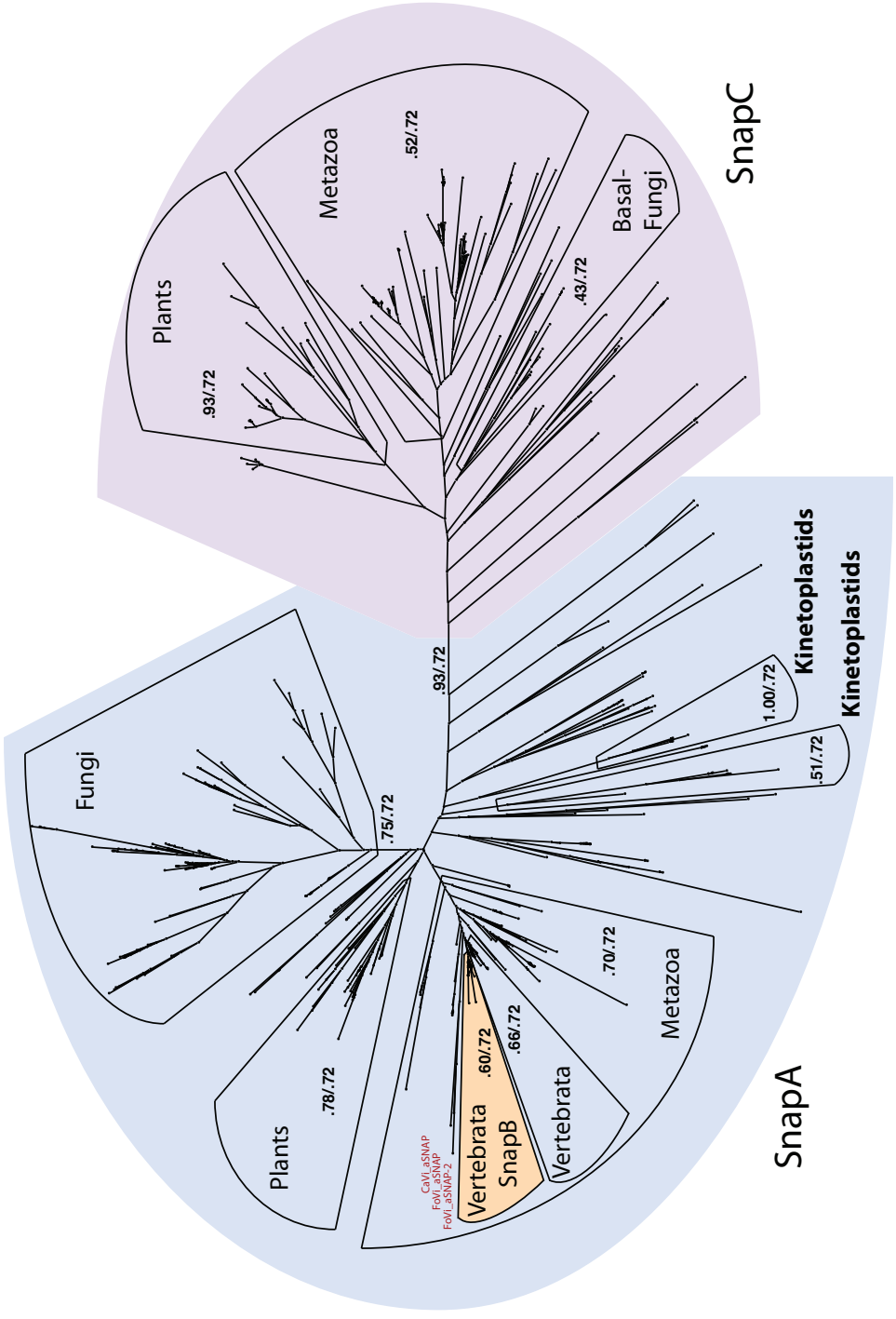

Figure 3.4-29: Unrooted phylogenetic SNAP tree of various eukaryotic lineages. α-, β-, and γ-SNAP branches are color-coded. The labels on the major branches represent the Likelihood Mapping (left) and AU support values (right)

84% identity) and vertebrate β-SNAPs (e.g. *Homo sapiens* 83%, *Xenopus laevis* 85%, *Danio rerio* 81% identity), whereas it is much less identical to non-vertebrate α-SNAPs (e.g. *Branchiostoma floridae* 75%, *Nematostella vectensis* 64%, *Ciona intestinalis* 58%), not to mention γ-SNAPs (e.g. *Homo sapiens* 22%, *Xenopus laevis* 22%, *Danio rerio* 22%, *Branchiostoma floridae* 20%, *Nematostella vectensis* 20%, *Ciona intestinalis* 17%). Strikingly, the inspected birds, *Gallus gallus* and *Taeniopygia guttata*, only possess β-SNAP, but no α-SNAP. Unfortunately, these are the only two bird genomes available. Inspections of additional genomes are necessary to confirm that birds, maybe even their reptile ancestors, lost α-SNAP.

Interestingly, in addition to eukaryotic α-SNAP, the analysis also showed α-SNAPs in *Canarypox virus* and *Fowlpox virus* (see fig. 3.4-29, color-coded in red). Presence of a gene, encoding α-SNAP in both species, was already reported previously [138, 139]. The viruses belong to the *Avipoxvirus* genus, which is a member of the *Poxviridae* family. The virus can be transmitted either by insect bites or wound contamination (moderate form) or by airborne infection (severe form) and is able to spread onto humans. *Canarypox virus* α-SNAP has 47% sequence identity with α-SNAP and 45% identity with β-SNAP of *Homo sapiens*, whereas the *Fowlpox virus* homolog has 43% and 41% identity, respectively. A high degree of sequence conservation in different strains of the *Fowlpox virus*, combined with no obvious replication defect upon α-SNAP deletion, suggests a role of viral α-SNAP in virus-host interaction [139]. Several ideas on the function of α-SNAP in virus-host interaction have been postulated [139], but so far none of them could be confirmed.

Another interesting duplication of α-SNAP took place in the kinetoplastids (see fig. 3.4-29). The species *Leishmania infantum*, *Leishmania major*, *Trypanosoma brucei*, and *Trypanosoma cruzi* all contain more than one distinct α-SNAP. All these species are parasitic and possess a single emergent flagel-

lum. So far, it is not clear what advantage that additional SNAP provided for the organisms.

As mentioned earlier, the γ-SNAP type was found all kingdoms of the *Eukarya* domain. However, the presence of few fungi species in the γ-SNAP branch was somewhat surprising, since Sec17 (i.e. the bona-fide α-SNAP homologue) was the only reported SNAP in fungi up to now. Interestingly, all fungi species containing a γ-SNAP are rather basal (see fig. 3.4-29). A similar loss occurred in the phylum *Apicomplexa*. Except of *Toxoplasma gondii* all inspected members of this lineage seem to also have lost γ-SNAP. Unfortunately, the cause for the losses of γ-SNAP are rather obscure, but might be due to the different lifestyle of these organisms.

In general, all eukaryotic species contain only a small number of SNAPs. Most fungi and the *Apicomplexa* reveal that only one SNAP type is sufficient to ensure functionality of the SNARE disassembly mechanism. Different groups of SNAREs mediate fusion of vesicles to different compartments and they are tightly regulated by various factors. It is quite imaginable that disassembly is a SNARE unspecific process that does not require any regulators.

3.5 C2 domain proteins in fungi

The first collection of C2 domain sequences contained about 150 domains from various proteins, but only few organisms. A first phylogenetic analysis revealed three distinct groups. HMMs were trained for these groups and a scan of the NCBI nr database resulted in more than 1500 C2 domain proteins altogether. The analysis of a subsequent phylogenetic reconstruction resulted in 15 refined HMMs. Currently, the database contains almost 3500 sequences with about 5300 C2 domains from NCBI databases (nr, RefSeq, and EST) and several genome projects. These proteins are distributed over 160 organisms from various eukaryotic kingdoms. A general refinement of

		BUD2	MUG190	NEDD4	PSD2	TCB	fUnc13	fPKCa	fPKCb	fPLCa	fPLCb	fc2a	t1
Basidiomycota		*	**	**	*	*	*	*	*	*	(*)	*(*)	*
Ascomycota	Pezizomycotina	*	*	*	*	*	*	*	*	*	*(*)	*	*
	Saccharomycotina	*		*	*	**(*)	*	*	*	*	*(*)	*	
	Schizosaccharomycetes		*	**	*	**	**	*		*		*	
Basal Fungi		*	*	**	**	***	**	*	*	*	*	**(*)	*

Table 3.5.1: The table shows the occurrence of the basic C2 domain proteins in the different fungi lineages. Asterisks in brackets indicate proteins that can only be found in some organisms of the lineage.

the HMMs is currently in progress.

The large dataset renders the classification and analysis of the C2 domain family challenging. C2 domains are particularly abundant in animals, even in basal organisms (e.g. 50 proteins with 90 C2 domains in *Trichoplax adhaerens* or 84 proteins with 144 C2 domains in *Nematostella vectensis*). The analysis of SNAREs in fungi revealed that fungi encompass a relative simple SNARE set, mostly comprising the SNAREs of the proto-eukaryotic ancestor. With this in mind, the C2 domain analysis was firstly focused on the fungi kingdom.

With the refined HMMs, it was possible to identify the C2 domain proteins in fungi. Generally, 12 proteins with C2 domains were found in most fungi. These are BUD2, meiotically up-regulated gene 190 protein (MUG190), neural precursor cell expressed, developmentally down-regulated 4 (NEDD4), phosphatidylserine decarboxylase 2 (PSD2), TCB, fungi uncoordinated family member 13 (fUnc13), two PKCs, two PLCs, and two proteins with unknown function, referred to as fc2a and t1. Some lineages contain multiple of the basic C2 domain proteins, whereas other lineage lost some of the basic factors (see table 3.5.1).

BUD2 is a GTPase activating factor for Rsr1p/Bud1p and it seems to be required for axial and bipolar budding patterns [140]. The presence of a C2 domain in BUD2 (bud2 C2 domain) was reported before [33]. The conducted analysis revealed that BUD2 is present as singleton in almost all fungi lineages, except for *Schizosaccharomycetes*, in which it seemed to be

lost independently (for an overview see table 3.5.1). Homologs of this protein were also found in choanoflagellates and animals.

MUG190 has two consecutive C2 domains (mug190 C2 domains). So far nothing was known about the function of protein, except that it was found to be up-regluated during meiosis [141]. The analysis showed that most fungi contain one copy of this protein, except of the *Basidiomycota*, in which it seems to be duplicated. Interestingly, MUG190 is not present in *Saccharomycotina*. This might be due their ability to reproduce asexually by budding.

NEDD4 is a ubiquitin ligase involved in regulating various intracellular processes [142]. It contains an N-terminal C2 domain (nedd4a C2 domain), several WW domains and a C-terminal HECT domain [142]. The conducted analysis revealed that basal fungi and the *Basidiomycota* possess two different NEDD4, whereas the *Ascomycota* seem to have lost one of them. Interestingly, *Schizosaccharomycetes*, which are part of the *Ascomycota* phylum, also contain two NEDD4 (for an overview see table 3.5.1). This indicates that one NEDD4 must have been lost independently in the subphylum *Saccharomycotina* and *Pezizomycotina* of the *Ascomycota*. Homologs of fungi NEDD4 were also found in of all animal lineages.

PSD is an enzyme that converts phosphatidylserine to phosphatidylethanolamine in both prokaryotes and eukaryotes [143]. Previous studies have shown that two types of PSD can be found in fungi, but only one with a C2 domain [143]. Interestingly, the refined C2 domain HMMs were able to find two consecutive C2 domains in PSD. The presence of the first domain in PSD was not known before. More derived fungi lineages possess a single PSD with C2 domains, whereas basal fungi contain two (for an overview see table 3.5.1). Interestingly, PSDs with a single C2 domain were also found in *Capsaspora owczarzaki* and the slime molds *Dictyostelium discoideum* and *Dictyostelium purpureum*.

The occurrence of C2 domains in fungi TCB has been reported before, but the number of C2 domains vary between three [144] and six [34]. Indeed the TCBs analyzed in this study feature between three and five C2 domains (tcb_a, tcb_b, tcb_c, tcb_d, and tcb_e). It is assumed that these proteins have a role in membrane trafficking, but their exact function is yet unknown [144, 34]. *Pezizomycotina* and *Basidiomycota* contain a single TCB. *Schizosaccharomycetes* and most *Saccharomycotina* contain two TCB proteins, whereas some *Saccharomycotina* (e.g. *Candida albicans*, *Saccharomyces cerevisiae*) and most basal fungi possess three TCBs (for an overview see table 3.5.1). It has been reported before that TCB can also be found in animals and plants [34]. This could not be confirmed, since the refined fungi C2 HMMs were only able to find TCB in the basal animals *Nematostella vectensis* and *Hydra magnipapillata*, but not in other animal species. Additionally, conducted BLAST searches against the reported TCBs did not result in significant hits. Interestingly, the refined TCB HMMs were able to find TCB in the protists *Capsaspora owczarzaki* and *Encephalitozoon cuniculi*.

fUnc13 proteins contain a single C2 domain (f_unc13) that is embedded within a MUN domain [145]. Very likely, it has a role in membrane trafficking, potentially as priming or tethering factor [145]. Most fungi lineages posses a single fUnc13, but basal fungi and *Schizosaccharomycetes* contain two (for an overview see table 3.5.1). Interestingly, fUnc13 in *Batrachochytrium dendrobatidis* exhibits an additional C2 domain at the N-terminus. The fact that Unc13 homologs in animals contain two or three C2 domains respectively, might indicate that these proteins derived from a common ancestor.

Two types of PKCs can be distinguished with the refined HMM. The f_pkc_a C2 domain was predicted in fPKCa, while the f_pkc_b C2 domain was predicted in fPKCb. Both proteins have potential roles in signal transduction

processes. So far, only one PKC (fPKCa) was found in fungi [146]. Most fungi lineages contain fPKCa and fPKCb as singletons. Interestingly, fPKCb seems to have been lost in *Schizosaccharomycetes*. This loss must have been occurred independently, since fPKCb is present in other *Ascomycota* lineages (for an overview see table 3.5.1).

Similar to the PKCs, PLCs with C2 domains can also generally be divided into fPLCa (f_plca_a C2 domain) and fPLCb (f_plc_b C2 domain). These proteins are involved in regulating various cellular processes by hydrolyzing phosphatidylinositol 4,5-biphosphate to inositol 1,4,5-triphosphate [147, 148]. The occurrence of fPLCa has been reported before. The preliminary analysis of the animal C2 domains suggests that it might be an animal PLC-δ homolog [149]. fPLCa is present as a singleton in all fungi lineages. The presence of a second PLC (fPLCb) was not reported before. Most *Ascomycota* often feature two of these proteins, but fPLCb seems to be lost independently in *Schizosaccharomycetes*. Only a few *Basidiomycota* fungi contain fPLCb. With further refinement of the HMMs, it might be possible to identify more fPLCb C2 domains in *Basidiomycota*.

Additionally, the refined HMMs were able to identify C2 domains (fc2a and t1 C2 domain) in two proteins with unknown function. Predictions with Pfam [38] and SMART [39] do not show additional domains besides the C2 domain. The fc2a proteins were found as singletons in most fungi lineages. Only basal fungi and *Basidiomycota* possess more of these proteins (for an overview see table 3.5.1). The t1 protein on the other hand was found as singletons in basal fungi and *Basidiomycota*. In *Ascomycota*, only *Pezizomycotina* posses this protein, whereas *Saccharomycotina* and *Schizosaccharomycetes* seem to have lost t1 independently (for an overview see table 3.5.1).

Fig. 3.5-30 shows a phylogenetic reconstruction of the fungi C2 domains, contained in a representative set of 14 fungal species (i.e. *Ascomycota*, *Basid-*

iomycota, and basal fungi species). The C2 domains of the aforementioned proteins, highlighted by frames, split into distinct branches. Domains that fall not into these major branches are mostly C2 domains of basal fungi that were not found in more derived fungi. The tree shows the C2 domains of the two PKCs (f_pkc_a and f_pkc_b) in neighboring branches. This suggests that these domains are closely related and probably originated from a common ancestor. The same applies for the C2 domains of the two PLCs (f_plc_a and f_plc_b).

Three of the TCB C2 domains (tcb_a, tcb_c, and tcb_d) are grouped together in one bigger branch, whereas tcb_b and tcb_e, each on separate branches, seem to be more distant. Interestingly, the second C2 domain of TCB (tcb_b) seems to be closely related to the second C2 domain of MUG190 (mug190_b). A closer look revealed that the branch of the first C2 domain of TCB (tcb_a) also contains several mug190_a C2 domains of basal fungi species. Additionally, TCB and MUG190 both possess N-terminal TMRs (predicted with TMHMM [150]). This might indicate that both proteins derived from a common ancestor.

Overall, the refined HMMs were able to confirm several known C2 domains in fungi proteins (e.g fUnc13, fPLCa, fPKCa, NEDD4), but additionally they also found novel C2 domains that had not been reported before (e.g. fPKCb, fPLCa, MUG190). In general, basal fungi contain more proteins with C2 domains than organisms of more derived lineages. Mostly, these are duplications of the described proteins, but additional proteins with C2 domains can also be found in these organisms. Interestingly, those C2 domains are not present in more derived fungi. For example, the chytrids *Batrachochytrium dendrobatidis* and *Spizellomyces punctatus* contain the proteins Intersectin and RAS p21 protein activator. These proteins are not present in any other fungi, but can be found in other kingdoms. Intersectin plays a role in endocytic membrane trafficking [151] and RAS p21 protein

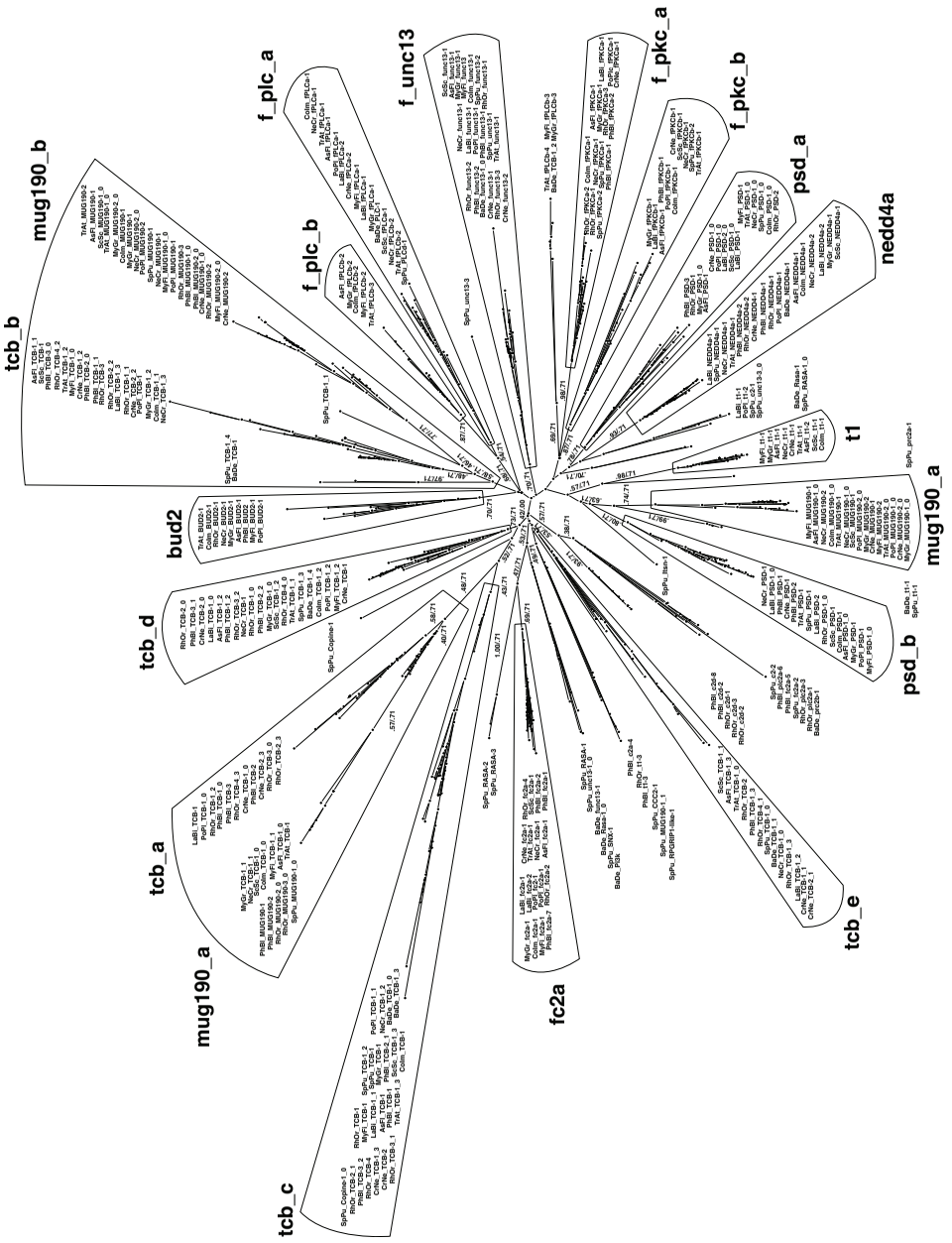

Figure 3.5-30: Unrooted phylogenetic C2 domain tree of 14 fungal species. Major domain groups are marked with their names. The labels on the major branches represent the Likelihood Mapping (left) and AU support values (right)

activator stimulates the GTPase activity of RAS p21 [152]. Additionally, *Spizellomyces punctatus* is the only fungi that possesses a Copine. Copines have two consecutive C2 domains and a C-terminal domain that might be important for protein-protein interaction [153]. The existence of proteins and domains in basal fungi, which cannot be found in more derived fungi lineages, indicates that the C2 domains in more derived fungi very likely do not represent the whole proto-eukaryotic repertoire of these domains. This phenomenon was observed already for fungi SNAREs, as Npsn was thought to be only present in plants [91] and protists [7], but could be found in the chytrids *Batrachochytrium dendrobatidis* and *Blastocladiella emersonii* as well. This indicates that Npsn was part of the assumed SNARE repertoire of the proto-eukaryotic ancestor, but was later lost in choanoflagellates, metazoa, and more derived fungi (see section 3.3).

For the thorough reconstruction of the evolutionary history of C2 domains it is indispensable to include species from other kingdoms into the analysis. This needs to be done very carefully, since the abundance of C2 domains in higher organisms ([154, 155, 156, 157, 158] and the C2 domain collection established during this thesis) render the computational task very challenging. As it has been shown for basal organisms of other kingdoms (e.g. animals, plants) that their SNARE repertoires are rather simple and that expansions mostly occur in more derived organisms [91, 88, 89, 7], it seem probable that comparable patterns will also be found for C2 domains. A possible strategy to analyze the C2 domain proteins would therefore be to focus on more basal organisms first. In addition, a preliminary inspection indicates that some lineages of protists do contain a limited number of C2 domain proteins as well (e.g. *Apicomplexa*, *Heterokonta*). Comparable to the strategy used for the analysis of the repertoire of fungi, these groups should thus be analyzed separately first.

A more formidable task is the analysis of the evolutionary history of the C2

domains in animals, as the repertoire appears to be expanded drastically, in particular in vertebrates.

4 Conclusion & Outlook

The *Tracey* management system is a flexible basis that facilitates the classification and analysis of the protein families involved in vesicular trafficking. In fact, with this system it is easily possible to investigate any domain or protein family. Although, the basic functionality of the *Tracey* management system is now implemented, maintenance and improvement will continue. The intention of the web interface was to be a universal tool for all project participants, but eventually also a public source for all classified domains. In its current version, no public section is implemented and this would therefore be the most urgent improvement to the web interface. Additionally, the main focus, so far, was set to functionality, but only little effort has been put into the design of the web interface. A more esthetic design would be more appealing and might thus extend the acceptance and usage of the public part of the web interface in the field.

The analyses of protein families involved in vesicular trafficking served as comprehensive tests of the novel management system. Although the conducted analyses showed that the system works well in principal, it is of course possible that it needs to be extended for additional domain or protein families. Our analyses revealed some interesting new facts about the evolutionary history of the investigated protein families. For example, the fungi SNARE set remained largely unchanged and only a little over twenty SNAREs could be found in most fungi. Multicellular members of other kingdoms (e.g. animals, plant) often feature much larger SNARE repertoires and therefore

the idea was promoted that the SNARE set expansion is generally linked to the rise of multicellularity [88, 9, 89, 90]. The fact that fungi developed multicellularity independently, but in contrast to plants and metazoa, do not exhibit a extended SNARE set suggests that the rise of multicellularity is not generally linked to an expansion of the SNARE repertoire. Another interesting finding was the discovery of Npsn in basal fungi. Before this study, Npsn was thought to be only present in plants [91] and protists [7], but its discovery in basal fungi suggests that Npsn was part of the assumed SNARE repertoire of the proto-eukaryotic ancestor, but was lost in animals and more derived fungi. It would be interesting to include additional basal fungi to shed more light at the species development closer to the root of this kingdom.

The analysis of the SNAP family uncovered that α- and γ-SNAP can be found in most eukaryotic lineages. This indicates that these two SNAP genes were already part of the assumed proto-eukaryotic SNAP repertoire. By contrast, β-SNAPs probably arose by duplication of α-SNAP in vertebrates. Interestingly, the inspected birds seemed to have lost α-SNAP. It is thought that all SNAPs mediate binding of NSF to the SNARE complex, in order to initiate SNARE complex disassembly. Regarding this, it would be of great importance to also investigate the evolutionary history of NSF, especially with respect to co-evolutionary aspects. NSF belongs to a subgroup of the AAA+ ATPase family. Preliminary studies have already been conducted to analyze the entire AAA+ ATPase family [159, 160]. These could serve as a starting point for a detailed analysis of NSF.

As another family, we started to investigate the evolutionary history of the C2 domains. This family is rather large and complex, therefore we focussed first on the C2 domains in fungi. The general fungi set is comprised of only about 12 different C2 domain proteins (BUD2, MUG190, NEDD4, PSD2, TCB, fUnc13, two PKCs, two PLCs, fc2a, and t1). A closer look revealed

that basal fungi often contain additional types of C2 domain proteins than more derived fungi. Remarkably, some of these types are not present in more derived fungi, but in animals (Intersectin, RAS p21 protein activator, and Copine). This suggests that the repertoire of the common ancestor of fungi and animals, maybe even all eukaryotes, most have been larger than that of higher fungi.

So far, it is not entirely clear whether the current models are sufficient enough for the analysis of all C2 domains. The next step is to refine the models to have highly sensitive and specific predictors that are able to identify all C2 domains. This task is very complex, since C2 domains are highly abundant in many species. By analyzing the C2 domains of mainly basal organisms from all eukaryotic kingdoms, it might be possible to eventually identify the C2 repertoire of the assumed proto-eukaryotic ancestor. Subsequently, these basic C2 domains could serve as a starting point for the refinement of the current models. This has already started, but the models are still not sensitive enough. Most of the known C2 domains can be detected already, but the current models still have difficulties finding certain types of C2 domains (e.g. Calcium activated protein for secretion, phosphatase and tensin homolog). These C2 domains appear to be more derived. Calcium activated protein for secretion proteins belong to a specific group of C2 domain proteins that contain Munc13-homology-domains [161]. Recently, it has been proposed that Munc13-homology-domains are only a part of a larger domain, called the MUN domain [145]. Interestingly, this domain shows weak, but significant sequence similarity to specific subunits of complexes involved in vesicle tethering [145]. Analyzing the MUN domain could complement the analysis of certain types of C2 domain proteins and subsequently provide deeper insights into the mechanisms of intracellular vesicle trafficking.

References

[1] J. S. Bonifacino and B. S. Glick, "The mechanisms of vesicle budding and fusion," vol. 116, pp. 153–66, Jan 2004.

[2] H. Stenmark, "Rab gtpases as coordinators of vesicle traffic," *Nat Rev Mol Cell Biol*, Jul 2009.

[3] R. Jahn and R. H. Scheller, "Snares–engines for membrane fusion," *Nat Rev Mol Cell Biol*, vol. 7, pp. 631–43, Sep 2006.

[4] R. F. G. Toonen and M. Verhage, "Vesicle trafficking: pleasure and pain from sm genes," *Trends Cell Biol*, vol. 13, pp. 177–86, Apr 2003.

[5] J. B. Pereira-Leal and M. C. Seabra, "Evolution of the rab family of small gtp-binding proteins," *J Mol Biol*, vol. 313, pp. 889–901, Nov 2001.

[6] V. L. Koumandou, J. B. Dacks, R. M. R. Coulson, and M. C. Field, "Control systems for membrane fusion in the ancestral eukaryote; evolution of tethering complexes and sm proteins," *BMC Evol Biol*, vol. 7, p. 29, Jan 2007.

[7] T. H. Kloepper, C. N. Kienle, and D. Fasshauer, "An elaborate classification of snare proteins sheds light on the conservation of the eukaryotic endomembrane system," *Mol Biol Cell*, Jun 2007.

[8] J. B. Dacks, P. P. Poon, and M. C. Field, "Phylogeny of endocytic components yields insight into the process of nonendosymbiotic organelle evolution," *Proc Natl Acad Sci USA*, vol. 105, pp. 588–93, Jan 2008.

[9] J. B. Dacks and M. C. Field, "Evolution of the eukaryotic membrane-trafficking system: origin, tempo and mode," *J Cell Sci*, vol. 120, pp. 2977–85, Sep 2007.

[10] H. Cai, K. Reinisch, and S. Ferro-Novick, "Coats, tethers, rabs, and snares work together to mediate the intracellular destination of a transport vesicle," *Dev Cell*, vol. 12, pp. 671–82, May 2007.

[11] C. Gurkan, A. Koulov, and W. Balch, "An evolutionary perspective on eukaryotic membrane trafficking," *Origins and Evolution of Eukaryotic Endomembranes and ...*, Jan 2006.

[12] W. Hong, "Snares and traffic," vol. 1744, pp. 493–517, Jul 2005.

[13] J. B. Bock, H. T. Matern, and A. A. Peden, "A genomic perspective on membrane compartment organization," vol. 409, pp. 839–41, Feb 2001.

[14] D. Fasshauer, R. B. Sutton, A. T. Brunger, and R. Jahn, "Conserved structural features of the synaptic fusion complex: Snare proteins reclassified as q- and r-snares," *Proc Natl Acad Sci USA*, vol. 95, pp. 15781–6, Dec 1998.

[15] T. Weimbs, S. H. Low, S. J. Chapin, K. E. Mostov, P. Bucher, and K. Hofmann, "A conserved domain is present in different families of vesicular fusion proteins: a new superfamily," vol. 94, pp. 3046–51, Apr 1997.

[16] M. Veit, T. H. Söllner, and J. E. Rothman, "Multiple palmitoylation of synaptotagmin and the t-snare snap-25," *FEBS Lett*, vol. 385, pp. 119–23, Apr 1996.

[17] J. A. McNew, M. Sogaard, N. M. Lampen, S. Machida, R. R. Ye, L. Lacomis, P. Tempst, J. E. Rothman, and T. H. Söllner, "Ykt6p, a prenylated snare essential for endoplasmic reticulum-golgi transport," *J Biol Chem*, vol. 272, pp. 17776–83, Jul 1997.

[18] D. Zwilling, A. Cypionka, W. H. Pohl, D. Fasshauer, P. J. Walla, M. C. Wahl, and R. Jahn, "Early endosomal snares form a structurally conserved snare complex and fuse liposomes with multiple topologies," *EMBO J*, vol. 26, pp. 9–18, Jan 2007.

[19] W. Antonin, D. Fasshauer, S. Becker, R. Jahn, and T. R. Schneider, "Crystal structure of the endosomal snare complex reveals common structural principles of all snares," *Nat Struct Biol*, vol. 9, pp. 107–11, Feb 2002.

[20] R. B. Sutton, D. Fasshauer, R. Jahn, and A. T. Brunger, "Crystal structure of a snare complex involved in synaptic exocytosis at 2.4 a resolution," *Nature*, vol. 395, pp. 347–53, Sep 1998.

[21] K. Wiederhold and D. Fasshauer, "Is assembly of the snare complex enough to fuel membrane fusion?," *J Biol Chem*, vol. 284, pp. 13143–52, May 2009.

[22] D. Fasshauer, W. Antonin, V. Subramaniam, and R. Jahn, "Snare assembly and disassembly exhibit a pronounced hysteresis," *Nat Struct Biol*, vol. 9, pp. 144–51, Feb 2002.

[23] P. I. Hanson and S. W. Whiteheart, "Aaa+ proteins: have engine, will work," *Nat Rev Mol Cell Biol*, vol. 6, pp. 519–29, Jul 2005.

[24] C. Wimmer, T. M. Hohl, C. A. Hughes, S. A. Müller, T. H. Söllner, A. Engel, and J. E. Rothman, "Molecular mass, stoichiometry, and assembly of 20 s particles," *J Biol Chem*, vol. 276, pp. 29091–7, Aug 2001.

[25] D. O. Clary, I. C. Griff, and J. E. Rothman, "Snaps, a family of nsf attachment proteins involved in intracellular membrane fusion in animals and yeast," *Cell*, vol. 61, pp. 709–21, May 1990.

[26] J. Furst, R. B. Sutton, J. Chen, A. T. Brunger, and N. Grigorieff, "Electron cryomicroscopy structure of n-ethyl maleimide sensitive factor at 11 a resolution," *EMBO J*, vol. 22, pp. 4365–74, Sep 2003.

[27] T. M. Hohl, F. Parlati, C. Wimmer, J. E. Rothman, T. H. Söllner, and H. Engelhardt, "Arrangement of subunits in 20 s particles consisting of nsf, snaps, and snare complexes," *Mol Cell*, vol. 2, pp. 539–48, Nov 1998.

[28] K. G. Fleming, T. M. Hohl, R. C. Yu, S. A. Müller, B. Wolpensinger, A. Engel, H. Engelhardt, A. T. Brünger, T. H. Söllner, and P. I. Hanson, "A revised model for the oligomeric state of the n-ethylmaleimide-sensitive fusion protein, nsf," *J Biol Chem*, vol. 273, pp. 15675–81, Jun 1998.

[29] P. I. Hanson, R. Roth, H. Morisaki, R. Jahn, and J. E. Heuser, "Structure and conformational changes in nsf and its membrane receptor complexes visualized by quick-freeze/deep-etch electron microscopy," *Cell*, vol. 90, pp. 523–35, Aug 1997.

[30] L. Coussens, P. J. Parker, L. Rhee, T. L. Yang-Feng, E. Chen, M. D. Waterfield, U. Francke, and A. Ullrich, "Multiple, distinct forms of bovine and human protein kinase c suggest diversity in cellular signaling pathways," *Science*, vol. 233, pp. 859–66, Aug 1986.

[31] D. Murray and B. Honig, "Electrostatic control of the membrane targeting of c2 domains," *Mol Cell*, vol. 9, pp. 145–54, Jan 2002.

[32] W. Cho and R. V. Stahelin, "Membrane binding and subcellular targeting of c2 domains," *Biochim Biophys Acta*, vol. 1761, pp. 838–49, Aug 2006.

[33] E. A. Nalefski and J. J. Falke, "The c2 domain calcium-binding motif: structural and functional diversity," *Protein Sci*, vol. 5, pp. 2375–90, Dec 1996.

[34] J. L. Jiménez and B. Davletov, "Beta-strand recombination in tricalbin evolution and the origin of synaptotagmin-like c2 domains," *Proteins*, vol. 68, pp. 770–8, Aug 2007.

[35] J. Pereira-Leal, "The ypt/rab family and the evolution of trafficking in fungi," *Traffic*, Nov 2007.

[36] A. C. Yoshizawa, S. Kawashima, S. Okuda, M. Fujita, M. Itoh, Y. Moriya, M. Hattori, and M. Kanehisa, "Extracting sequence motifs and the phylogenetic features of snare-dependent membrane traffic," *Traffic*, vol. 7, pp. 1104–18, Aug 2006.

[37] J. Dacks and W. Doolittle, "Molecular and phylogenetic characterization of syntaxin genes from parasitic protozoa," *Mol Biochem Parasitol*, Jan 2004.

[38] R. D. Finn, J. Mistry, J. Tate, P. Coggill, A. Heger, J. E. Pollington, O. L. Gavin, P. Gunasekaran, G. Ceric, K. Forslund, L. Holm, E. L. L. Sonnhammer, S. R. Eddy, and A. Bateman, "The pfam protein families database," *Nucleic Acids Res*, vol. 38, pp. D211–22, Jan 2010.

[39] I. Letunic, T. Doerks, and P. Bork, "Smart 6: recent updates and new developments," *Nucleic Acids Res*, vol. 37, pp. D229–32, Jan 2009.

[40] N. Kienle, "Building a Management System for the *SNARE-Project*," Master's thesis, University of Tübingen, Germany, 2006.

[41] "Java." `http://java.sun.com`.

[42] "NCBI - National Center for Biotechnology Information." `http://www.ncbi.nlm.nih.gov`.

[43] N. Kienle, T. H. Kloepper, and D. Fasshauer, "Differences in the snare evolution of fungi and metazoa," *Biochem Soc Trans*, vol. 37, pp. 787–91, Aug 2009.

[44] H. R. Pelham, "Snares and the specificity of membrane fusion," *Trends Cell Biol*, vol. 11, pp. 99–101, Mar 2001.

[45] D. K. Banfield, "Snare complexes–is there sufficient complexity for vesicle targeting specificity?," *Trends Biochem Sci*, vol. 26, pp. 67–8, Jan 2001.

[46] "MySQL." `http://www.mysql.com`.

[47] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: A novel method for fast and accurate multiple sequence alignment," *J Mol Biol*, vol. 302, pp. 205–17, Sep 2000.

[48] C. B. Do, M. S. P. Mahabhashyam, M. Brudno, and S. Batzoglou, "Probcons: Probabilistic consistency-based multiple sequence alignment," *Genome Res*, vol. 15, pp. 330–40, Feb 2005.

[49] K. Katoh, K. ichi Kuma, H. Toh, and T. Miyata, "Mafft version 5: improvement in accuracy of multiple sequence alignment," *Nucleic Acids Res*, vol. 33, pp. 511–8, Jan 2005.

[50] R. C. Edgar and S. Batzoglou, "Multiple sequence alignment," *Curr Opin Struct Biol*, vol. 16, pp. 368–73, Jun 2006.

[51] R. C. Edgar, "Muscle: multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Res*, vol. 32, pp. 1792–7, Jan 2004.

[52] R. C. Edgar, "Muscle: a multiple sequence alignment method with reduced time and space complexity," *BMC Bioinformatics*, vol. 5, p. 113, Aug 2004.

[53] M. Kimura, *The Neutral Theory of Molecular Evolution*. Cambridge Unversity Press, 1983.

[54] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[55] S. Eddy, *HMMER User's Guide*, 2.3.2 ed., 2003.

[56] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic Local Alignment Search Tool," *J. Mol. Biol*, vol. 215, no. 3, pp. 1073–1082, 1990.

[57] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proc. Natl. Acad. Sci. USA*, vol. 85, pp. 2444–2448, 1988.

[58] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic Acids Res*, vol. 25, pp. 3389–402, Sep 1997.

[59] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc Natl Acad Sci USA*, vol. 89, pp. 10915–9, Nov 1992.

[60] M. Gerstein, E. L. Sonnhammer, and C. Chothia, "Volume changes in protein evolution," *J Mol Biol*, vol. 236, pp. 1067–78, Mar 1994.

[61] A. Krogh and G. Mitchison, "Maximum entropy weighting of aligned sequences of proteins or DNA," *Proc Int Conf Intell Syst Mol Biol.*, pp. 215–221, 195.

[62] S. Henikoff and J. G. Henikoff, "Position-based sequence weights," *J Mol Biol*, vol. 243, pp. 574–8, Nov 1994.

[63] P. R. Sibbald and P. Argos, "Weighting aligned protein or nucleic acid sequences to correct for unequal representation," *J Mol Biol.*, vol. 216, no. 4, 1990.

[64] L. S. Vinh and A. V. Haeseler, "Iqpnni: moving fast through tree space and stopping in time," *Mol Biol Evol*, vol. 21, pp. 1565–71, Aug 2004.

[65] O. Gascuel, "Bionj: an improved version of the nj algorithm based on a simple model of sequence data," *Mol Biol Evol*, vol. 14, pp. 685–95, Jul 1997.

[66] S. Guindon and O. Gascuel, "A simple, fast, and accurate method to estimate large phylogenies by maximum likelihood," *Syst Biol*, Jan 2003.

[67] K. Strimmer and A. von Haeseler, "Likelihood-mapping: a simple method to visualize phylogenetic content of a sequence alignment," *Proc Natl Acad Sci USA*, vol. 94, pp. 6815–9, Jun 1997.

[68] J. Felsenstein, "Phylip - phylogeny inference package (version 3.2)," *Cladistics*, vol. 5, 1998.

[69] M. Sanderson and M. Wojciechowski, "Improved bootstrap confidence limits in large-scale phylogenies, with an example from neo-astragalus (leguminosae)," *Syst Biol*, vol. 49, pp. 671–685, Dec 2000.

[70] J. Felsenstein and H. Kishino, "Is there something wrong with the bootstrap on phylogenies? a reply to hillis and bull," *Syst Biol*, Jan 1993.

[71] D. Hillis and J. Bull, "An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis," *Syst Biol*, Jan 1993.

[72] A. Zharkikh and W. Lit, "Statistical properties of bootstrap estimation of phylogenetic variability from nucleotide sequences: Ii. four taxa without a molecular clock," *Journal of molecular evolution*, Jan 1992.

[73] H. Shimodaira, "An approximately unbiased test of phylogenetic tree selection," *Syst Biol*, vol. 51, pp. 492–508, Jan 2002.

[74] D. Jones, W. Taylor, and J. Thornton, "The rapid generation of mutation data matrices from protein sequences," *Bioinformatics*, Jan 1992.

[75] H. Shimodaira and M. Hasegawa, "Consel: for assessing the confidence of phylogenetic tree selection," *Bioinformatics*, vol. 17, pp. 1246–7, Dec 2001.

[76] J. Pei and N. V. Grishin, "Al2co: calculation of positional conservation in a protein sequence alignment," *Bioinformatics*, vol. 17, pp. 700–12, Aug 2001.

[77] "XML." `http://www.w3.org/TR/REC-xml`.

[78] "NCBI FTP server." `ftp://ftp.ncbi.nih.gov/pub/taxonomy`.

[79] "Apache Commons." `http://commons.apache.org`.

[80] "GridGain." `http://www.gridgain.com`.

[81] "DOE Joint Genome Institute." `http://www.jgi.doe.gov`.

[82] "Baylor College of Medicine." `http://www.bcm.edu`.

[83] "J. Craig Venter Institute." `http://www.jcvi.org`.

[84] "BROAD Institute." `http://www.broadinstitute.org`.

[85] "Fungal Genomics Project." `https://fungalgenomics.concordia.ca/home`.

[86] E. A. O'Brien, L. B. Koski, Y. Zhang, L. Yang, E. Wang, M. W. Gray, G. Burger, and B. F. Lang, "Tbestdb: a taxonomically broad database of expressed sequence tags (ests)," *Nucleic Acids Res*, vol. 35, pp. D445–51, Jan 2007.

[87] N. Kienle, T. Kloepper, and D. Fasshauer, "Phylogeny of the snare vesicle fusion machinery yields insights into the conservation of the secretory pathway in fungi," *BMC Evol Biol*, vol. 9, p. 19, Jan 2009.

[88] T. Kloepper, C. Kienle, and D. Fasshauer, "Snareing the basis of multicellularity: Consequences of protein family expansion during evolution," *Mol Biol Evol*, Jul 2008.

[89] A. Sanderfoot, "Increases in the number of snare genes parallels the rise of multicellularity among the green plants," *Plant Physiol*, vol. 144, pp. 6–17, May 2007.

[90] J. B. Dacks and W. F. Doolittle, "Molecular and phylogenetic characterization of syntaxin genes from parasitic protozoa," *Mol Biochem Parasitol*, vol. 136, pp. 123–36, Aug 2004.

[91] A. A. Sanderfoot, F. F. Assaad, and N. V. Raikhel, "The arabidopsis genome. an abundance of soluble n-ethylmaleimide-sensitive factor adaptor protein receptors," *Plant Physiol*, vol. 124, pp. 1558–69, Dec 2000.

[92] T. Y. James, F. Kauff, C. L. Schoch, P. B. Matheny, V. Hofstetter, C. J. Cox, G. Celio, C. Gueidan, E. Fraker, J. Miadlikowska, H. T. Lumbsch, A. Rauhut, V. Reeb, A. E. Arnold, A. Amtoft, J. E. Stajich, K. Hosaka, G.-H. Sung, D. Johnson, B. O'Rourke, M. Crockett, M. Binder, J. M. Curtis, J. C. Slot, Z. Wang, A. W. Wilson, A. Schüssler, J. E. Longcore, K. O'Donnell, S. Mozley-Standridge, D. Porter, P. M. Letcher, M. J. Powell, J. W. Taylor, M. M. White, G. W. Griffith, D. R. Davies, R. A. Humber, J. B. Morton, J. Sugiyama, A. Y. Rossman, J. D. Rogers, D. H. Pfister, D. Hewitt, K. Hansen, S. Hambleton, R. A. Shoemaker, J. Kohlmeyer, B. Volkmann-Kohlmeyer, R. A. Spotts, M. Serdani, P. W. Crous, K. W. Hughes, K. Matsuura, E. Langer, G. Langer, W. A. Untereiner, R. Lücking, B. Büdel, D. M. Geiser, A. Aptroot, P. Diederich, I. Schmitt, M. Schultz, R. Yahr, D. S. Hibbett, F. Lutzoni, D. J. McLaughlin, J. W. Spatafora, and R. Vilgalys, "Reconstructing the early evolution of fungi using a six-gene phylogeny," *Nature*, vol. 443, pp. 818–22, Oct 2006.

[93] T. K. Sato, T. Darsow, and S. D. Emr, "Vam7p, a snap-25-like molecule, and vam3p, a syntaxin homolog, function together in yeast vacuolar protein trafficking," *Mol Cell Biol*, vol. 18, pp. 5308–19, Sep 1998.

[94] C. Ungermann and W. Wickner, "Vam7p, a vacuolar snap-25 homolog, is required for snare complex integrity and vacuole docking and fusion," *EMBO J*, vol. 17, pp. 3269–76, Jun 1998.

[95] Y. Wada, Y. Ohsumi, and Y. Anraku, "Genes for directing vacuolar morphogenesis in saccharomyces cerevisiae. i. isolation and characterization of two classes of vam mutants," *J Biol Chem*, vol. 267, pp. 18665–70, Sep 1992.

[96] Y. Wada and Y. Anraku, "Genes for directing vacuolar morphogenesis in saccharomyces cerevisiae. ii. vam7, a gene for regulating morphogenic assembly of the vacuoles," *J Biol Chem*, vol. 267, pp. 18671–5, Sep 1992.

[97] M. L. Cheever, T. K. Sato, T. de Beer, T. G. Kutateladze, S. D. Emr, and M. Overduin, "Phox domain interaction with ptdins(3)p targets the vam7 t-snare to vacuole membranes," *Nat Cell Biol*, vol. 3, pp. 613–8, Jul 2001.

[98] C. W. Ostrowicz, C. T. A. Meiringer, and C. Ungermann, "Yeast vacuole fusion: a model system for eukaryotic endomembrane dynamics," *Autophagy*, vol. 4, pp. 5–19, Jan 2008.

[99] W. Wickner, "Yeast vacuoles and membrane fusion pathways," *EMBO J*, vol. 21, pp. 1241–7, Mar 2002.

[100] G. D. Gupta and I. B. Heath, "Predicting the distribution, conservation, and functions of snares and related proteins in fungi," *Fungal Genet Biol*, vol. 36, pp. 1–21, Jun 2002.

[101] A. Fagarasanu and R. A. Rachubinski, "Orchestrating organelle inheritance in saccharomyces cerevisiae," *Curr Opin Microbiol*, vol. 10, pp. 528–38, Dec 2007.

[102] L. S. Weisman, "Organelles on the move: insights from yeast vacuole inheritance," *Nat Rev Mol Cell Biol*, vol. 7, pp. 243–52, Apr 2006.

[103] L. S. Weisman, R. Bacallao, and W. Wickner, "Multiple methods of visualizing the yeast vacuole permit evaluation of its morphology and inheritance during the cell cycle," *J Cell Biol*, vol. 105, pp. 1539–47, Oct 1987.

[104] F. Filippini, V. Rossi, T. Galli, A. Budillon, M. D'Urso, and M. D'Esposito, "Longins: a new evolutionary conserved vamp family sharing a novel snare domain," *Trends Biochem Sci*, vol. 26, pp. 407–9, Jul 2001.

[105] V. Rossi, D. K. Banfield, M. Vacca, L. E. P. Dietrich, C. Ungermann, M. D'Esposito, T. Galli, and F. Filippini, "Longins and their longin domains: regulated snares and multifunctional snare regulators," *Trends Biochem Sci*, vol. 29, pp. 682–8, Dec 2004.

[106] W. Wen, L. Chen, H. Wu, X. Sun, M. Zhang, and D. K. Banfield, "Identification of the yeast r-snare nyv1p as a novel longin domain-containing protein," *Mol Biol Cell*, vol. 17, pp. 4282–99, Oct 2006.

[107] L. C. Gonzalez, W. I. Weis, and R. H. Scheller, "A novel snare n-terminal domain revealed by the crystal structure of sec22b," *J Biol Chem*, vol. 276, pp. 24203–11, Jun 2001.

[108] H. Tochio, M. M. Tsui, D. K. Banfield, and M. Zhang, "An autoinhibitory mechanism for nonsyntaxin snare proteins revealed by the structure of ykt6p," *Science*, vol. 293, pp. 698–702, Jul 2001.

[109] D. R. Scannell, G. Butler, and K. H. Wolfe, "Yeast genome evolution–the origin of the species," *Yeast*, vol. 24, pp. 929–42, Nov 2007.

[110] B. Dujon, "Yeasts illustrate the molecular mechanisms of eukaryotic genome evolution," *Trends Genet*, vol. 22, pp. 375–87, Jul 2006.

[111] A. M. Neiman, "Prospore membrane formation defines a developmentally regulated branch of the secretory pathway in yeast," *J Cell Biol*, vol. 140, pp. 29–37, Jan 1998.

[112] A. M. Neiman, "Ascospore formation in the yeast saccharomyces cerevisiae," *Microbiol Mol Biol Rev*, vol. 69, pp. 565–84, Dec 2005.

[113] A. M. Neiman, L. Katz, and P. J. Brennwald, "Identification of domains required for developmentally regulated snare function in saccharomyces cerevisiae," *Genetics*, vol. 155, pp. 1643–55, Aug 2000.

[114] J. Jäntti, M. K. Aalto, M. Oyen, L. Sundqvist, S. Keränen, and H. Ronne, "Characterization of temperature-sensitive mutations in the yeast syntaxin 1 homologues sso1p and sso2p, and evidence of a distinct function for sso1p in sporulation," *J Cell Sci*, vol. 115, pp. 409–20, Jan 2002.

[115] M. Oyen, J. Jäntti, S. Keränen, and H. Ronne, "Mapping of sporulation-specific functions in the yeast syntaxin gene sso1," *Curr Genet*, vol. 45, pp. 76–82, Feb 2004.

[116] D. A. Hattendorf, A. Andreeva, A. Gangar, P. J. Brennwald, and W. I. Weis, "Structure of the yeast polarity protein sro7 reveals a snare regulatory mechanism," *Nature*, vol. 446, pp. 567–71, Mar 2007.

[117] A. V. Pobbati, A. Razeto, M. Böddener, S. Becker, and D. Fasshauer, "Structural basis for the inhibitory role of tomosyn in exocytosis," *J Biol Chem*, vol. 279, pp. 47192–200, Nov 2004.

[118] E. S. Masuda, B. C. Huang, J. M. Fisher, Y. Luo, and R. H. Scheller, "Tomosyn binds t-snare proteins via a vamp-like coiled coil," *Neuron*, vol. 21, pp. 479–80, Sep 1998.

[119] Y. Fujita, H. Shirataki, T. Sakisaka, T. Asakura, T. Ohya, H. Kotani, S. Yokoyama, H. Nishioka, Y. Matsuura, A. Mizoguchi, R. H. Scheller, and Y. Takai, "Tomosyn: a syntaxin-1-binding protein that forms a novel complex in the neurotransmitter release process," *Neuron*, vol. 20, pp. 905–15, May 1998.

[120] M. Kagami, A. Toh-e, and Y. Matsui, "Sro7p, a saccharomyces cerevisiae counterpart of the tumor suppressor l(2)gl protein, is related to myosins in function," *Genetics*, vol. 149, pp. 1717–27, Aug 1998.

[121] F. Wirtz-Peitz and J. A. Knoblich, "Lethal giant larvae take on a life of their own," *Trends Cell Biol*, vol. 16, pp. 234–41, May 2006.

[122] V. Vasioukhin, "Lethal giant puzzle of lgl," *Dev Neurosci*, vol. 28, pp. 13–24, Jan 2006.

[123] D. Fasshauer and R. Jahn, "Budding insights on cell polarity," *Nat Struct Mol Biol*, vol. 14, pp. 360–2, May 2007.

[124] G. D. Gupta, S. J. Free, N. N. Levina, S. Keränen, and I. B. Heath, "Two divergent plasma membrane syntaxin-like snares, nsyn1 and nsyn2, contribute to hyphal tip growth and other developmental processes in neurospora crassa," *Fungal Genet Biol*, vol. 40, pp. 271–86, Dec 2003.

[125] M. Valkonen, E. R. Kalkman, M. Saloheimo, M. Penttilä, N. D. Read, and R. R. Duncan, "Spatially segregated snare protein interactions in living fungal cells," *J Biol Chem*, vol. 282, pp. 22775–85, Aug 2007.

[126] D. A. Fitzpatrick, M. E. Logue, J. E. Stajich, and G. Butler, "A fungal phylogeny based on 42 complete genomes derived from supertree and combined gene analysis," *BMC Evol Biol*, vol. 6, p. 99, Jan 2006.

[127] J. E. Galagan, M. R. Henn, L.-J. Ma, C. A. Cuomo, and B. Birren, "Genomics of the fungal kingdom: insights into eukaryotic biology," *Genome Res*, vol. 15, pp. 1620–31, Dec 2005.

[128] C. P. Kurtzman, "Phylogenetic circumscription of saccharomyces, kluyveromyces and other members of the saccharomycetaceae, and the proposal of the new genera lachancea, nakaseomyces, naumovia, vanderwaltozyma and zygotorulaspora," *FEMS Yeast Res*, vol. 4, pp. 233–45, Dec 2003.

[129] S.-O. Suh, M. Blackwell, C. P. Kurtzman, and M.-A. Lachance, "Phylogenetics of saccharomycetales, the ascomycete yeasts," *Mycologia*, vol. 98, pp. 1006–17, Jan 2006.

[130] S. Diezmann, C. J. Cox, G. Schönian, R. J. Vilgalys, and T. G. Mitchell, "Phylogeny and evolution of medical species of candida and related taxa: a multigenic analysis," *J Clin Microbiol*, vol. 42, pp. 5624–35, Dec 2004.

[131] T. Ohama, T. Suzuki, M. Mori, S. Osawa, T. Ueda, K. Watanabe, and T. Nakase, "Non-universal decoding of the leucine codon cug in several candida species.," *Nucleic Acids Res*, vol. 21, p. 4039, Aug 1993.

[132] Y. Kawaguchi, H. Honda, J. Taniguchi-Morimura, and S. Iwasaki, "The codon cug is read as serine in an asporogenic yeast candida cylindracea," *Nature*, vol. 341, pp. 164–6, Sep 1989.

[133] T. Ohama, T. Suzuki, M. Mori, S. Osawa, T. Ueda, K. Watanabe, and T. Nakase, "Non-universal decoding of the leucine codon cug in several candida species," *Nucleic Acids Res*, vol. 21, pp. 4039–45, Aug 1993.

[134] P. Novick, C. Field, and R. Schekman, "Identification of 23 complementation groups required for post-translational events in the yeast secretory pathway," *Cell*, vol. 21, pp. 205–15, Aug 1980.

[135] L. M. Rice and A. T. Brunger, "Crystal structure of the vesicular transport protein sec17: implications for snap function in snare complex disassembly," *Mol Cell*, vol. 4, pp. 85–95, Jul 1999.

[136] E. Bitto, C. A. Bingman, D. A. Kondrashov, J. G. McCoy, R. M. Bannen, G. E. Wesenberg, and G. N. Phillips, "Structure and dynamics of gamma-snap: insight into flexibility of proteins from the snap family," *Proteins*, vol. 70, pp. 93–104, Jan 2008.

[137] S. W. Whiteheart, I. C. Griff, M. Brunner, D. O. Clary, T. Mayer, S. A. Buhrow, and J. E. Rothman, "Snap family of nsf attachment proteins includes a brain-specific isoform," *Nature*, vol. 362, pp. 353–5, Mar 1993.

[138] E. R. Tulman, C. L. Afonso, Z. Lu, L. Zsak, G. F. Kutish, and D. L. Rock, "The genome of canarypox virus," *J Virol*, vol. 78, pp. 353–66, Jan 2004.

[139] S. M. Laidlaw, M. A. Anwar, W. Thomas, P. Green, K. Shaw, and M. A. Skinner, "Fowlpox virus encodes nonessential homologs of cellular alpha-snap, pc-1, and an orphan human homolog of a secreted nematode protein," *J Virol*, vol. 72, pp. 6742–51, Aug 1998.

[140] H. O. Park, J. Chant, and I. Herskowitz, "Bud2 encodes a gtpase-activating protein for bud1/rsr1 necessary for proper bud-site selection in yeast," *Nature*, vol. 365, pp. 269–74, Sep 1993.

[141] C. Martín-Castellanos, M. Blanco, A. E. Rozalén, L. Pérez-Hidalgo, A. I. García, F. Conde, J. Mata, C. Ellermeier, L. Davis, P. San-Segundo, G. R. Smith, and S. Moreno, "A large-scale screen in s. pombe identifies seven novel genes required for critical meiotic events," *Curr Biol*, vol. 15, pp. 2056–62, Nov 2005.

[142] P. Kaliszewski and T. Zoładek, "The role of rsp5 ubiquitin ligase in regulation of diverse processes in yeast cells," *Acta Biochim Pol*, vol. 55, pp. 649–62, Jan 2008.

[143] D. Voelker, "Phosphatidylserine decarboxylase," *Biochimica et Biophysica Acta (BBA)-Lipids and Lipid ...*, Jan 1997.

[144] C. E. Creutz, S. L. Snyder, and T. A. Schulz, "Characterization of the yeast tricalbins: membrane-bound multi-c2-domain proteins that form complexes involved in membrane trafficking," *Cell Mol Life Sci*, vol. 61, pp. 1208–20, May 2004.

[145] J. Pei, C. Ma, J. Rizo, and N. Grishin, "Remote homology between munc13 mun domain and vesicle tethering complexes," *J Mol Biol*, Jun 2009.

[146] R. H. Palmer, J. Ridden, and P. J. Parker, "Cloning and expression patterns of two members of a novel protein-kinase-c-related kinase family," *Eur J Biochem*, vol. 227, pp. 344–51, Jan 1995.

[147] T. Yoko-o, H. Kato, Y. Matsui, and T. Takenawa, "Isolation and characterization of temperature-sensitiveplc1 mutants of the yeastsaccharomyces cerevisiae," *Molecular and General ...*, Jan 1995.

[148] J. S. Flick and J. Thorner, "An essential function of a phosphoinositide-specific phospholipase c is relieved by inhibition of a cyclin-dependent protein kinase in the yeast saccharomyces cerevisiae," *Genetics*, vol. 148, pp. 33–47, Jan 1998.

[149] M. J. Rebecchi and S. N. Pentyala, "Structure, function, and control of phosphoinositide-specific phospholipase c," *Physiol Rev*, vol. 80, pp. 1291–335, Oct 2000.

[150] A. Krogh, B. Larsson, G. von Heijne, and E. L. Sonnhammer, "Predicting transmembrane protein topology with a hidden markov model: application to complete genomes," *J Mol Biol*, vol. 305, pp. 567–80, Jan 2001.

[151] A. Pechstein, O. Shupliakov, and V. Haucke, "Intersectin 1: a versatile actor in the synaptic vesicle cycle," *Biochem Soc Trans*, vol. 38, pp. 181–6, Feb 2010.

[152] A. Wittinghofer, K. Scheffzek, and M. R. Ahmadian, "The interaction of ras with gtpase-activating proteins," *FEBS Lett*, vol. 410, pp. 63–7, Jun 1997.

[153] J. L. Tomsig and C. E. Creutz, "Copines: a ubiquitous family of ca(2+)-dependent phospholipid-binding proteins," *Cell Mol Life Sci*, vol. 59, pp. 1467–77, Sep 2002.

[154] M. Craxton, "A manual collection of syt, esyt, rph3a, rph3al, doc2, and dblc2 genes from 46 metazoan genomes–an open access resource for neuroscience and evolutionary biology," *BMC Genomics*, vol. 11, p. 37, Jan 2010.

[155] J. Kopka, C. Pical, A. M. Hetherington, and B. Müller-Röber, "Ca2+/phospholipid-binding (c2) domain in multiple plant proteins: novel components of the calcium-sensing apparatus," *Plant Mol Biol*, vol. 36, pp. 627–37, Mar 1998.

[156] S. Martens and H. T. McMahon, "Mechanisms of membrane fusion: disparate players and common principles," *Nat Rev Mol Cell Biol*, vol. 9, pp. 543–56, Jul 2008.

[157] M. Craxton, "Evolutionary genomics of plant genes encoding n-terminal-tm-c2 domain proteins and the similar fam62 genes and synaptotagmin genes of metazoans," *BMC Genomics*, vol. 8, p. 259, Jul 2007.

[158] M. Craxton, "Synaptotagmin gene content of the sequenced genomes," *BMC Genomics*, vol. 5, p. 43, Jul 2004.

[159] T. Frickey and A. N. Lupas, "Phylogenetic analysis of aaa proteins," *J Struct Biol*, vol. 146, pp. 2–10, Jan 2004.

[160] L. M. Iyer, D. D. Leipe, E. V. Koonin, and L. Aravind, "Evolutionary history and higher order classification of aaa+ atpases," *J Struct Biol*, vol. 146, pp. 11–31, Jan 2004.

[161] H. Koch, K. Hofmann, and N. Brose, "Definition of munc13-homology-domains and characterization of a novel ubiquitously expressed munc13 isoform," *Biochem J*, vol. 349, pp. 247–53, Jul 2000.

A Contributions

Aligning Sequences and *Tracey*

All components of the Aligning Sequences (3.1), *Tracey* Database (3.2.1), and the *Tracey* Java Database Package (3.2.2) sections were designed and implemented in collaboration with Tobias H. Kloepper.

Web Access Manager

The Web Access Manager (3.2.4) was designed in collaboration with and implemented by Tobias H. Kloepper.

SNARE Proteins in Fungi

Chapter 3.3 SNARE Proteins in Fungi is based on the publication
Phylogeny of the SNARE vesicle fusion machinery yields insights into the conservation of the secretory pathway in fungi,
Kienle N, Kloepper H. Tobias, and Fasshauer Dirk,
BMC Evol Biol. 2009 Jan 23;9:19.

Phylogenetic Trees

The actual calculations of all phylogenetic trees were conducted by Tobias H. Kloepper.

Other Contributions

Figure 1.1-4 was kindly provided by Anand Radhakrishnan.

Curriculum Vitae

Personal Information

Name	Carl Nickias Kienle
Address	Theaterstrasse 26, 37073 Göttingen
Telephone	+49 - 551 - 281 3893
E-Mail	nkienle@gwdg.de
Date of Birth	22.08.78
Place of Birth	Tübingen, Germany
Nationality	German
Marital Status	not married

Research Experience

since Dec. 2006	Max-Planck-Institute for biophysical Chemistry, Goettingen
Project	Phylogenetic studies of the vesicular fusion machinery
Supervisor	Dr. Dirk Fasshauer
Position	PhD student
Oct. 2005 - Nov. 2006	Eberhard Karls University, Tübingen
Project	Building a Management System for the SNARE-Project
Supervisor	Prof. Dr. Daniel Huson & Dr. Dirk Fasshauer
Position	Diploma student

University Studies

Oct. 2000 - Nov. 2006	Eberhard Karls University, Tübingen
Subject	Bioinformatics

Apprenticeship

1998 - 2000	Apprenticeship as a butcher in the butcher shop Egeler in Ammerbuch-Reusten, Germany. Finished as a journeyman.

Schooling

1995 - 1998	Secondary School - Tübingen, Germany
1989 - 1995	Secondary School - Tübingen, Germany
1985 - 1989	Primary School - Tübingen Germany

Conferences & Workshops

November 2009	Cold Spring Harbor Method Course on Computational & Comparative Genomics
September 2009	6th international PhD student symposium Horizons in Molecular Biology, Goettingen (Poster presentation: Evolution of the SNARE protein family in fungi)
January 2009	Biochemical Society / Wellcome Trust Focused Meeting on "Protein Evolution - sequences, structures and systems", Cambridge (Poster presentation and selected oral communication: Evolution of the SNARE protein family in fungi)
September 2007	4th international PhD student symposium Horizons in Molecular Biology, Göttingen (Poster presentation: The SNARE Database)

Teaching Experience

Lecturer (September 2009)	Advanced Method Course “Introduction to bioinformatic tools” within the Göttingen Graduate School for Neurosciences and Molecular Biology (GGNB)

Supervisor (August - September 2009)	6 weeks of practical intern with Mrs. Annette Weizbauer (Classification of the SNAP protein family)
Lecturer (July 2009)	Advanced Method Course "Introduction to bioinformatic tools" within the Göttingen Graduate School for Neurosciences and Molecular Biology (GGNB)

Scholarships

2009	GGNB Travel Grant for the Cold Spring Harbor Method Course on Computational & Comparative Genomics

Publications

Wiederhold K, Kloepper TH, Walter AM, Stein A, **Kienle N**, Sørensen JB, Fasshauer D (2010) A Coiled-Coil Trigger Site is Essential for Rapid Binding of Synaptobrevin to the SNARE Acceptor Complex. J Biol Chem, Epub ahead of print

Kienle N, Kloepper TH, Fasshauer D (2009) Differences in the SNARE evolution of fungi and metazoa. Biochem Soc Trans. 2009 Aug;37(Pt 4):787-91.

Kienle N, Kloepper TH, Fasshauer D (2009) Phylogeny of the SNARE vesicle fusion machinery yields insights into the conservation of the secretory pathway in fungi. BMC Evolutionary Biology, 9:19

Kloepper TH, **Kienle CN**, Fasshauer D (2008) SNAREing the basis of multicellularity: Consequences of protein family expansion during evolution. Molecular Biology and Evolution; doi: 10.1093/molbev/msn151

Kloepper TH, **Kienle CN**, Fasshauer D (2007) An elaborate classification of SNARE proteins sheds light on the conservation of the eukaryotic endomembrane system. Molecular Biology of the Cell, vol. 18, 3463-3471