DISS. ETH NO. 15947

OPTIMIZING THE PROCESS OF NUCLEAR MAGNETIC RESONANCE SPECTRUM ANALYSIS AND COMPUTER AIDED RESONANCE ASSIGNMENT

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of

Doctor of Natural Sciences

presented by

ROCHUS LEONHARD JOSEF KELLER

Diplomierter Elektroingenieur ETH

born 03.28.1966

citizen of Endingen (AG)

supervised by

Prof. Dr. Kurt Wüthrich

accepted on the recommendation of

Prof. Dr. Konstantin Pervushin, examiner
Prof. Dr. Beat H. Meier, co-examiner
Prof. Dr. Rudolf Glockshuber, co-examiner
Prof. Dr. Joachim Buhmann, co-examiner
Prof. Dr. Iain Campbell, co-examiner
Dr. habil. Peter Güntert, co-examiner

2004

Document NMR.017, Version 1.1

Acknowledgements

Many people have generously supported this project by spending their time discussing ideas, reviewing documents and applications, motivating and instructing new users and providing knowledge and infrastructure. I would specially like to thank Dr. Fred Damberger[1] and Pascal Bettendorff[1] for their hard work and dedication as advocates of CARA, convincing and supporting the scientists of the Wüthrich group (and even others). Their high commitment was very important for my motivation to conduct and carry on this project. Besides that I'm very grateful to Fred for reviewing my documents and providing me with valuable feedback. I would also like to thank Dr. Peter Güntert[2] for his great support as my official "Betreuer", and Prof. Dr. Kurt Wüthrich[1], Prof. Dr. Beat Meier[3], Prof. Dr. Konstantin Pervushin[3] and Prof. Dr. Roland Riek[4] for their support and the opportunity to cooperate with their groups.

Finally I would like to thank my examiners Prof. Dr. Konstantin Pervushin, Prof. Dr. Beat Meier, Prof. Dr. Rudolf Glockshuber[1], Prof. Dr. Joachim Buhmann[5], Prof. Dr. Iain Campbell[6] and Dr. Peter Güntert for the assessment of my work, and to Prof. Dr. Nikolaus Amrhein[7] for his great support and his continuous commitment.

[1] Institute of Molecular Biology and Biophysics, ETH Zürich

[2] RIKEN Genomic Science Center, Yokohama

[3] Laboratory of Physical Chemistry, ETH Zürich

[4] Structural Biology Laboratory, The Salk Institute

[5] Institute of Computational Science, ETH Zürich

[6] Department of Biochemistry, University of Oxford

[7] Biology Department, ETH Zürich

Rousseau hat, glaube ich, gesagt: "Ein Kind, das bloß seine Eltern kennt, kennt auch die nicht recht." Dieser Gedanke läßt sich auf viele andere Kenntnisse, ja auf alle anwenden, die nicht ganz r e i n e r Natur sind. Wer nichts als Chemie versteht, versteht auch die nicht recht.

Georg Christoph Lichtenberg

Ich habe übrigens keine Macht. Ich habe Autorität.
Macht heisst, anderen Menschen sagen zu können,
was sie machen sollen. Autorität heisst, dass man
gehört wird. Kardinal Richelieu hatte Macht,
Pater Joseph, sein Berater, hatte Autorität.
Mussolini hatte Macht, der Philosoph
Benedetto Croce hatte Autorität.
Man kann nie beides
gleichzeitig haben.

Umberto Ecco

The only thing necessary for the triumph of evil is for good men to do nothing

Edmund Burke

Table of Contents

Abstract

Nuclear magnetic resonance (NMR) spectroscopy is an important method which allows to determine the three-dimensional structure of proteins and other biological macromolecules. The process is based on the concept of sequence-specific resonance assignment, where cross-peaks between sequentially neighboring amino acids are observed in multidimensional NMR spectra and linked to fragments. The fragments can be uniquely mapped on the sequence, if they become sufficiently long, which allows the atoms of the residue types to be assigned to their corresponding chemical shifts. To assign a protein with 100 or more residues, several thousand signals have to be identified and analyzed. This task is very complex and usually takes several month of manual work. Up to now the available tools left most of the complex information management to the operator, who had to take care of plausibility and consistency by himself.

One of the major achievements of this dissertation is a formal and sufficiently complete information model. This model is able to describe and capture all information coming up during the analysis and resonance assignment of NMR spectra and to ensure consistency. The new software package CARA is a comprehensive implementation of this information model. It follows a semi-automatic approach and causes a significant increase of process efficiency and a decrease of error probability. In contrast to previous solutions like XEASY, whose information management is primarily based on peaklists, CARA makes use of a central repository able to manage abstract and semantically interlinked information objects. The availability of this repository allows CARA to dynamically calculate the needed projections (e.g. the cross-peaks expected in a concrete spectrum) by means of incremental inference algorithms. Further concepts have been developed to simulate the magnetization transfer pathways of NMR experiments, to integrate cross-peaks and to back-calculate and efficiently store NMR spectra.

Zusammenfassung

Die Kernresonanzspektroskopie (NMR) ist eine wichtige Methode zur Bestimmung der dreidimensionalen Struktur von biologischen Makromolekülen im allgemeinen und Proteinen im speziellen. Der Prozess basiert auf dem Konzept der sequenzspezifischen Zuordnung von Kernspinresonanzen. Aminosäuren, die in der Sequenz benachbart sind, manifestieren sich in mehrdimensionalen NMR-Spektren in der Form sog. *Cross-Peaks*, die anhand übereinstimmender chemischer Verschiebungen zu Fragmenten verbunden werden können. Wenn diese eine hinreichende Länge erreichen, ist deren eindeutige Abbildung auf die Sequenz und somit die Zuordnung der beobachteten Signale zu den einzelnen Atomen der Aminosäuren möglich. Die Zuordnung einer Sequenz von hundert oder mehr Aminosäuren erfordert die Analyse und Identifikation von tausenden von Signalen. Diese Aufgabe ist sehr komplex und erfordert oft einen mehrmonatigen Einsatz, verbunden mit viel Handarbeit. Bisherige Werkzeuge überliessen den Grossteil der komplexen Informationsverwaltung der Verantwortung des Operateurs, der sich selber um die Plausibilität und Konsistenz der Daten kümmern musste.

Im Rahmen der vorliegenden Dissertation wurde ein formales, hinreichend vollständiges Informationsmodell entwickelt, das in der Lage ist, sämtliche während der Analyse und Resonanzzuordnung von NMR-Spektren anfallenden Informationen aufzunehmen und deren Konsistenz zu gewährleisten. Das neue Anwendungsprogramm *CARA* ist eine vollständige Implementation dieses Informationsmodells und ermöglicht dank eines semi-automatischen Ansatzes eine erhebliche Steigerung der Prozesseffizienz und Reduktion des Fehlerrisikos. Im Gegensatz zu bisherigen Anwendungen wie XEASY, deren Informationsverwaltung primär auf Peaklisten basierte, verwendet CARA ein zentrales Repository, das alle Informationen in abstrakter und semantisch vernetzter Form verwaltet. Dies erlaubt die dynamische Berechung der benötigten Projektionen (z.B. Cross-Peaks zu einem konkreten Spektrum) mittels inkrementeller Inferenzmechanismen. Weitere Algorithmen ermöglichen die Simulation von Magnetisierungspfaden, sowie die Integration von Cross-Peaks verbunden mit deren Rückrechnung auf NMR-Spektren.

1 Introduction

To understand the function of biological macromolecules, it is indispensable to know their structures. Nuclear magnetic resonance (NMR) spectroscopy today is one of the most important technologies for structure determination. During the last twenty years methods have been developed which allow structure determination of proteins and other biological macromolecules [Wüthrich 1995]. The main advantage of using NMR for structure determination is the possibility to investigate the biological macromolecule in solution, and thus in its natural and physiological environment. This is an important complement to the other established method for structure determination, X-ray crystallography, where the molecules have to be arrayed in the artifical environment of a crystal. Furthermore, NMR can monitor dynamical aspects of proteins in solution, e.g. the interaction between a protein and water molecules in aqueous solutions, which fundamentally revised the picture of proteins as rigid particles [Otting et al. 1995].

The process of protein structure determination by nuclear magnetic resonance spectroscopy is based on the concept of sequence-specific resonance assignment [Wüthrich 1986], where cross-peaks between sequentially neighboring amino acids are observed in multidimensional NMR spectra, and the resulting fragments are mapped on the known amino acid sequence. Several different assignment strategies are available, and the standard approach to obtain complete assignments for proteins up to about 30 kDa in size involves uniform $^{13}C/^{15}N$-labeling and delineation of heteronuclear scalar couplings with triple resonance experiments [Ikura et al. 1990].

To assign a protein with 100 or more residues, several thousand signals have to be identified and analyzed. It is alike to solving a huge, multidimensional jigsaw puzzle with thousands of equal looking parts. Most process steps still have to be executed or - at least - revised manually, because none of the approaches to fully automate the process has been successful up to now. Resonance assignment is thus still the bottleneck of a structure determination project. It usually takes several month (or even years in hard cases) and has to be carried out by highly qualified personnel.

The goal of this dissertation was to improve the process of analysis and resonance assignment of NMR spectra by optimizing its supporting tools, using a semi-automatic approach. It has been seen, that up to now the available tools left most of the complexity of analysis and data management up to the operator. The immense quantity of interlinked information puts a high load upon the cognitive skills. Virtually each step demanded the operator to manually track the consistency and plausibility of the whole database. At the same time the tools themselves are complicated to handle, with the corresponding data models enabling management of only parts of the information. This forced the user to substitute the missing model parameters by provisional means sometimes as primitive as paper and pencil.

The following chapters first provide an overview of the process of NMR structure determination and the problems of current tools. Then a comprehensive model is described, being able to capture and manage all information obtained which is needed during spectrum analysis and resonance assignment. It allows automatic consistency monitoring and inference of new information. This model represents the result of requirements and process analysis performed by the author at the Institute for Molecular Biology and Biophysics during four years. Finally an application of this model to computer aided resonance assignment with the new software package CARA is presented along with the supporting technology of the software (such as new spectrum and file formats).

1.1 Requirements

This chapter documents the official requirements, as they had been formulated by the PhD supervisor at the beginning of the project.

The process of nuclear magnetic resonance assignment shall be analyzed with regard to causes of complexity and inefficiency. The analysis shall incorporate the resonance assignment software tools currently in use at the group of the PhD supervisor. The conclusions of the analysis shall lead to an optimization proposal. The resulting work steps should be conceptually and ergonomically optimized, as such that the interactive analysis of two and three dimensional spectra will be supported in an ergonomically reasonable and efficient way. The operator should be able to access all important operations with less effort. It should be widely possible to do without manual update of peaklists or handwritten logbooks.

The optimization therefore strives for the following goals:

1. increase of process performance and efficiency,
2. reduction of error rate and correction effort,
3. reduction of complexity by more optimal management and presentation of accumulated information,
4. better traceability from structural constraints to spectra,
5. applicability of results also to non-academic use.

The resulting concepts should meet the following requirements:

1. better support of one to three dimensional spectrum analysis,
2. simultaneous presentation of more than one spectrum,
3. adaequate presentation speed for efficient work,
4. better support for 1D slice displays,
5. optimal support for interactive resonance assignment in homonuclear, tripple-resonance and NOESY applications,
6. automated consistency monitoring of user input,
7. automated peak integration,
8. interactive, visual spectrum phasing,
9. extractability of important measurements (chemical shifts, etc.),
10. executable, documented proof of concept.

2 NMR Structure Determination in a Nutshell

This chapter gives a brief overview of the process of protein structure determination using NMR technology. The next chapter then concentrates on the analysis and assignment of NMR spectra, the main focus of this thesis.

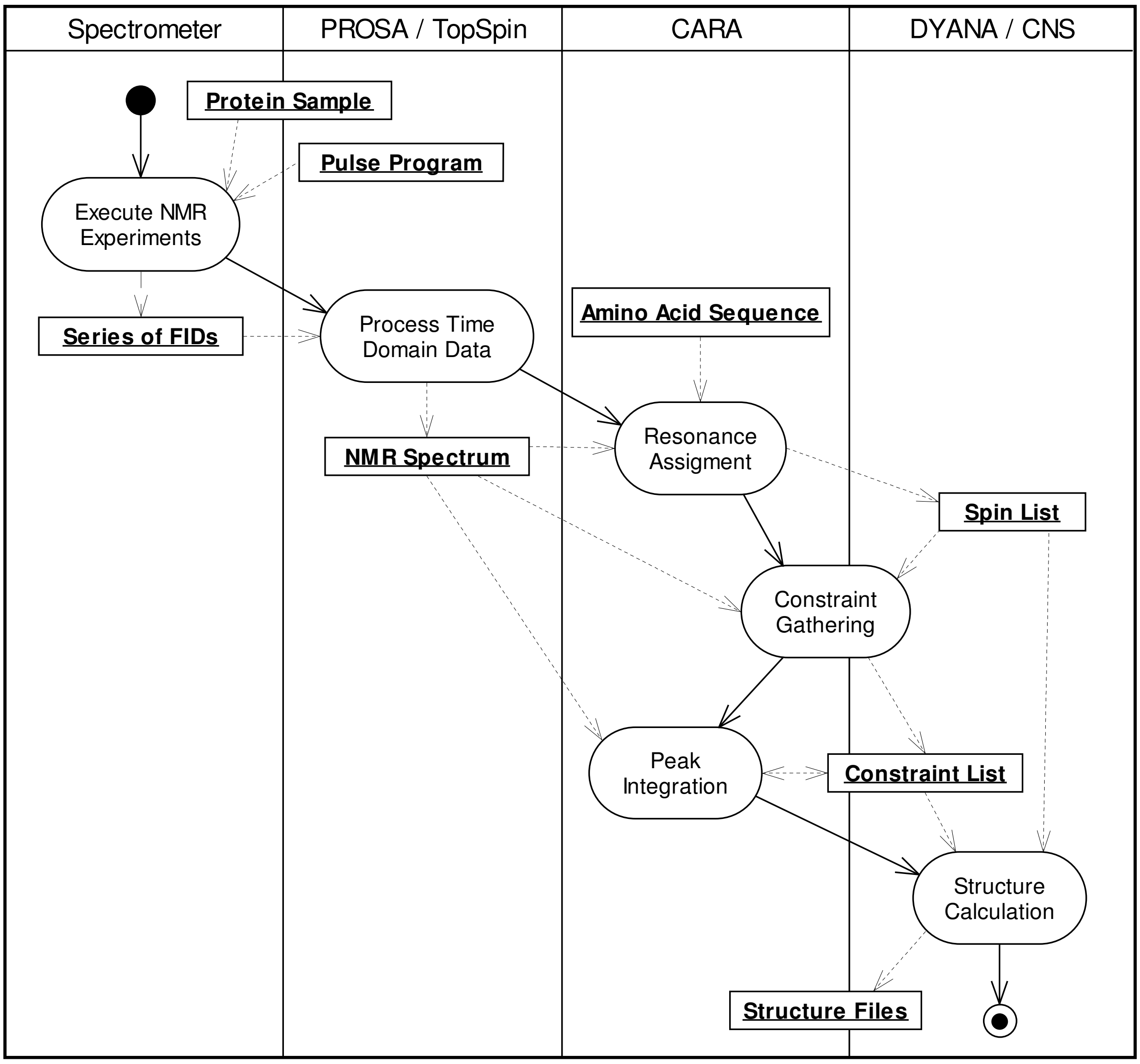


Figure 1: Conceptual Overview of Structure Determination Process

Figure 1 shows the whole process as an UML (*U*nified *M*odeling *L*anguage) Activity Diagram [Booch et al. 1999]. Activities are represented by lozenge shapes (symbols with horizontal top and bottom and convex sides). Like in a flow chart, the flow of control passes from one activity to the next, starting at the black circle. The lanes represent the responsible tools in the context of which the activities happen. The diagram also shows, how information objects (rectangles) flow along the activities.

2.1 Executing NMR Experiments

Before NMR spectroscopy can begin, there is usually a long going phase of sample preparation (not shown in the figure). The interesting protein has to be produced and isolated by complex biochemical processes. Various versions of the protein sample are generated in the course of the process, corresponding to the planned NMR experiments.

An NMR experiment is primarily specified by its characteristic pulse sequence. In practice a pulse sequence (together with other configuration information) is kept in a text file (also called *pulse program*) and used to setup the spectrometer. A pulse program is like a musical score, where each note represents a pulse. Pulses have attributes like carrier frequency, amplitude, phase, shape, starting time and duration, by which the magnetization transfer along the atoms (i.e. spins) of the molecule is controlled. The corresponding physical effects are best described by quantum mechanics [Abragam 1983], which is not the topic of this thesis.

Different types of NMR experiments exist with different characteristics, offering a wide range of strategic means for resonance assignment and constraint gathering. There is a strong interrelation between sample preparation and NMR experiment types, i.e. some experiments require labeling or deuteration of the sample. Labeling means the replacement of the naturally occurring, but not NMR sensitive C and N nuclei by their isotopes ^{15}N and ^{13}C.

One can distinguish between three major groups of NMR experiment types, reflecting their main purpose:

1. Experiments to detect all spins belonging to a single spin system (and therefore the spin systems themselves).
2. Experiments to detect sequential neighborhood of spin systems (i.e. connected by covalent bonds).

3. Experiments to detect close spatial vicinity of arbitrary spins of the molecule (i.e. not only of adjoining spin systems).

Before the advent of double and triple resonance experiments the last two groups were the same.

NMR experiments are usually executed at different times during a structure determination project, depending on the chosen assignment strategy and the available samples.

2.2 Processing Time Domain Data

During a multidimensional NMR experiment a signal (i.e. impulse response) is recorded as a function of several time variables. The signal is first digitized by an analog/digital converter (usually with a 16 bit word length). The subsequent processes need multidimensional spectra as input, so there must be a conversion from the time domain signals to a frequency domain spectrum. The calculation of the spectra is done using a discrete Fourier transformation for robustness and efficiency reasons.

Different optimization procedures are applied before, during or after the transformation to improve the quality of the spectrum, and therefore to ease its subsequent analysis (e.g. to suppress or reduce artifacts inherent to the experiments or the discrete Fourier transformation). The most important procedures thereof are linear prediction, application of suitable window functions, phase optimization, suppression of signals caused by the solvent, and base line correction.

To save sampling time, the slope of the impulse response is usually not recorded in full length. Instead a part of the signal is substituted by predicted points using linear prediction algorithms, which typically extends the digitized time domain data with additional points. The additional calculated points help to decrease line width and reduce oscillation effects on the edge of the frequency lines (which are effects of an abrupt cut off of the decay of the time domain signal). A similar improvement can be achieved by application of suitable window functions, which compensate for the finiteness of the recorded time domain signals. If the sampled FID is simply cut off at time T, this corresponds to a multiplication of the time domain signal by a rectangle function of width *T* and thus a convolution in the frequency domain by *sinc(f T)*. This effect can be diminished by selecting a more appropriate function than a rectangle.

The phase optimization is necessary to get spectra with pure absorptive signals. This is done by application of a constant and linear phase angle to the real and imaginary part of the original spectrum (either manually with visual feedback or automatically using programs like PROSA [Güntert et al. 1992]).

The baseline of an NMR spectrum can be distorted for several reasons. There are methods to reduce this distortions already during the experiment (e.g. by optimizing phase cycles or use of oversampling). But also linear prediction of time domain signals or application of corrective functions in the frequency domain are necessary in practice (as done in PROSA).

A comprehensive description of processing procedures and algorithms can be found in [Cavanagh et al. 1996].

2.3 Resonance Assignment

The goal of resonance assignment is to uniquely associate each nuclear spin of the sample molecule with a chemical shift. In a 1D spectrum, each frequency line (i.e. the position of the maximum intensity within the lorentzian line shape, called *peak*) represents one or more spins (i.e. atoms). Given a small molecule and a spectrum with a sufficiently good resolution (i.e. high number of sample points and long maximum observation time), there is a good chance that individual frequency lines can be recognized and associated with exactly one spin (or with a group spin such as rapidly rotating CH_3). With increasing molecule size, more and more frequency lines will fall on the top of each other (or overlap at least). It is also possible for experimental reasons that expected frequency lines are not visible, or spurious signals appear. All these effects thus make it nearly impossible in practice to uniquely associate each frequency line with an individual spin.

Over the years new experiment types have been developed to overcome some of these problems [Ikura et al. 1990]. An essential progress was the introduction of *correlated spectroscopy* (COSY) experiments, in which each signal not only represents a single atom or atom group (as in 1D spectra), but a specific correlation between two or more atoms (i.e. spins). For example in a 2D [^1H,^1H]-COSY (Aue et al. 1976) spectrum each peak represents a scalar spin-spin coupling between two hydrogen atoms being apart up to three covalent bonds. Because of this restriction the spectrum only shows correlations between spins of the same residue. Nearly all intraresidual hydrogen spins of an amino acid can be correlated, so that they form one connected graph (or up to three disconnected graphs in Histidine, Phenylalanine, Tryptophan and Tyrosine), where each vertex represents a spin (identified by its chemical shift) and each edge corresponds to a correlation (i.e. cross-peak). Such a graph is called a *spin system*. We still expect each spin or chemical group to have a unique chemical shift position in ideal case.

A spin system has a characteristic topology in a 2D [^1H,^1H]-COSY spectrum, by which it can be recognized. Seven different topologies can be uniquely associated with a residue type (i.e. amino acid) each. It is thus possible to directly identify the corresponding residue type (and so the spins it contains) by matching the observed peak pattern with the catalogue of topologies [Wüthrich 1986]. The remaining thirteen amino acids share three different topologies (thus only an ambiguous identification is possible by means of pattern matching). Their unambiguous assignment is only possible if additional information becomes available (i.e. after sequence-specific assignment).

To vary the number of visible signals in an NMR spectrum, the sample protein is usually measured both in H_2O and D_2O solution. In the latter the labile hydrogen atoms are replaced by deuterium, so that the ^1H spectrum only contains signals of the carbon-bound hydrogen atoms. By comparing the two spectra the nitrogen-bound hydrogen atoms can be identified.

Another very important type of experiment is the 2D [^1H,^1H]-NOESY (acronym for *nuclear Overhauser spectroscopy*, [Anil-Kumar et al 1980]), which shows a signal for each pair of hydrogen atoms (i.e. of their spins) with a internuclear distance of less than 5 Angstrom. The related atoms can belong to arbitrary residues, according to the 3D structure of protein, usually including sequential neighbors. A NOESY experiment can thus be used to sequentially connect the spin systems found in COSY spectra (using the fact that in ideal case a spin is represented by the same chemical shift in both spectra). The same experiment will also be used for constraint gathering described in the next chapter.

The process of homonuclear sequence-specific resonance assignment [Wüthrich 1986] consists of the following steps, orchestrating the experiment types previously introduced (usually applied to proteins of a molecular weight up to 10 kDa):

1. The spin systems of the residues are identified in D_2O solution using (indirect) through-bond ^1H-^1H couplings (i.e. COSY type spectra) as far as possible.
2. The spin systems are then completed by further studies in H_2O solution using J couplings with the labile hydrogen spins.
3. Sequentially neighboring spin systems are then identified using sequential NOESY cross-peaks, and as many spin systems as possible are thus combined to fragments (i.e. chains corresponding to peptide segments).
4. Sequence-specific assignments are then obtained by matching these fragments to the (previously known) amino acid sequence. A unique mapping of a fragment is possible, if it is sufficiently long to uniquely match a peptide segment of the primary structure of the protein.

If ^{15}N and ^{13}C labeled protein samples are available, other NMR experiments and assignment strategies can be applied. For sequential assignment of the protein backbone a combination of 2D [^1H,^{15}N]-HSQC (Mori et al. 1995) and 3D HNCA (Grzesiek et al. 1992) is widely used. These experiments extend the concept of COSY and additionally show correlations between spins of different types (i.e. by scalar through-bond spin-spin couplings). The 2D [^1H,^{15}N]-HSQC shows one peak per individual ^{15}N-^1H pair connected by a single covalent bond. Since the backbone of a protein only contains one ^{15}N spin, this experiment can be used to directly identify the H^N and N chemical shifts of each residue (except for Proline). An HNCA spectrum shows two peaks for each H^N-N pair (i.e. for each residue except the Prolines), the stronger one usually corresponding to the intraresidual $^{13}C^\alpha$ and the weaker one to the sequentially neighboring $^{13}C^\alpha$ (i.e. $^{13}C^\alpha{}_{i-1}$ in N-terminus direction).

Where as in homonuclear assignment the characteristic topology of the spin systems is mainly used to match the fragments with the sequence, the heteronuclear assignment can make use of the fact, that the $^{13}C^\alpha$ and $^{13}C^\beta$ spins of the amino acids have a quite characteristic chemical shift distribution [Grzesiek et al. 1991]. It has been shown, that there is a high probability to find a unique match for fragments of length three (or larger) [Grzesiek et al. 1993]. The process of heteronuclear backbone assignment therefore consists of the following steps (simplified):

1. Identify the H^N-N pair of all possible residues using a 2D [^1H,^{15}N]-HSQC spectrum of the protein.
2. For each H^N-N pair identify the $^{13}C^\alpha$ and $^{13}C^\alpha{}_{i-1}$ using a 3D HNCA spectrum. The H^N-N pairs are used generate out two-dimensional strips along the ^{13}C dimension of the HNCA, each showing the $^{13}C^\alpha$ and $^{13}C^\alpha{}_{i-1}$ peak.
3. Find pairs of strips with corresponding $^{13}C^\alpha{}_{i-1}$ and $^{13}C^\alpha$ and combine as many strips as possible to fragments (i.e. chains of strips).
4. Sequence-specific assignments are then obtained by matching these fragments to the (previously known) sequence. A unique mapping of a fragment is possible, if it is sufficiently long to uniquely match a peptide segment of the primary structure of the protein (according to the ^{13}C random coil shift distribution).

Several variations and extensions of this procedure exist (e.g. use of additional spectra like 3D ^{15}N-ed. [^1H,^1H]-NOESY (Talluri et al. 1996), HNCACB, CBCA(CO)NH, etc.), but the main concept remains the same and is applicable to proteins with molecular weights around 20 kDa. The ^{15}N-ed. [^1H,^1H]-NOESY can be used to confirm the correctness of the fragments built by CA/CA-1 and CB/CB-1 agreement (or to link fragments when there is no other way, as is the case with Prolines).

The atoms of the sidechains are usually assigned by simultaneously analyzing COSY and TOCSY spectra (and also NOESY if necessary). A 2D [^1H,^1H]-TOCSY (short form for *total correlation spectroscopy*, [Braunschweiler et al. 1983]) spectrum shows a superset of the signals of a COSY spectrum, i.e. several successive ^1H-^1H relations over up to three bonds each. The peaks visible in a COSY spectrum are therefore a subset of the peaks seen in TOCSY. The additional cross-peaks fill up the COSY topology of the spin system and constitute a characteristic pattern (the so called *TOCSY tower*), which repeats at the shift position of each ^1H spin of the system, and is symmetric with respect to the diagonal axis. This pattern can be used to find all fragments of strips belonging to a single spin system. By comparing the COSY and TOCSY spectra at the position of a pattern, the peaks remaining in the COSY spectrum can be directly assigned. This concept is applicable to both homonuclear and heteronuclear assignment strategies (the latter using 3D HC(C)H-TOCSY spectra (Bax et al. 1990), where the towers appear at all strips given by ^{1}H-^{13}C the pairs of a spin system).

The result of the resonance assignment is a so called *spinlist* (or *atomlist* - as it was called in XEASY - containing an atom with its associated chemical shift in each row). In chapter 4 a formal model of resonance assignment will be introduced.

2.4 Constraint Gathering and Peak Volume Determination

As described before a 2D [^{1}H,^{1}H]-NOESY shows a cross-peak for each pair of hydrogen atoms (i.e. spins) with a internuclear distance of less than 5 Angstrom. The intensity of the dipolar coupling between two atoms (and thus of the visible peak) depends on their distance, according to the formula $V = c\frac{1}{d^6}$, where *V* is the volume of the NOESY peak, *c* is a calibration constant and *d* is the spatial distance between the atoms involved (usually used as an upper distance limit during structure calculation). The spatial structure of a protein can be calculated by a sufficiently complete set of such distance constraints.

The resonance assignment process aimes to associate each NMR sensitive spin of the molecule with its chemical shift. With this knowledge the NOESY spectra can now be interpreted. Each peak corresponds to a distance constraint between the spins represented by their chemical shift position. The volume of the peak is used to calculate the upper distance limit of the constraint (according to the given formula above). The precision of the structure grows with the number of discovered distance constraints and the accuracy of the peak volume determination.

As seen before the chemical shifts of more than one atoms can fall together (i.e. be degenerate). A distance constraint can thus become ambiguous. This can partly be circumvented by use of 3D ^{13}C-ed. [^{1}H,^{1}H]-NOESY spectra (Ikura et al. 1990), in which the ^{1}H-^{1}H-cross-peaks are dispersed along the ^{13}C dimension (thus reducing the probability that they overlap). Another way is to directly use the ambiguous assignments [Nilges et al. 1997] for structure calculation (as it is done in CANDID

[Herrmann et al. 2002b]), using condition $\bar{d} \geq \left(\sum_i d_i^{-6} \right)^{-\frac{1}{6}}$, where d_i are the distances between all atoms involved in the assignment and $\bar{d}$ is the resulting ambiguous distance constraint (which is fulfilled if just one of the d_i fulfills it). Given a reasonably complete assignment, the corresponding NOESY peaks can even be automatically found in most cases (as is done in the program ATNOS [Herrmann et al. 2002a]). In chapter 4.5.1 an algorithm for the automation of peak volume determination is described.

The result of the process of constraint gathering and peak volume determination is a *constraint list*, which contains the upper distance limits of all spatially neighboring atom pairs.

2.5 Structure Calculation

In the last two sections we saw how distance constraints of a protein structure can be found by analyzing NMR spectra. Each constraint represents a pair of spins with an internuclear distance of less than 5 Angstrom. The volumes of the corresponding NOESY peaks were used to calculate the upper distance limits (which are between 2 and 5 Angstrom). With this information a structure can be calculated by folding the primary structure of the protein in such a way, that its atoms optimally fulfill the distance constraints. It is not possible to directly formulate and solve an equation system due to the complexity of the problem. A solution has to be found by an optimization procedure instead. Programs like DYANA [Güntert et al. 1997] or CNS [Brüngera et al. 1998] use the same constraint set to simultaneously calculate a set of structures (called *conformers*). Each calculation starts from an initial random conformation, which is then optimized by simulated annealing, i.e. by simulating the movement of the atoms under heating condition and their return to an energy minimum when cooling down while narrowing the upper limits of the given distance constraints. The quality of the conformers can be assessed by monitoring the convergence rate and comparing their variance (i.e. the degree of their topological agreement). The resulting structure is considered to be "optimal", if the deviation of the conformers is minimal, while only a minimal number of constraints are violated.

Even if it is not possible to prove that the procedure has found a global minimum and there was no alternative solution of equal significance, it could be shown in practice (i.e. by comparing the results with structures already known from crystallography) that it is possible to determine the correct structure, if enough distance constraints are available. The quality of the structure is thus directly dependent on the completeness and correctness of the resonance assignments.

3 Discussion of Previous Approaches

This chapter gives a brief overview of the currently used procedures of resonance assignment, constraint gathering and peak volume determination. The discussion focuses on the program XEASY [Bartels et al. 1995], since it was developed at the Institute for Molecular Biology and Biophysics, where this research project was carried out. Most facts also apply to other tools like Sparky [Goddard et al. 2000], NMRView [Johnson et al. 1994] or Felix [Krezel et al. 2000].

3.1 Peaklist Based Process

All NMR software packages known to the author base their information management on peak- and spinlists. A peak is a position vector pointing to an intensity maximum in a given NMR spectrum. The peaklist is the set of all peaks belonging to this spectrum. Each peak is handled as an individual entity, which is uniquely identified by a peak number within its peaklist. A peaklist itself is identified by a name which is usually unique within a project. Each peak can optionally be associated with an arbitrary label, which is only used for descriptive purposes (i.e. without any semantic significance).

For each project an spinlist is first generated from a domain specific molecule library (i.e. a file listing all spins of all amino acids). Each atom of the molecule library is uniquely identified within its molecule by an ordinal number and a mnemonic (i.e. a descriptive label). According to the given amino acid sequence, the atoms of each referenced amino acid are then copied from the library and renumbered to be uniquely addressable within the spinlist (by atom number). The spinlist has an independent life cycle from that point (i.e. changes within the library are not automatically reflected in the spinlist).

Each element of the peak position vector from a peaklist can be associated with an atom number pointing into the spinlist. This reference is the result of the resonance assignment process and associates each spin with the resonances observed in the spectra. A peak is first picked in a spectrum and later - as soon as this information is known - assigned to the atoms it represents (one per dimension).

This peaklist based approach has many serious disadvantages leading to redundancy, ambiguity and disturbing cross-dependencies. It is thus directly responsible for the additional complexity and the corresponding loss of efficiency.

A peaklist is an entity with an independent life cycle. Since it is associated with a spectrum, it directly inherits all features of the corresponding spectrum type (which in turn inherits its features from the originating NMR experiment): the number and spin type of its dimensions and implicitly the magnetization transfer pathway represented by its peaks. It is thus not trivial to compare or combine peaklists. Even if two spectra have the same or a similar spectrum type, usually separate peaklists have to be managed, because peak positions can slightly differ between spectra. Some programs at least allow the user to decide on the peak number, so equal peaks can be recognized by equal numbers (if not, the user has to manage his or her own lists relating peaks of different peaklists). It becomes even more complex, if a peaklist corresponds to a subset of another peaklist (as is the case with 3D HNCA/HNCACB spectra), or only vector components of a peak correspond to vector components of another peak from another spectrum type (as is usually the case with HSQC/HNCA spectra) or from the same spectrum (as is the case with all 3D COSY, TOCSY and NOESY spectra). In these cases the synchronization of peak numbers is no longer feasible or possible (because one peak can relate to many other peaks) and the user definitely becomes responsible for the management of the interdependencies. To reduce the need for synchronization, the user is forced to follow a rigid order of process steps, where each step is based on a independent copy of the peaklist established in the previous step (e.g. first pick HSQC, then HNCA, then HNCACB, and so on).

Another serious problem is the fact, that the assignments of the peak position vector elements to a spin can become ambiguous in many aspects. On one hand a spin can be referenced by an arbitrary number of peaks, each at potentially another chemical shift position. It is often difficult to decide whether the peak is only moved or the assignment is wrong. In practice a chemical shift tolerance range is used to make this decision. Usually the mean chemical shift position of all assigned peaks is used as the reference chemical shift of the atom. This works reasonably well as long as the deviation is not to high and there is no mix-up of folded and non-folded peak positions. This happens quite often, can go unnoticed and may only be discovered by carefully analyzing the output from structure calculation programs (e.g. DYANA).

On the other hand also the assignments of the dimensions of a single peak can contradict to other peaks. This happens if an assignment is not properly updated in all peaks where it occurs (e.g. after a change of a fragment during sequential assignment).

3.2 Bucket Chain Process Model

As described in the previous chapter the user is forced to follow a rigid order of process steps because the synchronization of peaklists is otherwise unfeasible. For example the backbone assignment usually starts with the analysis of an HSQC spectrum, where the HN and N chemical shifts of all spin systems (besides Prolines) can be identified. The resulting peaklist is then converted to a striplist and 3D peaklist to be used in the HNCA spectrum, where all C chemical shifts initially are at a constant value. Each strip is then analyzed, moving one peak and picking a second peak per strip (corresponding to CA and CA-1). If by chance a strip is found which was not visible in the 2D [^{1}H,^{15}N]-HSQC, two peaklists and one striplist have to be updated. If the user then wants to simultaneously use a HNCACB spectrum, the existing HNCA peaklist is copied and extended by the CB and CB-1 peaks. Usually also a ^{15}N-ed. [^{1}H,^{1}H]-NOESY spectrum is used for confirmation of the fragment building process, demanding again another peaklist. If there is now a change to one peak of the HSQC peaklist, this introduces an inconsistency to many peaks in three other peaklists and to the striplist. The same problems of synchronizing information occurs during sidechain assignment and structure calculation. It is also important to note, that the tracing of a change along all peaklists is all but trivial in programs like XEASY, because peak numbers are the major navigation unit, although they have no semantic meaning (i.e. the user has to first seek for all peak numbers where a certain assignment occurs). That's why users typically note down the interesting peak numbers, to which they came across recently, on a sheet of paper.

To conclude, the later a change occurs the more time expensive it is. A fairly efficient use of the available tools is only possible, if work can be divided up into sequential, autonomous phases, where each phase can be finalized before the next one begins, i.e. information is handed over like water in a bucket chain. But this completely contradicts the way spectroscopists usually work. It regularly happens that a decision can only be taken or has to be revised, if more information is available (e.g. a peak in a spectrum is often not visible or covered by a cluster and can therefore only be indirectly deduced by taking into account other assignments).

Instead of the bucket chain approach, an iterative, incremental process model would be more natural to the user. This would require a central repository which allows an incremental, redundancy-free management of information (see chapter 4 and 6). Because of the highly interlinked nature of this information, the current, table-based approach, where the user has to manually enter reference numbers, is very inefficient. The repository should incorporate all information needed to carry out the process, i.e. there shouldn't be a need for the user to substitute the missing model dimensions by paper and pencil (or other non-integrated means).

3.3 Dispersed Information Management

Many of the available tools challenge the user by numerous input and output files, sometimes obscuring where the precious results of the work are kept. It is often difficult to determine, which files belong to a project. XEASY has to handle many file formats, e.g. spectra, peaklists, sequence, spinlists and the like. Nearly all of these formats are proprietary and do not adhere to any standards. Users often have to program their own converters, if they want to reuse the information or cooperate with other software packages (which can be quite complex if the format cannot be described by regular expressions). Experience showed that file formats are only documented in rare cases. Text-based formats can usually be reverse-engineered (which seems to be the common approach within the NMR community). This is not always possible with binary formats, e.g. like the XEASY default file format, which is even hard to understand if the source code is available. The XEASY default file in principle contains all project information and thus can be used as a "project file" (although its original purpose is to store the current program state), but the program is quite stingy with the user when it comes to presentation and management of its information content (e.g. it is not possible indeed to list or unload spectra, and

XEASY crashes if more than 14 spectra were loaded). For this reason users usually work with different default files in parallel (e.g. one for HSQC/HNCA/HNCACB, one for HC(C)H-TOCSY/^{13}C-ed. [^{1}H,^{1}H]-NOESY, one for NOESY so as not to mix up the spinlist with the triple resonance peaks when executing the AC command, etc.), which additionally complicates the synchronization of peaklists and assignments. Much of the available information remains unused though, because it is not accessible by the user.

Because the default file is non-transparent, users usually keep information in redundant peaklists and spinlists. They are then responsible by themselves to establish a useful naming scheme (to describe the contents of the file, i.e. to avoid mix-up of peaklists for a spectrum) and to track the file versions. There is always a danger to inadvertently open or edit the wrong default file, because XEASY opens and writes to the one in the current directory or in an arbitrary path preset by an environment variable. The same problems apply to library files, where the user has to always take special care that the right version is used.

Another common problem is platform independence. Many scientific groups work on different processor and operating system architectures in parallel. If a binary file format is dedicated to a certain platform type, its use in such an environment is quite restricted (e.g. an XEASY default file created on an Irix machine cannot be used on Linux. The same problem exists for NMR Pipe spectra [Delaglio et al. 1995]). If the file gets corrupted (e.g. because of program crash), there is usually no way to restore the information.

A new software package should therefore avoid the use of undocumented, proprietary, binary file formats. Information should not be dispersed in many files, but be kept in one single place which is unambiguously identifiably by the user. The integration of supporting algorithms should be possible on application level, i.e. by direct memory access or inter-process communication. Data exchange on file level should be avoided, or at least be done using standard file formats (e.g. on XML base). It shouldn't be necessary anymore to manually write a configuration file for an external algorithm, and then again manually enter the results of this algorithm into a dialog of the original program (as is the case with XEASY when using MAPPER).

3.4 Automatic vs. Manual Approach

Even if spectroscopists obviously don't appreciate fully automated programs, but prefer to keep every process aspect under their control, today's programs are expected to intelligently follow the actions of the user and to make proposals or complaints where necessary. This is only possible if the program has enough information about the application domain. It is astonishing to see, how many information sources are regularly used by spectroscopists, but how few of them are yet integrated into the available software packages (e.g. the printed figures of all amino acids, the COSY/TOCSY spin system patterns and the lists with ^{13}C chemical shift statistics seem still to be a faithful companion of each spectroscopist working with these programs).

The conceptual model of a program should be comprehensive enough to be able to absorb all needed information of the problem domain, and to infer new facts and ensure consistency as the user is acting on it. The user should not be forced to enter information which could be inferred by the software.

3.5 Usability Challenge

It is already known from other application domains that the usability of a system can have a tremendous impact on the efficiency of a process. The author was himself responsible for projects where the efficiency could be raised by at least a factor of two by only improving the usability (i.e. without any process changes).

Programs like XEASY are an outstanding piece of work. Of course the developers concentrated on the essential functionality and tried to incorporate every feature into a single window as far as possible. The advantage of this approach - next to the reduction of development effort - is the fact, that every function is accessible from one place. Commands can either be executed by entering a mnemonic or activating a popup menu. The disadvantage is the equal distribution of interaction cost over all functions, not considering the frequency of their use (e.g. it requires less steps to average all chemical shifts than to pick a single peak), and the relatively complex feature accessibility and interaction gestures (e.g. it takes one command and up to twenty mouse clicks to display and scale a slice window). Because most things in

XEASY happen in only one window, the user must always undertake many configuration steps to use the window in another context (or simultaneously work with different instances of XEASY, raising the cost of information synchronization).

If the effort to access a feature is too high, users tend to not use it. In XEASY users couldn't afford to verify each peak position in the slice window, which led to less precision in peak picking and thus a larger chemical shift deviation (one couldn't even pick peaks if the slice window was open). Another tedious problem is the placement of a peak along depth direction (i.e. the selection of the optimal plane). For each peak the users usually enter a sequence of FP/BP commands to switch between adjacent planes before they can enter PP and click the mouse at the peak position.

There are many other things which proved to reduce the efficiency and to increase the error rate of the user. The navigation is costly (only by mnemonic, not by mouse or cursor keys), many things can only be done by mouse (e.g. a subset of peaks has to be selected by mouse to create a striplist), there is no undo function and some functions cannot be reversed (e.g. loading a spectrum), the user has to take care of implementation details (e.g. the folding concept is based on a spectrum-dependent peak status and thus the source of many errors), many functions require the user to memorize parameter values (e.g. the calibration) and finally there is no preview possibility for the printing function (e.g. the user has to generate and open a postscript file if she wants to see the effect of a parameter change).

A user interface should be optimized for efficient access to the most likely use cases. Related use cases should be grouped into dedicated environments, to not overstress the interface or confuse the users with many unneeded options. Advanced users should be enabled to extend or customize the environment to their specific needs, without danger of corrupting the fundamental functions and concepts (as can be the case in programs written using a scripting language, e.g. such as Sparky [Goddard et al. 2000]). Also the installation of the tools should be easy (i.e. possible without assistance of an administrator and without first compiling a lot of source files).

3.6 Peak Volume Determination Concepts

The volume determination of NOESY peaks is an important step, directly influencing the quality of the structure. Up to now the volume determination was mostly done using a manual approach, because automatic volume determination algorithms couldn't gain wide acceptance. The manual volume determination using XEASY can be done e.g. by selecting a rectangle or elipse around a selected peak and then an algorithm sums up all intensity values within the area. This works quite well with 2D spectra and non-overlapping peaks (although it takes a while). Even automatic algorithms could do quite a good job under these conditions. The complexity arises when peaks are overlapped. The manual decomposition of overlapping peaks is very difficult and time consuming. Up to now there were no robust algorithms available which allow a reliable automation of this process. Programs like SPSCAN [Glaser, R., www.molebio.uni-jena.de/~rwg/ spscan/] or AUTOPSY [Koradi et al. 1998] make use of sophisticated decomposition algorithms, which try to assign the sample points of the spectrum file to the picked peaks. This approach has several limitations. Low resolution of the spectra can lead to ambiguity and round off problems and therefore large errors for overlapping peaks. Some algorithms require users to adjust many parameters with often rather unintuitive semantics. Adjustment of parameters and assessment of output quality is even more difficult if an algorithm makes use of random functions. Online optimization of parameters is only possible if calculations can be done in a reasonably short time (e.g. a few minutes on an average workstation). Therefore, in practice interactive volume determination methods are still preferred over automatic algorithms. Apparently there is a fundamental need for a robust and understandable algorithm which is able to automatically determine the original i.e. decomposed amplitudes of overlapping peaks in short time with a small number of intuitive parameters to be adjusted, enabling the spectroscopist to optimize and verify the outcome and assess its quality. Chapter 4.5.1 describes an algorithm offering these features.

3.7 Discussion

The analysis of the current resonance assignment process and its supporting tools leads to the following overall requirements and recommendations:

1. An appropriate conceptual model is needed. The resonance assignment process shall no longer be *peak centric*, but rather *spectrum transcendent* (i.e. the resulting information should be abstracted from concrete spectra). The conceptual

model should provide appropriate degrees of freedom, such that an efficient, redundancy-free representation and management of the information resulting from the resonance assignment process is possible.

2. A repository based information management approach is needed. The conceptual model should be translateable to a domain specific repository, which allows all information to be managed on one place and in a canonical way. It should be able to define formal integrity conditions which support automatic referential consistency checking, allowing for a more optimal incremental and iterative resonance assignment process. The repository should provide for dynamic extension possibilities of the model.
3. A semi-automatic approach is needed. All decisions shall be under control by the operator, but the system shall be able to make proposals and to maintain consistency. It shouldn't be necessary to enter information, which can be inferred by the system.
4. An extensible workbench approach is needed. Tools should offer a graphical user interface adhering to accepted standards and providing state-of-the-art usability features (e.g. undo). The user interface should be task centric and job specific, i.e. the operator on one hand only has to care about the features needed for the current task, and on the other hand has immediate access to the most needed operations with minimal interaction cost. The workbench shall be extensible, so the operator can prototype and implement custom algorithms and plugins for specific needs. The features of the conceptual model, the repository and the workbench shall be reusable, thus leveraging adaption to new application domains.

The following chapters describe a conceptual model and a proof of concept which satisfy the given requirements and recommendations. The scientific contribution of this thesis consists of the proven conceptual model of the resonance assignment process and the resulting increase of the process performance and efficiency (as described in chapter 1.1).

4 Conceptual Model of Spectrum Analysis and Assignment

In this chapter a conceptual model of the problem domain of this thesis is presented. As we will see it is sufficiently comprehensive to formally describe the process of analysis and assignment of NMR spectra and its associated information. It is complete enough to capture the complexity a spectroscopist is confronted with in the course of assignment, but still lean enough to be comprehensible. This model represents the result of requirements and process analysis performed by the author at the Institute for Molecular Biology and Biophysics. Several prototype implementations and evaluations led to its refinement and optimization. As such it forms the foundation of the optimized process introduced in chapter 6.

The model differentiates itself from the processing (see chapter 2.2) and structure calculation (see chapter 2.5) stage in that it doesn't incorporate time domain or structure information.

This thesis makes use of the Unified Modeling Language (UML), an international standard maintained by the Object Management Group (OMG), to specify its models. UML is ubiquitous and well documented [Booch et al. 1999]. It is the result of a long going community process and incorporates a wealth of experiences made with earlier specification languages. One of the great achievements of this kind of specification methodology is the separation of conceptual and technical aspects of a system, i.e. it can be seen as a formal communication medium between software engineers and their customers, without the assumption of the latter to be computer scientists. As such UML has for a long time entered non-software-engineering domains like business process modeling and reengineering. For a brief overview see page 131.

4.1 Modeling Molecules

Nuclear magnetic resonance spectroscopy is about detecting spins and spin relations of a molecule and associating their chemical shifts with the atoms (or group of atoms) of the molecule. This thesis puts the focus on proteins, which are macromolecules consisting of linear chains of amino acids as uniform building blocks. A similar construction principle adheres to DNA and nucleic acids which are also commonly analyzed using NMR technology.

We generalize the concept of the uniform building blocks by introducing the class *ResidueType* (see Figure 2). The name is derived from the primary structure of a protein which is a chain or sequence of amino acid residues, connected over a common backbone. A residue can be seen as an "instance" of an amino acid, which therefore becomes its type.

A *ResidueType* is identified by a name (e.g. "Alanine"), a shorthand (e.g. "ALA") and a letter code (e.g. "A"). The shorthand is used to uniquely identify an instance of a ResidueType within the model.

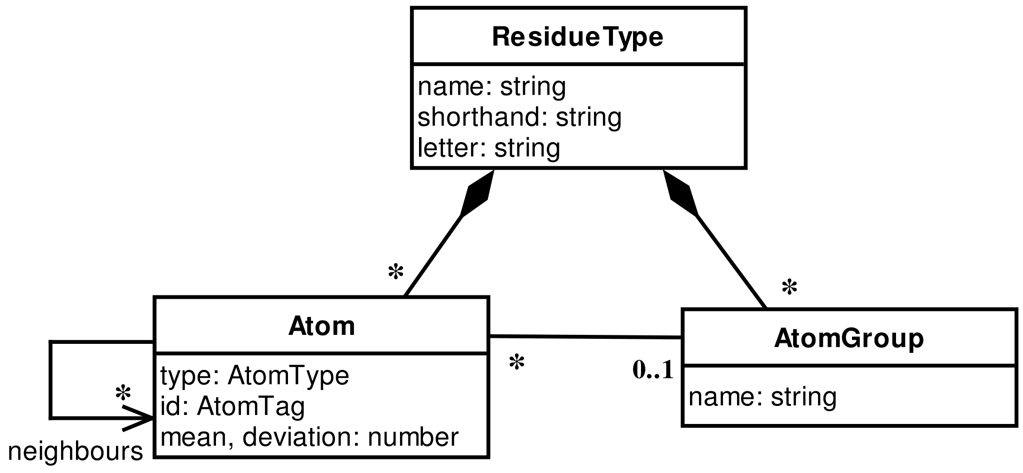


Figure 2: The building blocks of a macromolecule

A *ResidueType* is a molecule by itself, which can be seen as a network of atoms connected by covalent bonds. We therefore introduce the class *Atom*, representing an instance of an atom type. The type is specified by the attribute *type*, whose values are the symbols of the periodic system of elements (value domain *AtomType*). In case of amino acids only the symbols *H*, *N*, *C*, *S* and *O* are needed. We don't differentiate between isotopes and their normal forms, because the use of isotopes for protein labeling is an implementation detail of NMR and only relevant in the context of sample preparation and measurement (see chapter 2.1).

An *Atom* within a *ResidueType* is uniquely identified by its attribute *id*, which in practice is the atom type symbol combined with a Greek letter and an optional number (e.g. H^{α}, C^{α}, $C^{\beta 1}$, $C^{\beta 2}$, etc.). Because of compatibility with other software packages (e.g. DYANA [Güntert et al. 1997]) we propose to use ASCII [Sebesta 1999] letters and digits exclusively, but to otherwise obey the common nomenclature for amino acids. The Extended Backus-Naur Form (EBNF [Sebesta 1999]) rule for the domain *AtomTag* can therefore be written as follows:

Eq. 1

$$AtomTag ::= \{letter \mid digit\}$$

The atom network is represented in the model by the unidirectional *neighbors* association, which is a set of references from one *Atom* to many other *Atoms* of the same *ResidueType* (not including itself). The model is consistent if the relation in Eq. 2 always holds (with *A* and *B* being *Atoms*).

Eq. 2

$$A \in neighbours(B) \Leftrightarrow B \in neighbours(A)$$

The optional attributes *mean* and *deviation* allow an *Atom* to be associated with a random coil chemical shift distributions. They are explained in chapter 4.2.2.

The class *AtomGroup* is used to put together *Atoms* with similar properties. We use *AtomGroups* to mark *Atoms*, which usually cannot be uniquely assigned before structure calculation (i.e. the H^{δ} and H^{ε} pairs of the aromatic ring of Phenylalanaine). We don't use *AtomGroups* to handle chemical groups like CH_3 or NH_3, where the *H* always have the same (i.e. degenerate) chemical shift. Instead the hydrogen atoms of these chemical groups are directly modeled by a single *Atom* (e.g. Q^{β} of Alanine or Q^{δ} of Isoleucine), corresponding to the structure effectively seen by NMR. XEASY [Bartels et al. 1995] handles these two cases collectively by *pseudo atoms*, which eventually requires a reassignment when the stereo specific assignment becomes known during structure calculation.

The following diagram shows the application of the classes of Figure 2 for Alanine.

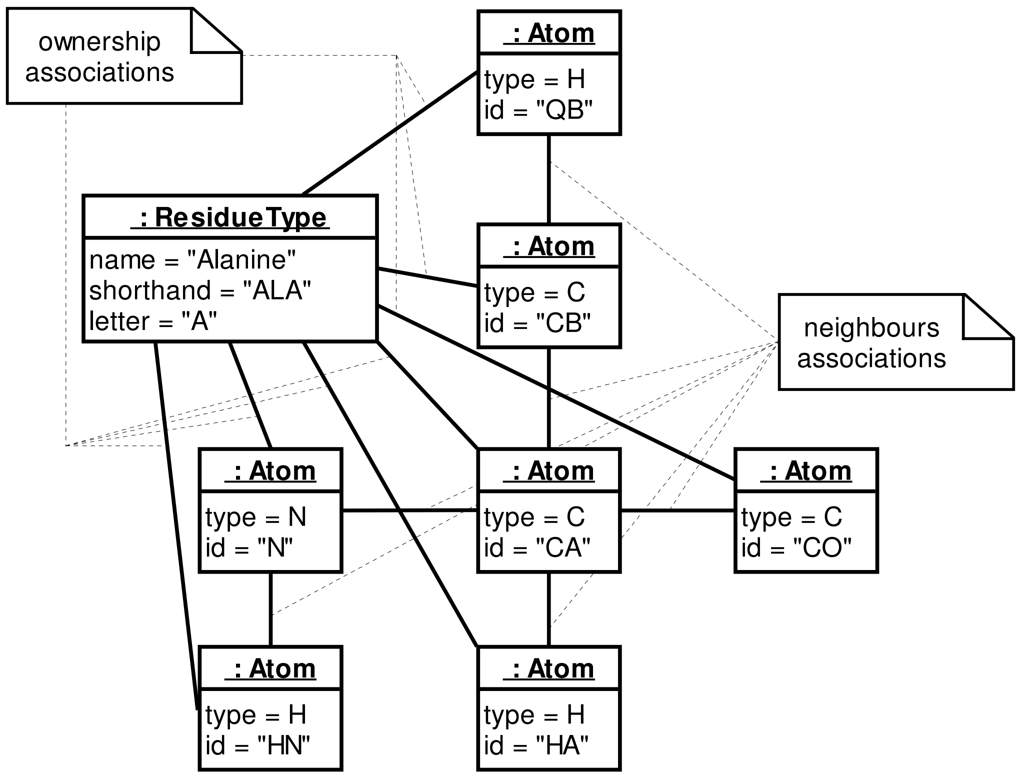


Figure 3: Object diagram of Alanine

Figure 4 introduces the class *SystemType*, which allows the optional classification of a *ResidueType*. Spin system types are visible as characteristic patterns in a COSY or TOCSY type spectrum. Using the homonuclear assignment strategy, spin systems and their respective types are identified in order to find the corresponding *ResidueType* (see chapter 6.6). As an example the Alanine in Figure 3 could be classified by a *SystemType* with a *name* attribute of "A3X".

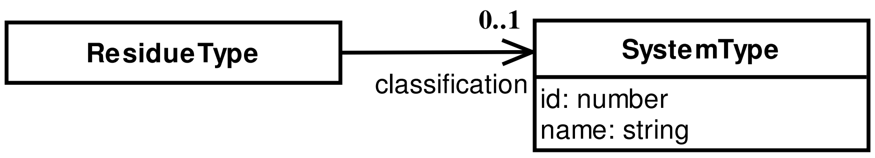


Figure 4: Spin system type classification for ResidueTypes

As said before, proteins are macromolecules consisting of a linear sequence of amino acid residues as uniform building blocks. We therefore introduce the classes *Sequence* and *Residue* to model this primary structure.

A *Residue* is an instance of a *ResidueType* (to which it points). Its only attribute is an arbitrary ordinal number, used to identify and order the *Residues* in the chain. Each

Residue knows its sequential neighbors by means of the *predecessor* and *successor* associations, which are redundantly derived from the *id* attributes. The condition in Eq. 3 thus always holds (where A and B are *Residues*).

Eq. 3

$$A = predecessor(B) \Leftrightarrow B = successor(A)$$

The *Sequence* class is only used for organizational purpose to hold the *Residues*.

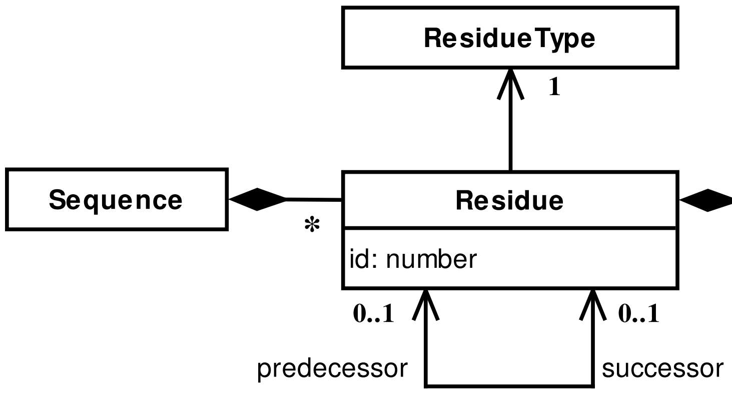


Figure 5: A Sequence of Residues

The following object diagram gives an example how to model a peptide chain of five amino acids. As one can see, *ResidueTypes* are reusable.

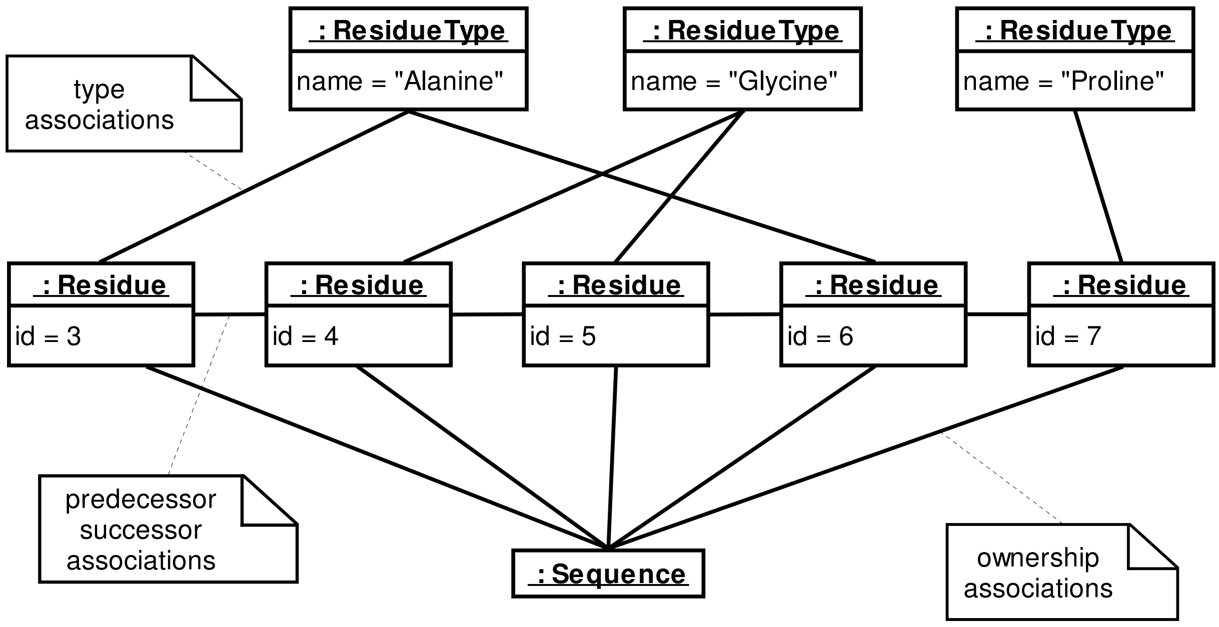


Figure 6: Object diagram of peptide chain

Each *Residue* can have as many *Distributions* as needed. Each is identified by a unique *id*, analogous to the corresponding attribute in *Atom*. The purpose of *Distribution* is to override the random coil distributions defined in the *Atoms* of the referenced *ResidueType*. So each *Residue* could have its own distribution values, e.g. to model secondary structure effects. Their use is described in chapter 4.2.2. Even if the model is designed for macromolecules consisting of uniform building blocks, it can also handle "monolithic" molecules by assuming a sequence length of one.

4.2 Modeling Spin Systems and Assignments

During an NMR experiment the spins of the atoms of the sample molecule are excited by radio frequency pulses. The spins then begin to radiate themselves inducing an electric current in a detection coil, which is the primary observation in an NMR experiment [Wüthrich 1986]. Since the spins are the origin of the observed signals, we here introduce the class *Spin* (see Figure 7). The goal of the assignment process is to find the *Spins* (i.e. their signals) in an NMR spectrum and to eventually associate them with their corresponding *Atoms*.

A *Spin* is uniquely identified in the model by an ordinal number (attribute *id*). It has at least one associated PPM *shift*. Also the corresponding *AtomType* is immediately known from the NMR experiment. In a 1D NMR spectrum virtually every observable peak (i.e. frequency line) corresponds to one or more *Spins*. Due to limited resolution of NMR spectra, structure inherent effects (e.g. circular movements or equal magnetic shielding of atoms) and the growing size of the molecules studied, there is an increasing probability that more than one *Spins* are observed at the same chemical shift (which is called *degenerate chemical shift*). Many of these degeneracies can be resolved using multidimensional NMR experiments, where each peak represents a relation of the *Spins* associated with the chemical shifts of the corresponding dimensions (also called *cross-peak*). In contrast to earlier approaches [Bartels et al. 1995] we explicitly refrain from introducing an explicit class for peak representation here, since it would introduce redundant information into the model and cause an unneccessary dependence on the spectra concretely used. In chapter

4.3.1 we show, how this information can be generated using the conceptual models of residue types and NMR experiments.

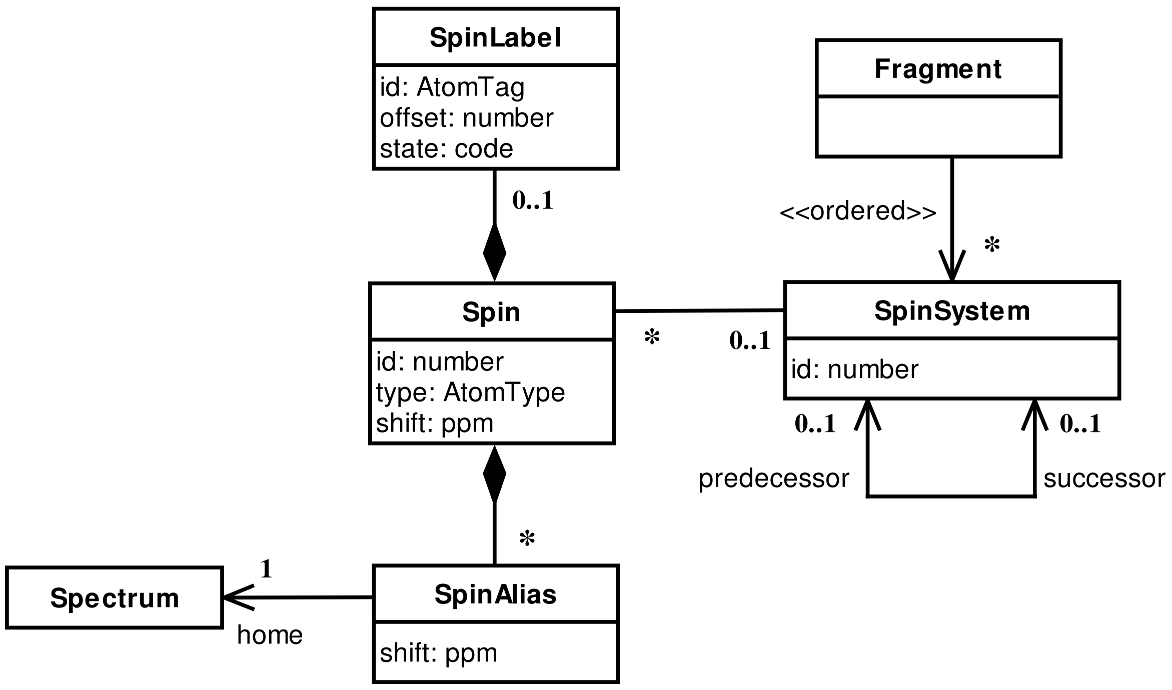


Figure 7: Spins and their classification

It happens in practice, that a given spin appears at varying chemical shifts in different spectra. This is often due to unintended variations in pH or temperature. Often one would like to observe the movement of a spin along a series of spectra (e.g. to track chemical shift changes during a pH titration as shown in chapter 6.8). Both cases can be represented by the class *SpinAlias*, which provides a *Spin* with an alternative *shift* for arbitrary spectra. *Spin* owns a list of *SpinAlias*, each referencing a *Spectrum* (a Spectrum must be unique within this list).

A *Spin* will eventually be associated with a certain *Atom* of a *Residue* (or a *ResidueType* to be precise, see chapter 4.1). This *assignment* is modeled by a *SpinLabel*. It has an *id* attribute, by which the association is established (by equality to the *id* of the *Atom*). The two remaining attributes are explained later.

Spins related by cross-peaks can be seen as a graph [Wilson 1972]. In a [^{1}H, ^{1}H]-COSY spectrum, cross-peaks can only be observed between hydrogen spins separated by up to three covalent bonds. A graph of these *Spins* can therefore span not more than one *Residue* in case of a protein sample. Such a graph is called a spin system [Wüthrich 1986] and represented by the class *SpinSystem* in our model.

SpinSystems are identified by an ordinal number and primarily used to group *Spins*. A *Spin* can belong to exactly one *SpinSystem*.

Spin system graphs have topologies characteristic for the types of the residues they span. This feature is used by the homonuclear assignment strategy to identify and categorize spin systems [Wüthrich 1986]. We already introduced the class *SystemType* in chapter 4.1, which represents such a characteristic topology.

Some residue types (e.g. Histidine or Tryptophan) give rise to two or even three disconnected spin systems representing the same residue. In our model we regard these to belong to the same disconnected graph and therefore to the same *SpinSystem*. We even extend the concept and allow a *SpinSystem* to also contain projections of *Spins* of sequentially neighboring *SpinSystems*. These are identified by an *offset* attribute of *SpinLabel* unequal to zero.

We now can also explain the third attribute *state* of *SpinLabel*. Its values correspond to the symbols *Draft*, *Assigned* and *Finalized*. The *Assigned* state means, that the spectroscopist is sure about the assignment, but the spins are not yet stereo-specifically assigned. *Finalized* is the final state of all *SpinLabels* after the structure calculation has finished and stereo-specific assignments have also been accomplished. The model is consistent, if all *Assigned* and *Finalized Spins* have unique *SpinLabels* within their corresponding *SpinSystems* (comparing the *id* and *offset* attributes of the *SpinLabels* of all *Spins* of the *SpinSystem*). *SpinLabels* can be written as ASCII string using the following EBNF rule:

Eq. 4

$$SpinLabel ::= [\,?\,|\,!\,]\{letter \mid digit\}[(+\mid-)\{digit\}]$$

The attentive reader might have noticed that there is no model element representing the cross-peak relations between spins. The expected cross-peaks can be calculated as shown in chapter 4.3 and 4.3.1.

In the course of the assignment procedure the spectroscopist tries to sequentially connect *SpinSystems* by cross-peaks detected in NOESY or triple-resonance type spectra. Two *SpinSystems* A and B are connected if their corresponding *predecessor* and *successor* attributes point to each other (whereas *predecessor* points to the N-terminus and *successor* to the C-terminus of a amino acid sequence). The model is consistent if the condition in Eq. 3 always holds (with A and B being *SpinSystems*).

Chains of as such connected *SpinSystems* are also called *Fragments* (a *Fragment* is calculated on demand from *SpinSystem* chains). A SpinLabel with an offset of -1 therefore identifies a *Spin* being a projection from the left neighbor *SpinSystem* of the *Fragment*.

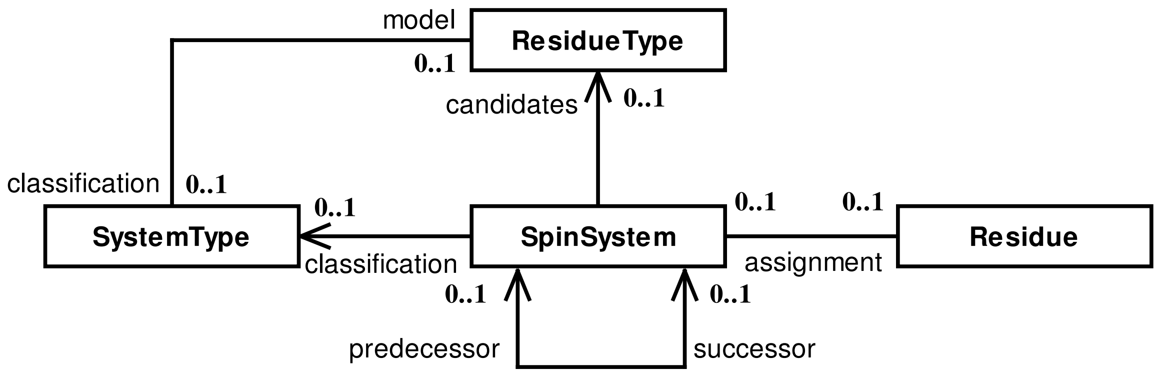


Figure 8: Sequential assignment

The sequential assignment strategy tries to build *Fragments* as long as possible and then to uniquely map them on the *Sequence*. A *SpinSystem* is assigned, if it references a *Residue*. The model is consistent, if the condition in Eq. 5 always holds (whereas A and B can either be a pair of *SpinSystems* or *Residues* and assuming Eq. 3 holds). This implies that a *Fragment* must always be assigned as a whole and match already assigned Fragments, if they are adjoining.

Eq. 5

$$A = predecessor(B) \Leftrightarrow assignment(A) = predecessor(assignment(B))$$

The following chapters show, how pairs of *SpinSystems* are matched and how *Fragments* are mapped on the *Sequence* (using *SpinLabels*, *SystemTypes* and *ResidueType* candidates).

4.2.1 An Algorithm for Matching SpinSystems

As shown above a *SpinSystem* can contain two sorts of *Spins*: locals and projections from neighboring *SpinSystems*. Projection *Spins* are usually discovered during sequential backbone assignment using spectra like 3D HNCA [Grzesiek et al. 1992], 3D HNCACB [Wittekind et al. 1993] and 3D ^{15}N-ed. [^{1}H,^{1}H]-NOESY [Fesik et al. 1988]. For example in a HNCA one can usually recognize two peaks per *strip* [Bartels et al. 1995] (i.e. *SpinSystem*), the stronger representing the local CA *Spin* and the weaker representing the CA *Spin* of the left neighbor *SpinSystem* (in N-terminus direction). The latter is a projection and thus labeled as CA-1. As soon as the *Spins* are attached with labels (i.e. assigned to an *Atom*), an algorithm can try to find pairs of matching *SpinSystems* and combine them to *Fragments*. The following object diagram shows three *SpinSystems* with seven labeled *Spins*. The *SpinLabels* are displayed as *label* attributes using the syntax of Eq. 4.

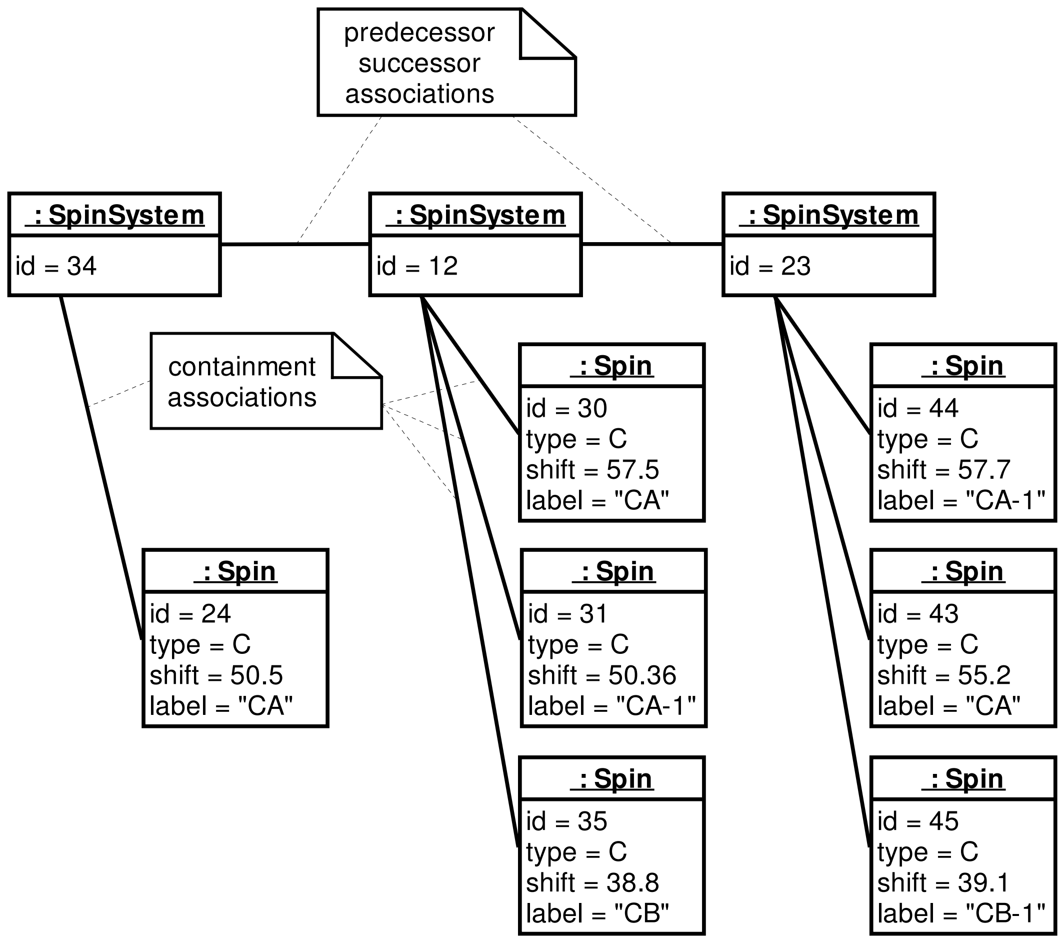


Figure 9: Matching SpinSystems using SpinLabels

SpinLabels provide the matching algorithm with more "intelligence" than earlier approaches based on peak comments, intensities and chemical shifts only (that's why they're also called *smart* spin labels). In Figure 9 it could for example find a match between *Spins* 44 and 30, 45 and 35 as well as 31 and 24, because they have equal *AtomTypes*, *AtomTags* and reasonably close chemical shifts. Due to the -1 offset the algorithm was able to arrange the *SpinSystems* as shown. The concept also works with larger inter-residual offsets usually seen in NOESY spectra.

The match is based on the agreement of the compared chemical shifts, weighted by the number of *Spins* involved. For the agreement we use a simple triangle function (Eq. 6), which seems to give the best efficiency/effectiveness tradeoff. More computation intensive functions also used in literature [Herrmann et al. 2002a] didn't render a higher decision quality.

Eq. 6

$$agreement(\omega, \omega_{ref}, \omega_{tol}) = \begin{cases} 1 - \frac{|\omega - \omega_{ref}|}{\omega_{tol}} & if\, |\omega - \omega_{ref}| \le \omega_{tol} \\ 0 & otherwise \end{cases}$$

In Eq. 6 ω_{ref} is the chemical shift of the reference Spin; ω_{tol} is an arbitrary tolerance, individually adjustable for each *AtomType* by the user. In practice it is a good strategy to start with large ω_{tol} in the beginning of a backbone assignment (e.g. 0.5 PPM for ^1H) and to incrementally reduce it as more *Spins* are assigned (i.e. attached by a label).

Algorithm 1 calculates the *fitness* value of the match of the two *SpinSystems* (parameters *refSystem* and *testSystem*) by summing up the agreement of all *Spins* having an equal *AtomTag* and an appropriate *offset*. The parameter *strict* controls whether the match should be dismissed if one of the *Spins* didn't contribute to the fitness value. The multiplication of *sum* and *weight* increases the significance of many matching Spins with low *fit* compared to a single match with a high *fit*, and thus imitates the assessment manually done by a spectroscopist.

Using this algorithm, a list of all *SpinSystems* matching a reference *SpinSystems* ordered in decreasing order of the calculated *fitness* can be presented to the user (e.g. the list of all possible predecessors if *offset* = -1).

Algorithm 1: Agreement between two SpinSystems

```
parameters: refSystem, testSystem, offset, strict
variables: zeroCount = 0, weight = 0, sum = 0, fit
constants: tolerance = 0.5

forall r in refSystem.spins do
  if r.label.offset = offset then
    forall t in testSystem.spins do
      if t.label.id = r.label.id and
         t.label.offset = 0 then
        fit = agreement( t.shift, r.shift, tolerance )
        sum = sum + fit
        if fit = 0 then
          zeroCount = zeroCount + 1
        else
          weight = weight + 1
        end
      end
    end
  end
end
if strict and zeroCount > 0 then
  return 0
else
  return sum * weight
end
```

Variations of Algorithm 1 exist to match all *Spins* independently of their *SpinLabels* (only matching their *AtomTypes*), or to only use one or more reference *Spins* instead of a complete *SpinSystem*. An application of these algorithms is described in chapter 6.4.

Instead of only matching one reference *SpinSystem*, a complete cross correlation of all *SpinSystems* can be calculated. The correlation matrix could then be used to build all possible *Fragments* and to assess their qualities by mapping them on the Sequence (see the following chapter).

4.2.2 An Algorithm for Mapping Fragments on the Sequence

Sequential assignment using triple-resonance experiments can make use of the fact, that the CA and CB of amino acids have quite a characteristic chemical shift distribution along all studied proteins [Grzesiek et al. 1991]. *Atom* and *Distribution* therefore have an optional attribute pair *mean* and *deviation*, with which one can realize a fuzzy pattern matching between *SpinSystem* and *Residue*. The algorithm tries to find a *mean/deviation* pair for every local *Spin* of the selected *SpinSystem*. It uses the *id* attribute of the *SpinLabel* (assuming it is available) to first search the *Distributions* of the *Residue* and then, if not yet successful, the *Atoms* of the referenced *ResidueType* (see chapter 4.1). If the search was successful, the agreement between the *Spin* and the *Distribution* can be calculated. This can either be done using functions known from error estimation theory (as done in the program MAPPER [Güntert et al. 2000]), where the total error is calculated and minimized, or with the concepts of fuzzy logic [Rouvray et al. 1997] in mind. We decided in favor for the latter for performance reasons and because logical operations can be consistently applied. The agreement is expressed by either fuzzy TRUE or FALSE or an arbitrary value between. Eq. 7 shows an obvious function, but Eq. 6 has shown to be precise enough and well performing (using $\omega_{mean} = \omega_{ref}$ and $\omega_{dev} = \omega_{tol}$ from page 35).

Eq. 7

$$agreement(\omega, \omega_{mean}, \omega_{dev}) = e^{-\frac{1}{2}\left(\frac{\omega-\omega_{mean}}{\omega_{dev}}\right)^2}$$

Algorithm 2 is used to calculate the *fitness* of the match between a *SpinSystem* and a *Residue* (parameters *system* and *residue*) by summing up the agreement between all *Spins* and the corresponding *Distribution* (found by equality of *AtomTag*). The parameter *strict* controls whether the match should be dismissed if one of the *Spins* didn't contribute to the fitness value. A *Spin* for which no *Distribution* can be found doesn't influence the result.

Algorithm 2 as shown is a bit simplified. In the current implementation only *Spins* with *Assigned* or *Finalized SpinLabels* are used for comparison, and already assigned *Residues* can optionally be excluded. Furthermore the match will be discarded, if the *SpinSystem* has a non-empty list of assignment *candidates*, which doesn't contain the referenced *ResidueType*, or if the *SpinSystem* as well as the *ResidueType* both have an attached, but different *SystemType*.

Algorithm 2: Agreement between a SpinSystem and a Residue

```
parameters: system, residue, strict
variables: zeroCount = 0, weight = 0, sum = 0, fit, dis

forall s in system.spins do
  if s.label # null and s.label.offset = 0 then
    dis = residue.findDistribution( s.label.id )
    if dis # null then
      fit = agreement( s.shift, dis.mean, dis.deviation )
      sum = sum + fit
      if fit = 0 then
        zeroCount = zeroCount + 1
      else
        weight = weight + 1
      end
    end
  end
end
if strict and zeroCount > 0 then
  sum = 0
end
return sum, weight    -- two values are returned
```

A given Fragment can then be matched to all possible positions on a *Sequence*. This is efficiently done by constructing an n x m adjacency matrix containing all n *SpinSystems* of the *Fragment* as rows and all m *Residues* of the *Sequence* as columns, and then applying Algorithm 2 to each cell being covered. We can then calculate the total fitness F_k of each position k of the *Fragment* on the *Sequence*, where k runs from 1 to m - n. The cells contain the *sum* $s_{i,k}$ and the *weight* $w_{i,k}$ calculated with Algorithm 2). The spectroscopist can either choose Eq. 8 (which could be seen as a fuzzy OR-relation between the elements of the *Fragment*) or Eq. 9 (in the sense of a fuzzy AND-relation) to calculate F_k. The value w_k is the maximum

$w_{i,k+i}$ with i running from 1 to n. The division by w_k again shall imitate the way a spectroscopist would manually assess the overall quality of a match k.

Eq. 8

$$F_k = \frac{1}{n}\sum_{i=1}^{n}\frac{s_{i,k+i}}{w_k}$$

Eq. 9

$$F_k = \sqrt[n]{\prod_{i=1}^{n}\frac{s_{k,i+i}}{w_k}}$$

All m - n matches can then be arranged in descending order. Usually only matches with $F_k > T$ are kept (T being an adjustable threshold).

4.3 Modeling NMR Experiments

An NMR spectrum is the result of an NMR experiment (see chapter 2.1 and 2.2). It can be seen as a n-dimensional matrix with each cell containing an intensity value [Eccles et al. 1991]. Cells are indexed by ordinal numbers, each representing a PPM position.

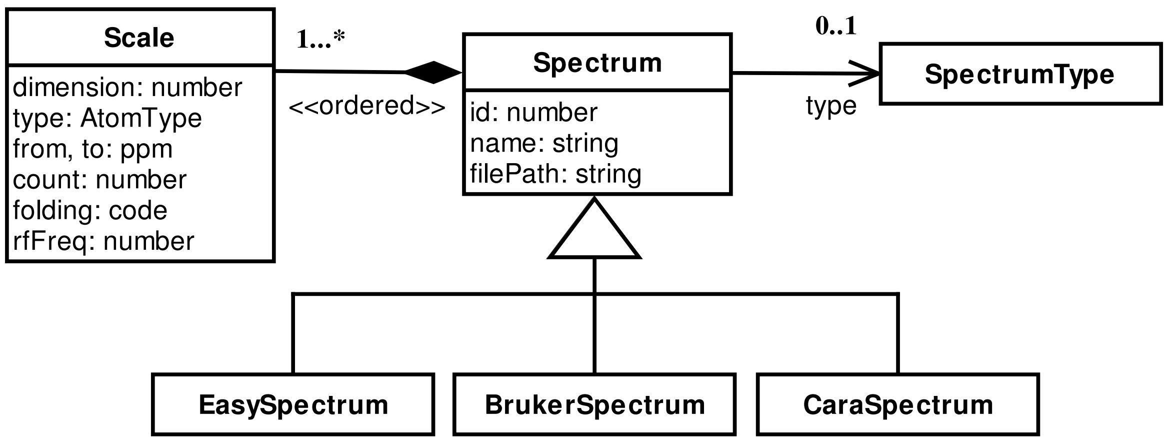


Figure 10: Model of an NMR spectrum

We introduce the two classes *Spectrum* and *Scale* (Figure 10) to model NMR spectra and their dimensions. A Spectrum owns as many Scales as it has dimensions. Each

Scale is uniquely identified by its *dimension* attribute (values 1 to number of dimensions). The attributes *from* and *to* define the PPM value at index 1 and index *count* respectively, where *count* is the number of cells along the dimension. The *folding* attribute can receive one of the symbols *Unfolded*, *Translated* (e.g. for RSH) and *Mirrored* (e.g. for TPPI). It controls how the dimension is presented outside its original PPM range. The *rfFreq* contains the carrier frequency of the spectrometer for the dimension. It can be used to convert PPM to Hz and vice versa. Each *Scale* is associated with an *AtomType* indicating the type of nuclei causing the signals observable along the dimension.

A *Spectrum* is a generic concept used to shield the model from the implementation details of specific spectrum formats (shown as sub-classes). Spectra are usually kept in files, accessible by the attribute *filePath*. Within the model a *Spectrum* is uniquely identified by an ordinal number. The features of an NMR spectrum are determined by the type of NMR experiment by which it was generated. This *type* is represented in the model by the class *SpectrumType* (see Figure 11). If a *Spectrum* references a *SpectrumType*, a wide range of algorithms can be applied to it (e.g. path simulation and peak inference, see below).

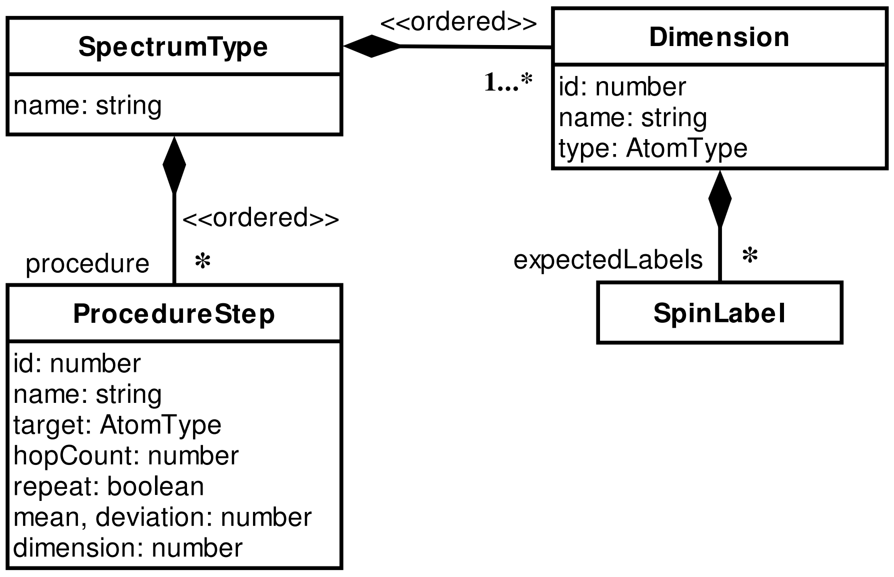


Figure 11: Classification of NMR spectra

SpectrumType is identified by a unique name (e.g. "HNCA" or "HSQC"). It offers two complementary concepts to describe the features of an NMR experiment: explicit specification of expected *SpinLabels* or of the magnetization procedure (to perform peak inference, see below).

Spectrum and *SpectrumType*, and therefore *Scale* and *Dimension* form complementary concepts, in that the former can be seen as an instance of the latter. *Dimension* has the same *id* and *type* attributes like *Scale*. The *name* attribute is only decorative.

Each *Dimension* can be associated with an explicit set of expected *SpinLabels*. This is particularly important for the specification of *SpectrumTypes* used for the sequential assignment, i.e. before the *ResidueType* of a *SpinSystem* is known. The concept of *SpinLabels* was already introduced in chapter 4.2. The following figure shows a HNCA *SpectrumType* as an example. It can immediately be seen that *Dimensions* 1 and 3 have unique *SpinLabels*. The model can thus automatically infer, which *Dimensions* (i.e. which *Spins*) represent independent *SpinSystems* and should be used as strip anchors. In the example the model would present all *SpinSystems* containing "HN" and "N" labeled Spins as strips. The strips would then automatically accept and display the "CA" and "CA-1" *Spins* of the corresponding *SpinSystem*. In XEASY [Bartels et al. 1995] this had to be done manually (involving two different peaklists) because of the lack of structured information about the *SpectrumType*. Chapter 6.4 shows an application of this concept.

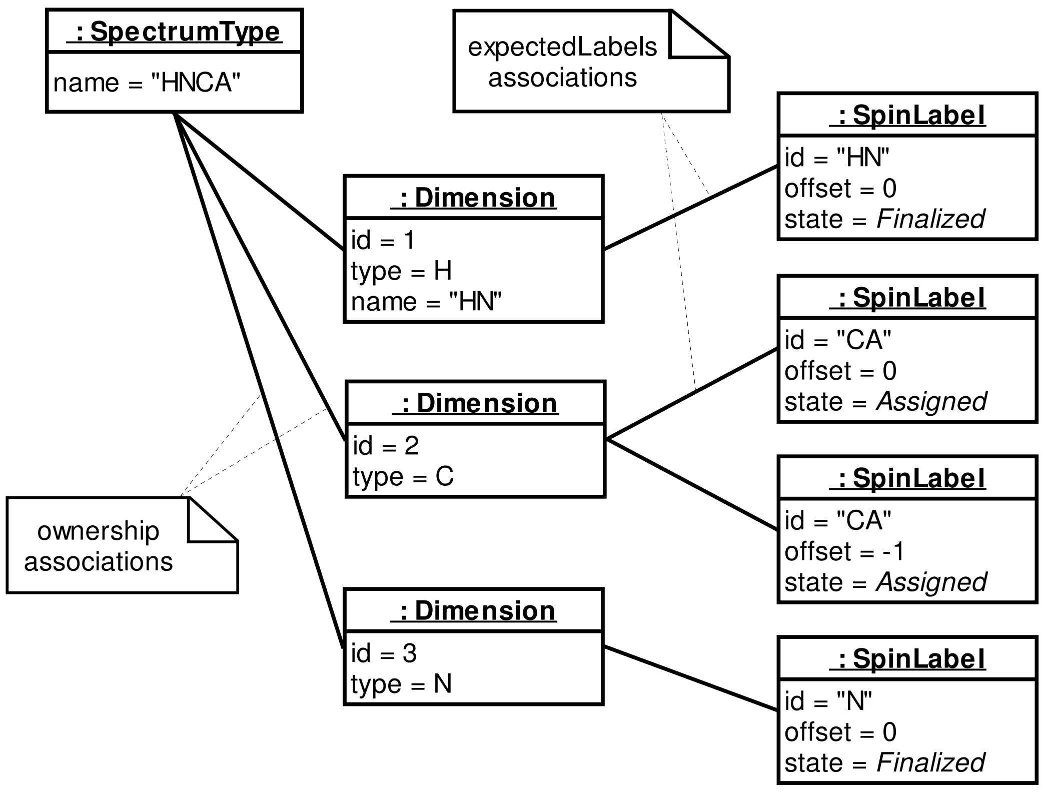


Figure 12: Object diagram of an HNCA SpectrumType

Additionally or alternatively to the explicit specification of expected *SpinLabels*, one can specify a magnetization transfer procedure for a *SpectrumType*, represented by the class *ProcedureStep* in Figure 11. The pulse sequence applied to a sample during an NMR experiment can be subdivided into consecutive periods.

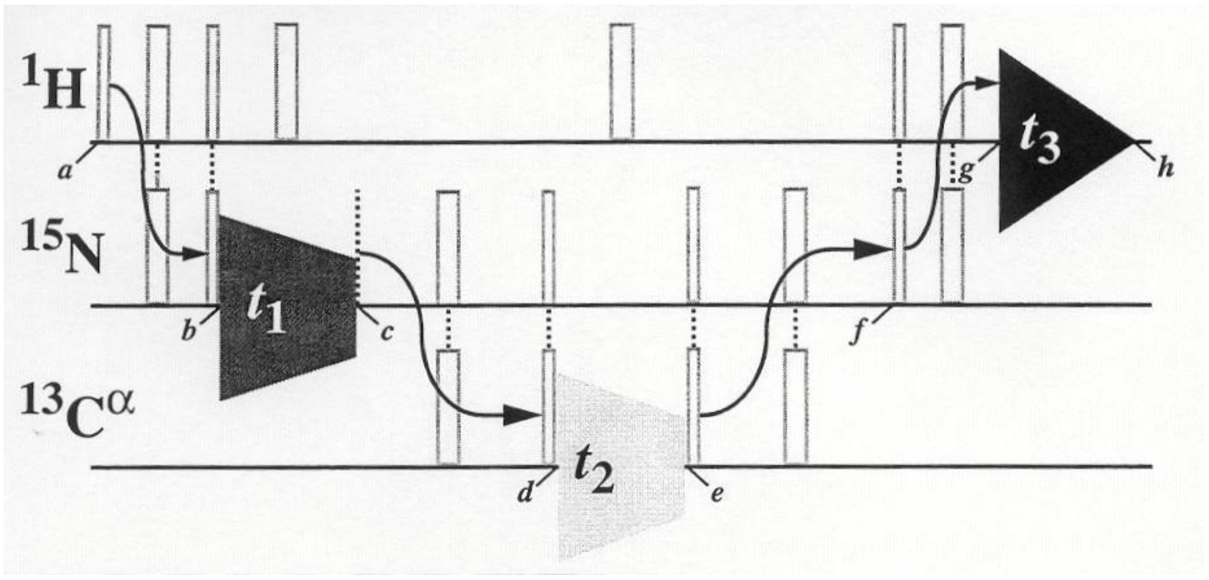


Figure 13: Pulse sequence of an HNCA experiment

Figure 13 shows a typical pulse sequence of a triple-resonance HNCA experiment [Grzesiek et al. 1992]. The following periods can be recognized:

Table 1: The periods of an HNCA experiment

$a \rightarrow b$	^{1}H, ^{15}N-INEPT
$b \rightarrow c$	^{15}N-Evolution
$c \rightarrow d$	^{15}N, $^{13}C^{\alpha}$-INEPT
$d \rightarrow e$	$^{13}C^{\alpha}$-Evolution
$e \rightarrow f$	$^{13}C^{\alpha}$, ^{15}N-INEPT
$f \rightarrow g$	^{15}N, ^{1}H-INEPT
$g \rightarrow h$	^{1}H-Acquisition

The periods shown in Table 1 control the magnetization transfer between the spins of the molecule. Each execution of the pulse sequence running through all periods renders the FID t_3 measured on the ^{1}H channel of the spectrometer which is then digitized and recorded. The two additional dimensions result from systematic incrementation of the two evolution periods t_1 and t_2 as shown in Figure 13 (i.e. from a cross product of all ^{15}N and ^{13}C evolution periods). In addition the phases of the pulses and receiver are varied systematically during several repetitions (scans) of each t_1, t_2 pair to select a specific magnetization transfer pathway and to reject alternative pathways. Figure 14 shows how the magnetization transfer pathway and labeling periods act in concert to correlate the frequencies of specific spins in a protein backbone fragment. Each step in the pathway defined in Table 1 is indicated as a labeled arrow in Figure 14.

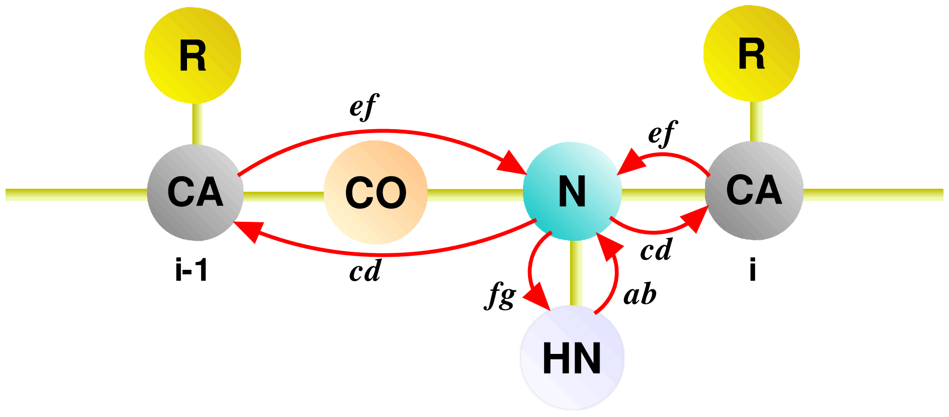


Figure 14: Effect of an HNCA to the backbone

The carrier frequency and excitation range of the ^{13}C pulses during the ^{15}N, ^{13}C$^{\alpha}$-INEPT (c $\rightarrow$ d) are chosen such that CO is not excited and therefore not visible in the spectrum. The CA are expected to have a chemical shift in the range of 30 to 50 PPM, whereas the chemical shift of CO will be around 178 PPM. It can also be seen that a $\rightarrow$ b and f $\rightarrow$ g as well as c $\rightarrow$ d and e $\rightarrow$ f are symmetrical operations; the inverse operations are only necessary for experimental implementation reasons and can be neglected in the model. This magnetization procedure is represented in our model by means of the class *ProcedureStep*, which is as an abstraction of a specific operation like INEPT. Figure 15 shows an example of such a procedure for an HNCA *SpectrumType*.

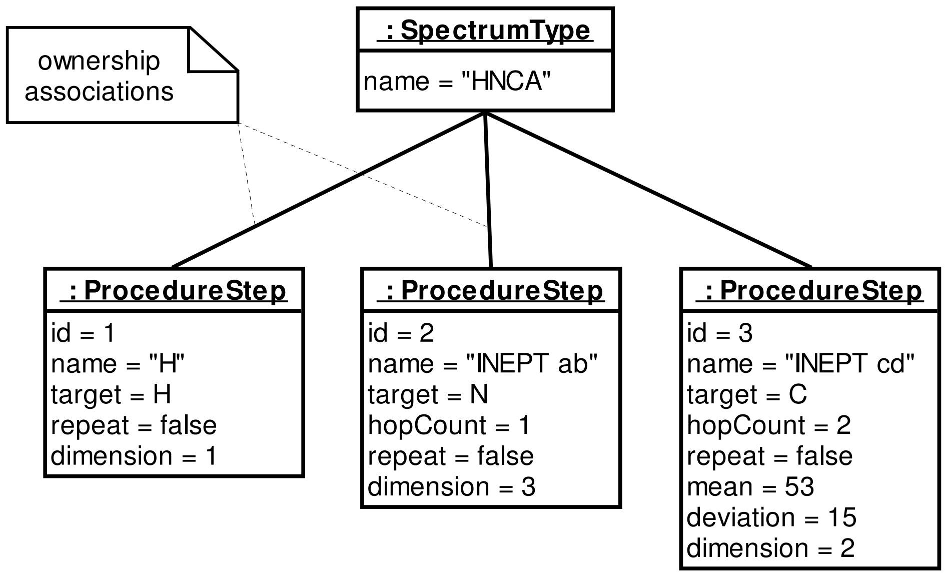


Figure 15: Magnetization transfer procedure of HNCA

A ProcedureStep is uniquely identified by an ordinal number, which also determines the execution order. The name attribute is for documentation purpose only. Each *ProcedureStep* can be seen as a binary relation *P* on the set of all atoms *A* of a given molecule. *P* is a subset of *A x A* and relates each element (i.e. *Atom*) *x* of its domain *D* with an element *y* of its codomain *C* (more formally written as $(x, y) \in P \Leftrightarrow xPy$, where $D = \{x; (\exists y)(x, y) \in P\}$ and $C = \{y; (\exists x)(x, y) \in P\}$). It is defined by the attributes of *ProcedureStep* and can be implemented using a graph search algorithm [Wilson 1972], marking each node when it was visited for the first time. The optional attributes *target*, *mean* and *deviation* can restrict the type and chemical shift of the *Atoms* in the codomain (where the restriction only applies if the attributes are available). The attribute *hopCount* restricts the number of covalent bonds separating an element x from an element y. E.g. if the *hopCount* is 3 only *Atoms* y which can be reached from *Atom* x by stepping through a maximum of three intervening bonds are selected. A *hopCount* of -1 has the special meaning that all *Atoms* of the given *type* in the *ResidueType* are selected (which is used for example to simulate the effect of a NOESY, i.e. to select all H *Atoms* in a *ResidueType* independent of the number of bonds separating them from the starting *Atom* x). If *repeat = true*, the relation is recursively applied to itself until all elements of the domain have been visited (which is what happens in a TOCSY step). If a *dimension* attribute is specified, the *ProcedureStep* applies to a *Dimension* of the *SpectrumType*, which results in detected signals. A magnetization transfer procedure can therefore contain steps which are not directly visible in a spectrum (as is the case e.g. in (H)CCH-TOCSY).

In chapter 5 we show that the concept presented here is comprehensive enough to model most commonly used types of NMR experiments (except projected experiments, which have to be substituted by their non-projected counterparts).

4.3.1 Simulation of Magnetization Transfer and Peak Inference

In this chapter we show how the concepts of *SpectrumType* and *ProcedureStep* can be used to simulate the magnetization transfer pathways of NMR spectra, when applied to a given *ResidueType*. The execution of an NMR spectrum is represented in the model by the class *Experiment* (Figure 16). It references the *ResidueType* and *SpectrumType* with which it was executed. To simulate all pathways of a complete protein we therefore need as many *Experiments* as the sequence contains different *ResidueTypes* (e.g. tree for a peptide shown in Figure 6).

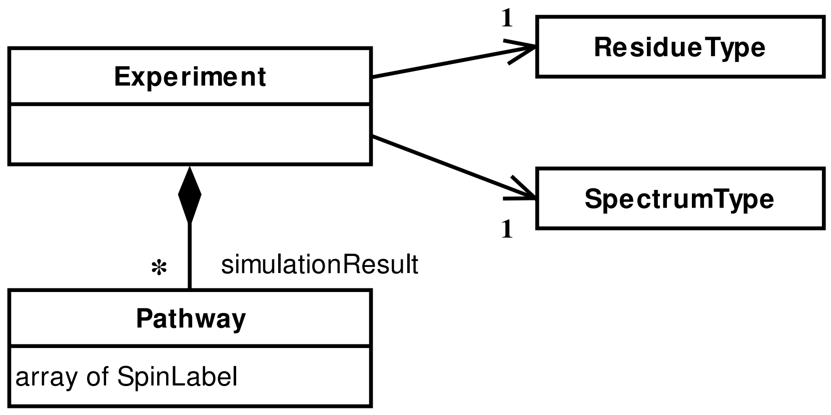


Figure 16: Magnetization Transfer Pathway table of an Experiment

The results of the simulation are kept in the class *Pathway*, each containing a trace (i.e. projection) of an observed magnetization transfer path.

The set of all magnetization transfer pathways along a molecule caused by an NMR experiment can be represented by a tree. The root of the tree corresponds to the start of the experiment (see Figure 19). Each edge connecting the nodes *x* and *y* is an element of the relation *P*, where $(x, y) \in P \Leftrightarrow xPy$. In the previous chapter we claimed the binary relation *P* to correspond to a *ProcedureStep*, which thus represents all edges connecting two levels of the tree (where level *i - 1* is the domain and level *i* the codomain of relation P_i, taking the root as level *0*). A single magnetization transfer pathway is a chain of nodes starting at a node of level 1 and ending at a leaf node of the tree. Only chains of length *n* are considered to belong to the result of the simulation, where *n* is the number of *ProcedureSteps* of the *SpectrumType*. All chains are then combined into a table with *n* columns, each chain forming a separate row. The chains are thereby aggregated, i.e. only one of a series of equal chains (equal regarding the *AtomTags* of the nodes) is kept as a row, so all rows of the table are distinct. The columns are then reordered according to the *dimension* attribute of the *ProcedureStep*, removing the *n - d* columns of ProcedureSteps with an empty *dimension* attibute. The columns of the remaining table thus represent the *Dimensions* of the *SpectrumType*. A row of this table is regarded as a projection of a magnetization path and kept in the class *Pathway*. This transformation of a tree into a table is called a *denormalization*.

If the *ResidueType* is not known or the *SpectrumType* doesn't contain *ProcedureSteps*, the denormalization table can also be built from a cross product of all *SpinLabels* of all *Dimensions*, which is useful during sequential assignment. Alternatively one can designate a *ResidueType* of the library to be used as a *generic spin system type*, i.e. to do the pathway simulation for a *SpinSystem* as long as its sequential assignment is not known.

In the following simple example we will do the simulation of a (H)CCH-COSY experiment for a Serine.

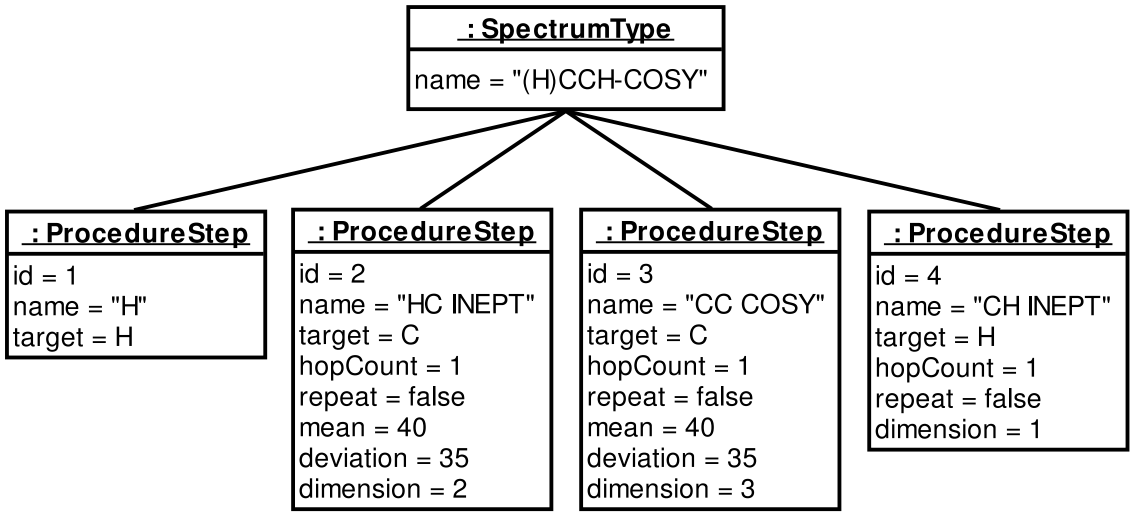


Figure 17: Object diagram of a (H)CCH-COSY SpectrumType

The displayed *SpectrumType* (Figure 17) has three *Dimensions* and four *ProcedureSteps*. Each step can potentially give rise to one Dimension. However not every period of an NMR experiment is detected (e.g. COSY or INEPT steps which only transfer magnetization). This is the case for the (H)CCH-COSY experiment where the first *H* is not detected and thus the first step of the corresponding procedure does not have an associated *Dimension* (i.e. it has no *dimension* attribute). A *SpectrumType* is consistent, if the number *d* of *Dimensions* is less or equal to the number of *ProcedureSteps* and exactly *d* steps disjointly reference a *Dimension*. Note that in our example all *hopCounts* are equal to one, and that the middle two steps restrict the PPM range of the codomain. In Figure 18, the pathways resulting from the procedure of Figure 17 are displayed. In the following the effect of each step is explained.

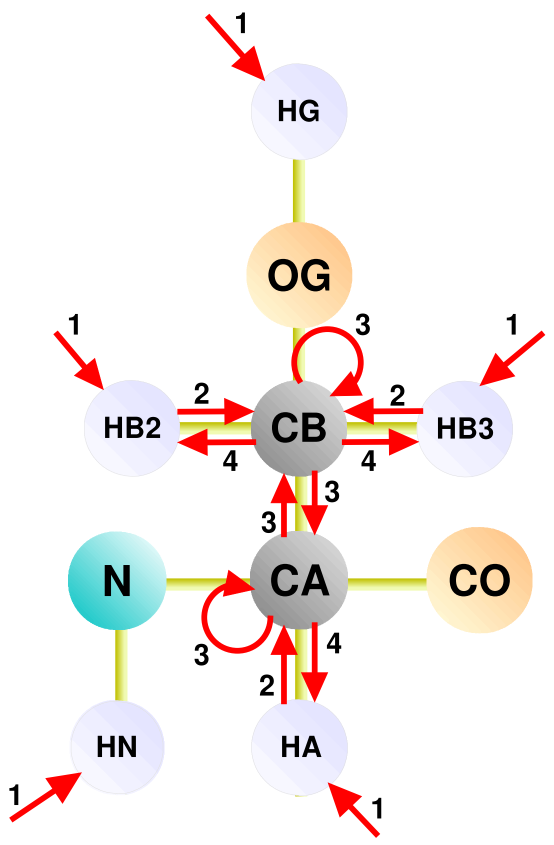


Figure 18: Application of an (H)CCH-COSY to a Serine

1. The domain of the first step corresponds to all *Atoms* of the Serine. Since *target* equals to *H*, the codomain of the relation are all *Atoms* with *type = H*.
2. The codomain of the first step becomes the candidate domain of the second. In the second step we are looking for all *Atoms* within one bond distance, whose *type* equals to *C* and *mean* is in the range of 40 ± 35 PPM. HG and HN don't meet these conditions and their path therefore breaks off.
3. The conditions of step three are identical to step two. CA and CB form the candidate domain for this step. Reflection is allowed, so CA and CB can relate to themselves. However the relation of CA and CO is not possible due to the restriction of the PPM range (i.e. the mean of CO is around 170 PPM).
4. This step tries to relate CA and CB to all *Atoms* with *type = H*. Due to the bond distance restriction (*hopCount* of one) this is only possible with HA, HB2 and HB3. If the step had defined a *hopCount* of two, then HN and HG would belong to the codomain.

Figure 19 shows the corresponding state tree representing all discovered paths through the molecule. The algorithm now prunes all branches of the tree with length smaller than four. After that, a denormalization step follows, where all remaining branches are aggregated into tabular form.

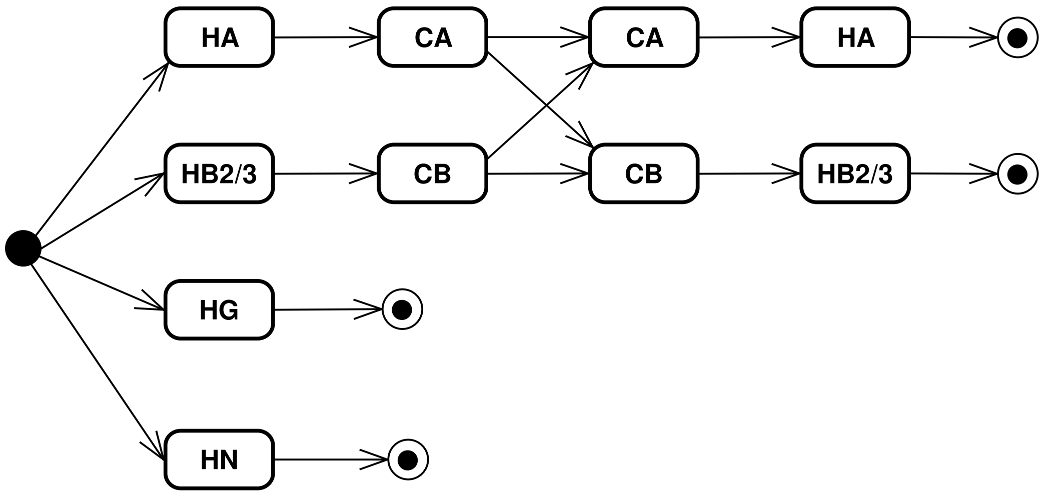


Figure 19: State tree of the experiment

The first column is removed because the first *ProcedureStep* does not have a *dimension* attribute. The columns of this table are then rearranged in the order given by the *dimension* attributes of Figure 17.

Table 2: Experiment pathway table

Dim. 1	Dim. 2	Dim. 3
HA	CA	CA
HB2	CA	CB
HB3	CA	CB
HB2	CB	CB
HB3	CB	CB
HA	CB	CA

Table 2 shows the result of the algorithm. Each row is a potential (i.e. expected) cross-peak, visible on spectra of the given *SpectrumType*. If such a *Spectrum* should be displayed, all *SpinSystems* of the model are searched for *Spins* with a *SpinLabel* corresponding to the *AtomTags* of each column of *Pathway*. We call this *peak* *inference*. If the *hopCount* of a dimension was -1, all *Spins* with an *AtomType* corresponding to the *Dimension* would be selected instead (to model the effect of NOESY steps). Figure 20 shows the object diagram of the *Experiment* corresponding to Table 2.

SpinSystem 12 of Figure 9 contains the two *Spins id=30* and *id=35* with labels CA and CB. If the spectroscopist added another *Spin id = 50* with a HA label to this *SpinSystem*, it would be presented with two cross-peaks on the (H)CCH-COSY, which are defined by the two *Spin* triples { 50, 30, 30 } and { 50, 35, 30 } (corresponding to the first and last row of Table 2). Applications of this algorithm are shown in the chapters 6.5 and 6.6.

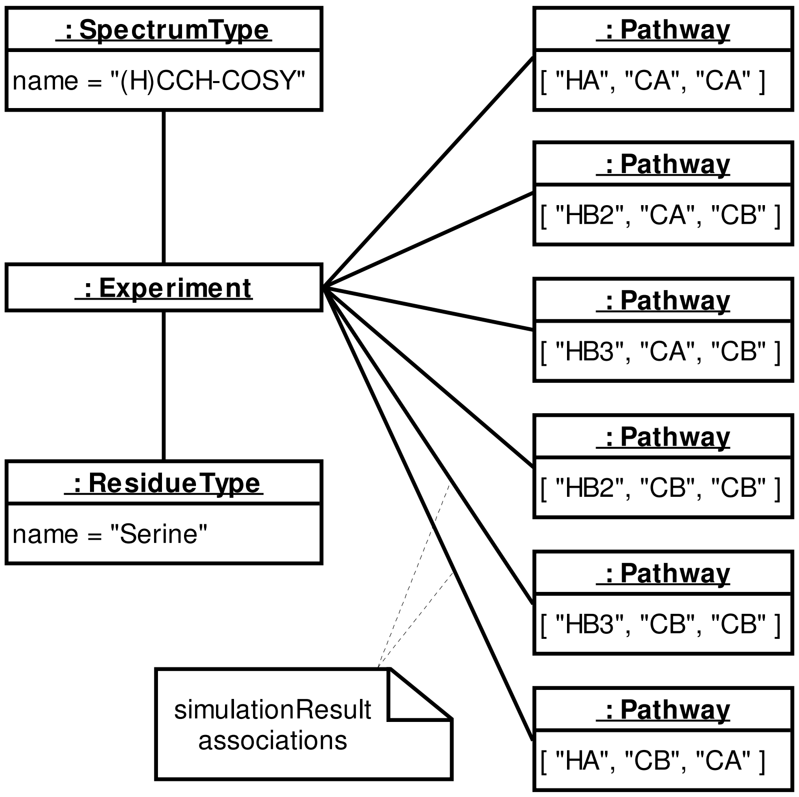


Figure 20: Object diagram of the resulting pathway table

The approach shown evidently offers several major improvements compared to the legacy concept of peaklists. It is no longer necessary to store redundant information and to manage and synchronize peaklists. A peak is nothing but a position vector, unaware of the semantics of the problem domain. Its dimensions are largely redundant, causing the project to become inconsistent if not all vector elements are updated accordingly (e.g. if a "TOCSY tower" moves). This cannot happen anymore with our approach, because *Spins* are independent, reusable objects, which are dynamically arranged in tuples to be presented as cross-peaks (e.g. the two tuples above would be immediately up to date if *Spin* 30 had changed position). Arbitrary *SpectrumTypes* and *ResidueTypes* can be added to the model later without invalidating the consistency or the implementation.

The examples shown in this section applied the pathway simulation to a single *ResidueType* only. But the model also incorporates informations about which *Atoms* of the *ResidueTypes* are responsible to link them to peptide chains. The pathway simulation can thus also cover the neighboring *ResidueTypes*. In the current implementation of the model the simulation only covers the left and right neighbors.

This model has the additional advantage that it avoids the need to peak pick new peaklists for spectra which are recorded during later stages of a project, so that only the new non-redundant information needs to be analysed.

4.4 Modeling NOESY Constraints

In chapter 2.4 constraint gathering and peak volume determination were presented as the final step before structure calculation. In chapter 4.2 a model for *Spin* and *SpinSystem* assignment was introduced, focusing on intra-residual and short-range relations between *Spins*. When the assignment has finished, each *Spin* is assigned to an *Atom* of the *Sequence*, i.e. each chemical shift position on the axis of an NMR spectrum can be explained (ideally speaking). The goal of constraint gathering is to find medium- and long-range relations between arbitrary *Spins*. Because all *Spins* are addressable by their *Residue* membership, one could in principle use the already introduced *SpinLabels* to also describe a long-range relation (i.e. by adding a redundant projection *Spin* and labeling it with an *offset* pointing to the sequence position of the original). But on the one hand a relative addressing scheme for arbitrary relations seems to be unpractical to handle. On the other hand some important requirements could not be properly met by this approach (e.g. associating a distance constraint with a volume, see below). We therefore introduce the class *SpinLink* to model such a relationship (see Figure 21).

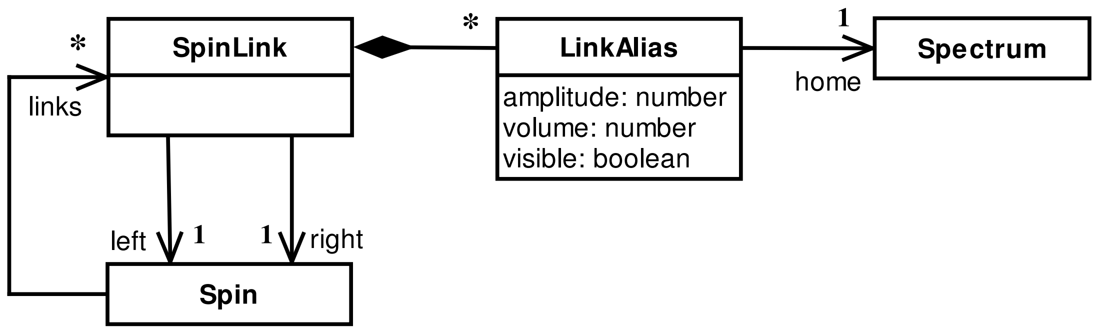


Figure 21: Relations between arbitrary Spins

A *SpinLink* is an independent entity referencing the two *Spins* it relates. Each Spin has a list of the *SpinLinks* it is referenced by (either as *left* or *right*, see Figure 22).

Each peak on a 2D [^{1}H,^{1}H]-NOESY type spectrum represents a distance constraint between the *Spins* at the particular chemical shift positions of the peak. Furthermore the upper distance limit of this constraint is a function of the intensity volume of the peak . The class *LinkAlias* is used to maintain this information. Since an intensity is a property of the specific *Spectrum* it is observed, *SpinLink* owns a *LinkAlias* for each of them. The attribute *amplitude* holds the intensity directly measured at the corresponding peak position, whereas the *volume* is the result of an either manual or automatic volume determination of the intensity curve covering the area of the peak (considering overlap). Chapter 4.5.1 shows an effective algorithm for automatic volume determination.

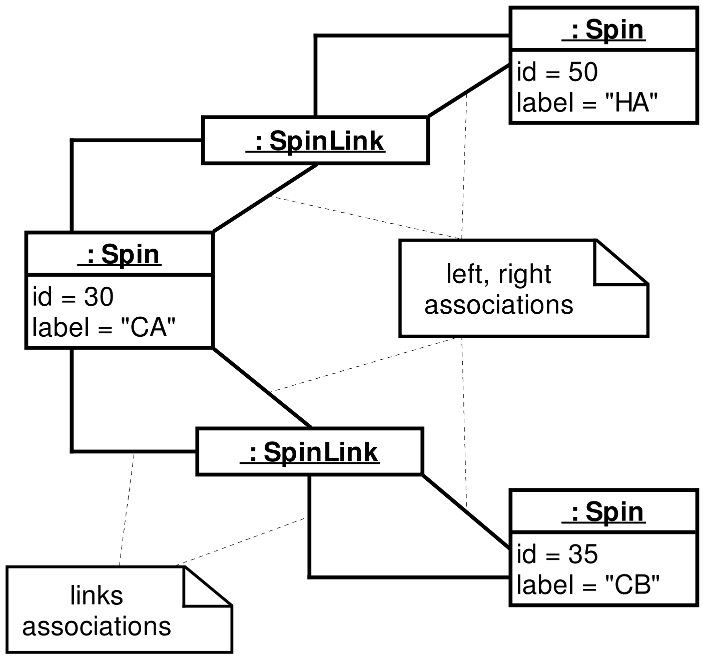


Figure 22: Object diagram of SpinLinks

One could argue that a SpinLink is identical to the concept of a peak, but this is not true. A peak is nothing more than a PPM position (a chemical shift vector). As we have seen in chapter 4.3.1 a peak represents a trace along a magnetization transfer pathway, i.e. it relates the *Spins* at the chemical shift positions. If we only work with 2D NMR experiments, a peak contains enough information to represent the trace (provided that there is no chemical shift degeneracy), because there is only one possibility to connect two elements. But if we analyze three or more dimensional spectra, a peak becomes ambiguous, i.e. there are *n* possibilities to connect the *n* elements of a peak by a path with length *n - 1*. This ambiguity doesn't apply to *SpinLinks*, because they directly represent the path.

SpinLinks are complementary to projection *Spins* (having *SpinLabels* with *offset* unequal zero), i.e. they can be converted to each other. The concept has not only proved to be useful for constraint gathering, but also for the homonuclear assignment strategy. For this purpose LinkAlias has the attribute *visible* to represent the feature, that only a subset of the cross-peaks of a TOCSY spectrum are visible in a COSY spectrum. An application of these concepts can be found in chapter 6.6.

4.5 Modeling Peaks, Volume Determination and Back-Calculation

Peaklists are a legacy concept integrated into the model for compatibility with other applications (e.g. XEASY [Bartels et al. 1995], DYANA [Güntert et al. 1997], etc.). They are represented in the model by the class *PeakList* (see Figure 23). The class *Peak* is inspired by the format defined by EASY [Eccles et al. 1991] and contains all common attributes like *id*, *label*, *color* and *assignment* (without further semantic meaning).

In contrast to previous applications the model allows a *Peak* to have different positions in different spectra (analogous to *SpinAlias* introduced in chapter 4.2), represented by *PeakAlias*. It has additional attributes to capture the *amplitude* measured and the *volume* calculated (see below).

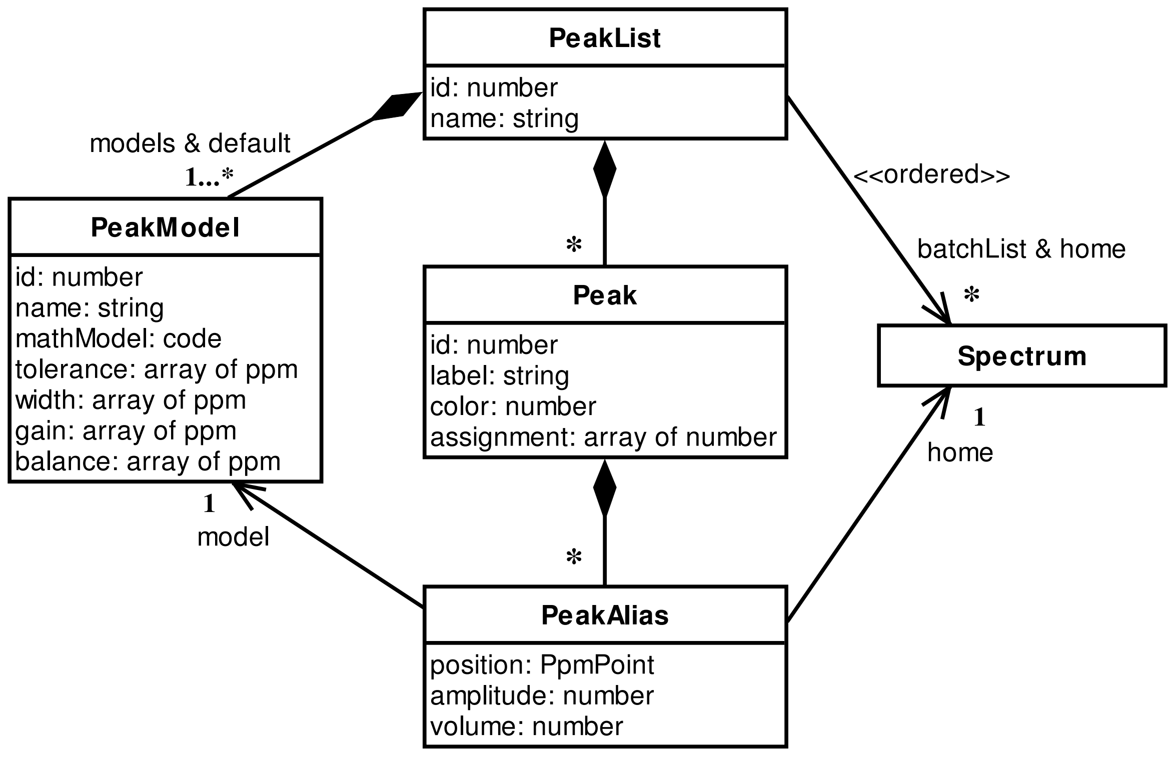


Figure 23: Peaks, PeakLists and PeakModels

The automatic peak volume determination algorithm introduced in the next chapter is supported by the class *PeakModel*, by which each *PeakAlias* can be extended with information about the mathematical model of the peak. *PeakModel* is intentionally associated with the alias and not the peak itself. It is thus possible to follow a Peak along a series of spectra.

A *PeakList* is generally associated with the *Spectrum* it was created for (defined by the *home* association). But it is also possible to associate it with an ordered series of spectra (the *batchList* association). An application of this feature is described in chapter 6.8).

It is at any time possible to generate *PeakLists* from *Spins* and *SpinSystems* (i.e. the concepts introduced in the chapters 4.2, 4.3 and 4.4) in order to pass information to other applications.

4.5.1 An Algorithm for Volume Determination and Spectrum Simulation

An NMR spectrum essentially consists of regular and artifact peaks superimposed by random noise. Even if the artifact peaks sometimes look like regular peaks, they do not belong to the result set expected by the NMR experiment performed. All regular peaks of a NOESY spectrum ideally have similar properties, e.g. peak shape, relative line width and noise floor. In practice these properties will slightly vary for different regions of the spectrum. We assume that all regular peaks of a spectrum can be described by a single, uniform mathematical model (e.g. a Gauss or Lorentz function). We regard a spectrum to be a superposition of noise and instances of this uniform model, one instance for each regular peak. The current model neglects artifact peaks and regional variations of properties.

Figure 24 schematically shows a one-dimensional slice of an NMR spectrum containing two peaks at positions Δ_1 and Δ_2 with the observed amplitudes A_1 and A_2.

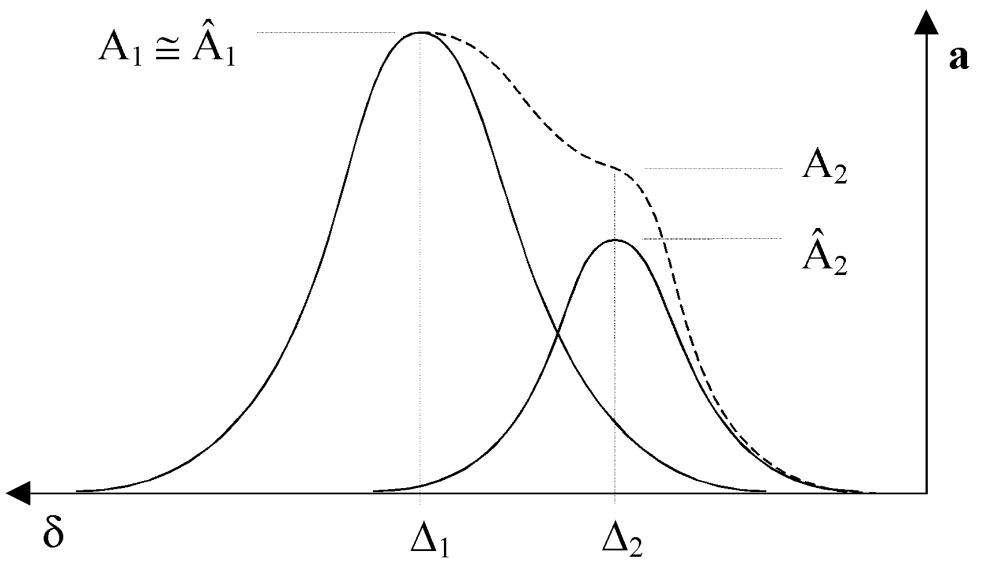


Figure 24: Slice through a schematic 1D NMR spectrum showing two overlapping peaks

The smaller peak Δ_2 partially overlaps with the larger one and is therefore only observable with a superposed amplitude A_2 (hatched line). In contrast to that peak Δ_1 is observable with nearly its original amplitude $\hat{A}_1$. Because of the superposition the original amplitude $\hat{A}_2$ of peak Δ_2 cannot be observed directly (the spectrum shows A_2 instead). This kind of overlapping is a common problem in crowded spectral regions (generally due to limited resolution and increased line width).

The mathematical model of a single peak can be described by the following function prototype:

Eq. 10

$$a(\vec{\delta}) = \hat{A} f(\vec{\delta} - \vec{\Delta})$$

The function f defines the shape of the peak. For each point $\vec{\delta}$ of the n-dimensional spectrum there exists an amplitude a($\vec{\delta}$). The n-dimensional model is usually assumed symmetric about the peak center, i.e. $f(\vec{\delta}) = f(-\vec{\delta})$, and can be constructed as the cross product of n one-dimensional models. As an example model Eq. 11 shows a one-dimensional gaussian function for a peak k at Δ_k with a gaussian shape and line width Γ_k.

Eq. 11

$$f(\delta - \Delta_k) = e^{-\frac{1}{2}(\frac{\delta - \Delta_k}{\Gamma_k})^2}$$

We require the function to be normalized, i.e. the amplitude at the peak center is equal to one (which is fulfilled by Eq. 11). In principle the function can be freely chosen as long as it is normalized and defined for all points of the spectrum.

For the current implementation a superposition of a Gauss and a Lorentz function

with a tunable balance Λ and uniform line width Γ for all peaks was chosen:

Eq. 12

$$f(\delta - \Lambda) = (1 - \Lambda)\frac{1}{1 + (\frac{\delta - \Delta_k}{\Gamma})^2} + \Lambda e^{-\frac{1}{2}(\frac{\delta - \Delta_k}{\Gamma})^2}$$

But other functions seem also to be useful, e.g. $\sin(x)/x$ to compensate for the truncation effect of the discrete Fourier transform.

The observable amplitude A_k of peak Δ_k can now be written as the superposition of the model functions of all n peaks of the spectrum:

Eq. 13

$$A_k = \sum_{i=1}^{n} \hat{A}_i f(\Delta_k - \Delta_i) + N(\Delta_i)$$

If we neglect noise N, we can build a linear equation system with an equation for each of the n peaks of the spectrum. The values of A_k can directly be observed in the spectrum and $f(\Delta_k - \Delta_i)$ is calculated using the chosen model functions. The original amplitudes of $\hat{A}_i$ are the only unknowns. We therefore get a solvable equation system with n equations and n unknowns (as long as the matrix is not singular).

If we now multiply the original amplitudes of all n peaks of the spectrum by their model function $f(\delta - \Delta_i)$ and add up the results (Eq. 14), we get an

ideal backcalculated spectrum, intrinsically without noise or artifacts.

Eq. 14

$$a(\vec{\delta}) = \sum_{i=1}^{n} \hat{A}_i f(\vec{\delta} - \vec{\Delta}_i)$$

This backcalculated spectrum can then be subtracted from the original one. In the ideal case the difference would be zero. But since we neglected noise and artifacts, at least these would remain if we had chosen well our model parameters. If we however still see peak like shapes, we either have not considered enough peaks or the model parameters do not fit well enough. Additional peaks can be picked (or moved), if necessary. In addition, the model properties can be optimized interactively by adjusting the relative line with or balance between Gauss/Lorentz with immediate feedback of the effect to the backcalculated difference spectrum. Yet another benefit is the fact, that we now can get an objective quality measure for the integrated peak list by calculating the percentage of the remaining intensity in the difference spectrum compared to the original spectrum.

Figure 25 shows the class *BackCalculation* representing Eq. 14. It is a subclass of *Spectrum* and thus perfectly integrated into the model.

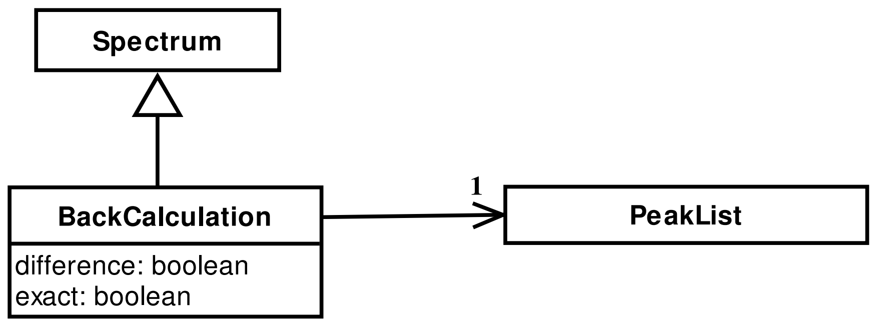


Figure 25: Model of a Spectrum calculated from a PeakList

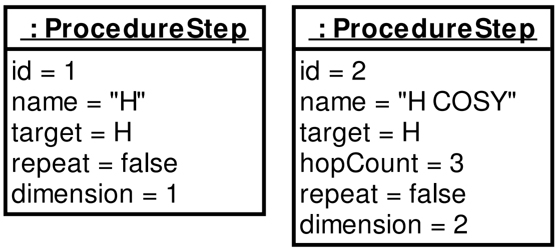
The algorithm proofed to be very robust and easy to apply. Even though (or because) the concept is quite simple, the output quality is good and reliable. The effect of parameter changes is immediately evident and results are reproducible with identical settings. The reason for the robustness is the fact that the algorithm does completely without decomposition or inference from the original spectrum. It is therefore independent of resolution or signal to noise ratio. The original spectrum is only used for reading out the amplitudes corresponding to the given peak positions (using the adjustable tolerance window for splited peaks).

The user can adjust all model parameters by interactively operating sliders with the computer mouse (e.g. for line width or Gaus/Lorentz balance). The adjustments have immediate effect. Any peak can sequentially be selected as reference in order to optimize the model over the whole peak set. The current implementation uses the same parameter settings for all peaks (which is very efficient and appropriate for our present applications). The matrix of Eq. 13 only became singular if more than one input peak had the same position. In practice this can easily be recognized and avoided. Chapter 6.8 shows an application of the algorithm.

5 Typical NMR Experiments

In this chapter the *SpectrumType* model introduced in chapter 4.3 is applied to specify common NMR experiment types used in homonuclear and heteronuclear backbone and sidechain assignment.

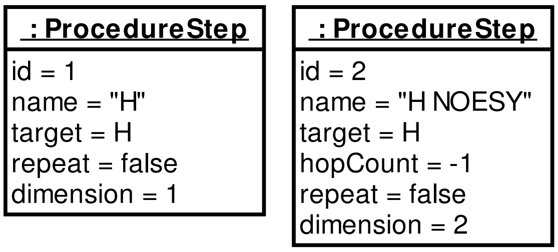
5.1 2D [^{1}H,^{1}H]-COSY



: ProcedureStep
id = 1
name = "H"
target = H
repeat = false
dimension = 1

: ProcedureStep
id = 2
name = "H COSY"
target = H
hopCount = 3
repeat = false
dimension = 2

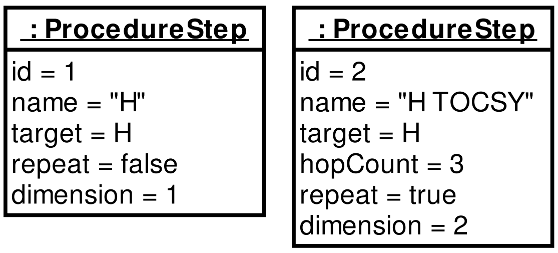
5.2 2D [^{1}H,^{1}H]-NOESY



: ProcedureStep
id = 1
name = "H"
target = H
repeat = false
dimension = 1

: ProcedureStep
id = 2
name = "H NOESY"
target = H
hopCount = -1
repeat = false
dimension = 2

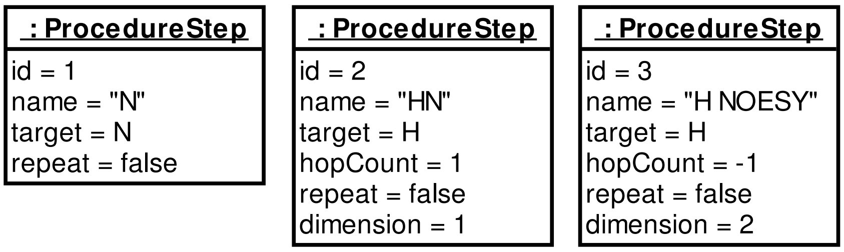
5.3 2D [^{1}H,^{1}H]-TOCSY



: ProcedureStep
id = 1
name = "H"
target = H
repeat = false
dimension = 1

: ProcedureStep
id = 2
name = "H TOCSY"
target = H
hopCount = 3
repeat = true
dimension = 2

5.4 2D [^{1}H,^{15}N]-NOESY

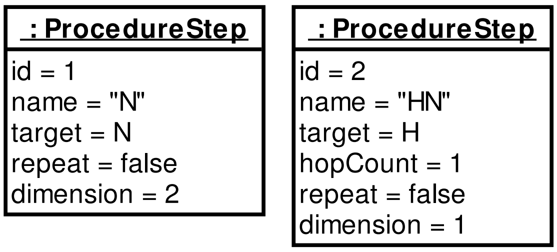


: ProcedureStep
id = 1
name = "N"
target = N
repeat = false

: ProcedureStep
id = 2
name = "HN"
target = H
hopCount = 1
repeat = false
dimension = 1

: ProcedureStep
id = 3
name = "H NOESY"
target = H
hopCount = -1
repeat = false
dimension = 2

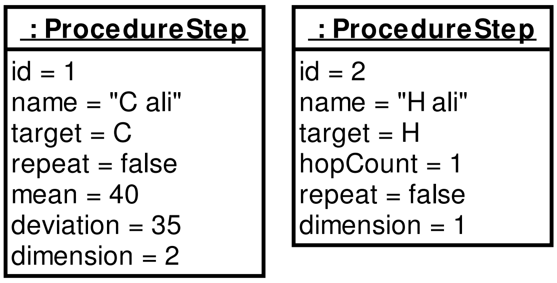
5.5 2D [^{1}H,^{15}N]-HSQC



: ProcedureStep
id = 1
name = "N"
target = N
repeat = false
dimension = 2

: ProcedureStep
id = 2
name = "HN"
target = H
hopCount = 1
repeat = false
dimension = 1

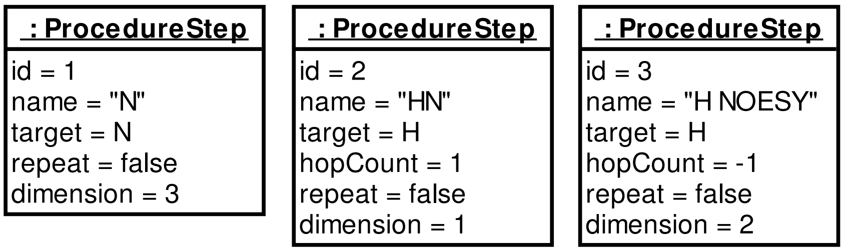
5.6 2D [^{1}H,^{13}C]-HSQC aliphatic



: ProcedureStep
id = 1
name = "C ali"
target = C
repeat = false
mean = 40
deviation = 35
dimension = 2

: ProcedureStep
id = 2
name = "H ali"
target = H
hopCount = 1
repeat = false
dimension = 1

5.7 3D ^{15}N-ed. [^{1}H,^{1}H]-NOESY



: ProcedureStep
id = 1
name = "N"
target = N
repeat = false
dimension = 3

: ProcedureStep
id = 2
name = "HN"
target = H
hopCount = 1
repeat = false
dimension = 1

: ProcedureStep
id = 3
name = "H NOESY"
target = H
hopCount = -1
repeat = false
dimension = 2

5.8 3D ^{15}N-ed. [^{1}H,^{1}H]-TOCSY

: ProcedureStep
id = 1
name = "N"
target = N
repeat = false
dimension = 3

: ProcedureStep
id = 2
name = "HN"
target = H
hopCount = 1
repeat = false
dimension = 1

: ProcedureStep
id = 3
name = "H TOCSY"
target = H
hopCount = 3
repeat = true
dimension = 2

5.9 3D ^{13}C-ed. [^{1}H,^{1}H]-NOESY

: ProcedureStep
id = 1
name = "C"
target = C
repeat = false
dimension = 3

: ProcedureStep
id = 2
name = "HC"
target = H
hopCount = 1
repeat = false
dimension = 1

: ProcedureStep
id = 3
name = "H NOESY"
target = H
hopCount = -1
repeat = false
dimension = 2

5.10 CBCA(CO)NH

: ProcedureStep
id = 1
name = "CA/CB"
target = C
mean = 40
deviation = 35
dimension = 2

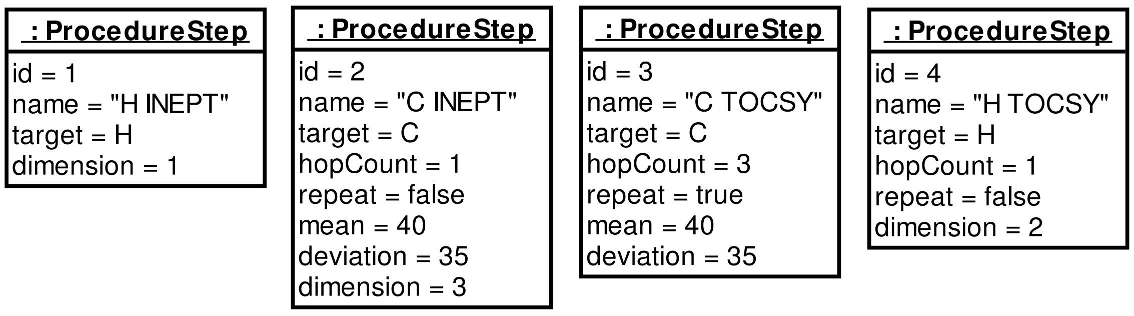
: ProcedureStep
id = 2
name = "CA"
target = C
hopCount = 1
repeat = false
mean = 40
deviation = 35

: ProcedureStep
id = 3
name = "C"
target = C
hopCount = 1
repeat = false
mean = 175
deviation = 20

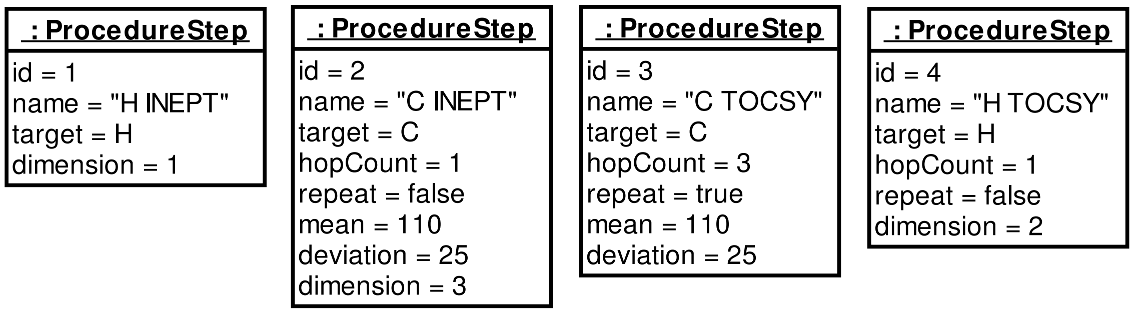
: ProcedureStep
id = 4
name = "N"
target = N
hopCount = 1
repeat = false
dimension = 3

: ProcedureStep
id = 5
name = "HN"
target = H
hopCount = 1
repeat = false
dimension = 1

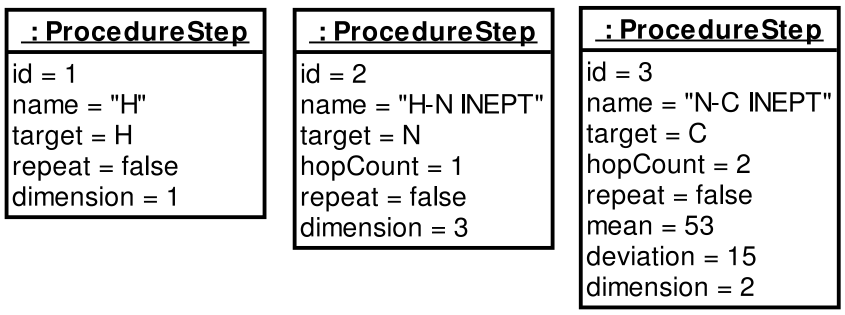
5.11 HC(C)H-TOCSY aliphatic



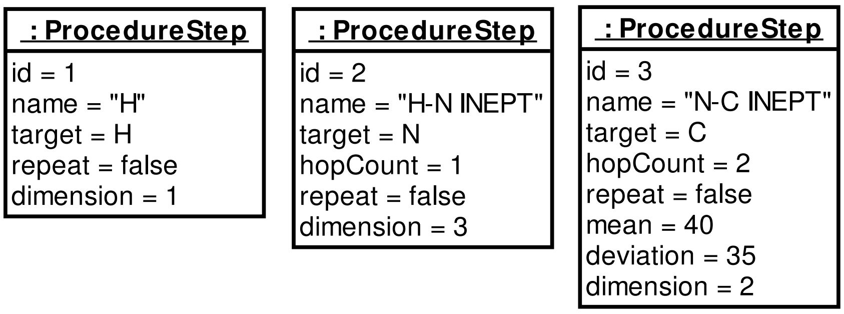
5.12 HC(C)H-TOCSY aromatic



5.13 HNCA

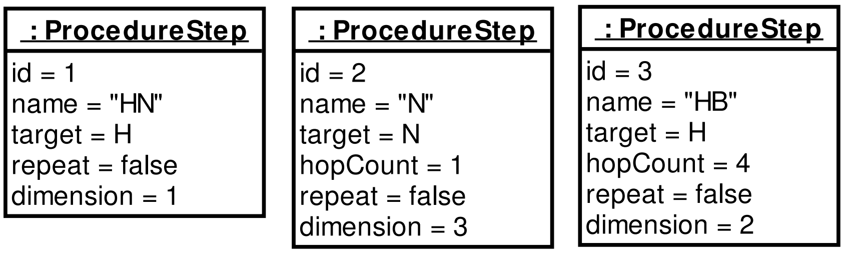


5.14 HNCACB



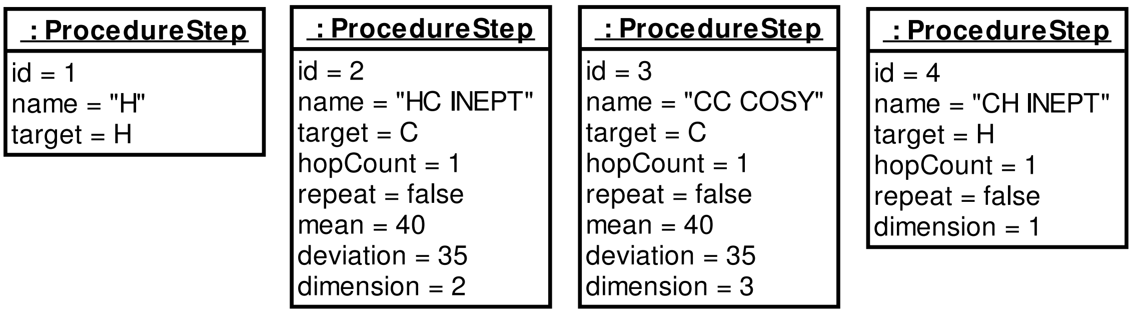
: ProcedureStep	: ProcedureStep	: ProcedureStep
id = 1	id = 2	id = 3
name = "H"	name = "H-N INEPT"	name = "N-C INEPT"
target = H	target = N	target = C
repeat = false	hopCount = 1	hopCount = 2
dimension = 1	repeat = false	repeat = false
	dimension = 3	mean = 40
		deviation = 35
		dimension = 2

5.15 HNHB



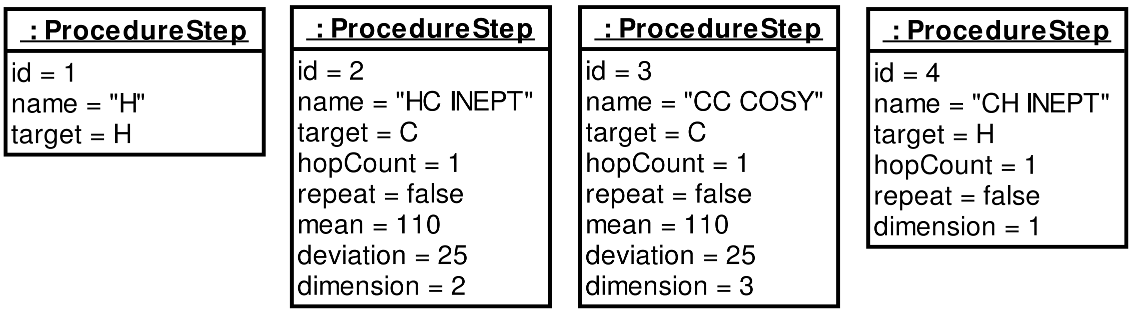
: ProcedureStep	: ProcedureStep	: ProcedureStep
id = 1	id = 2	id = 3
name = "HN"	name = "N"	name = "HB"
target = H	target = N	target = H
repeat = false	hopCount = 1	hopCount = 4
dimension = 1	repeat = false	repeat = false
	dimension = 3	dimension = 2

5.16 (H)CCH-COSY aliphatic



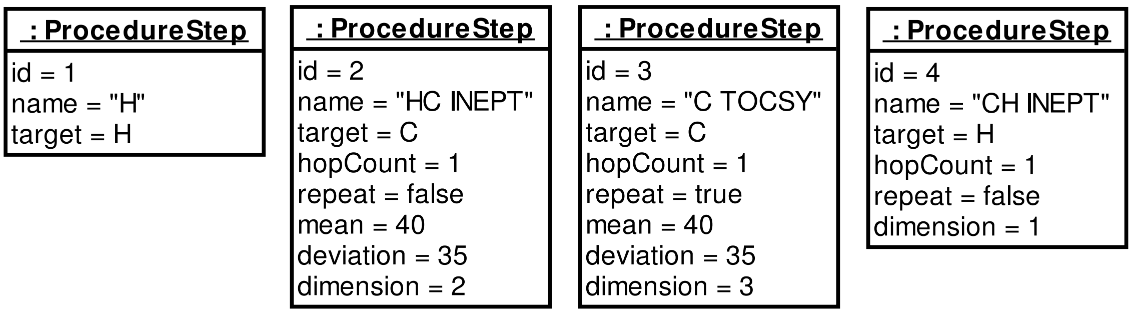
: ProcedureStep	: ProcedureStep	: ProcedureStep	: ProcedureStep
id = 1	id = 2	id = 3	id = 4
name = "H"	name = "HC INEPT"	name = "CC COSY"	name = "CH INEPT"
target = H	target = C	target = C	target = H
	hopCount = 1	hopCount = 1	hopCount = 1
	repeat = false	repeat = false	repeat = false
	mean = 40	mean = 40	dimension = 1
	deviation = 35	deviation = 35	
	dimension = 2	dimension = 3	

5.17 (H)CCH-COSY aromatic



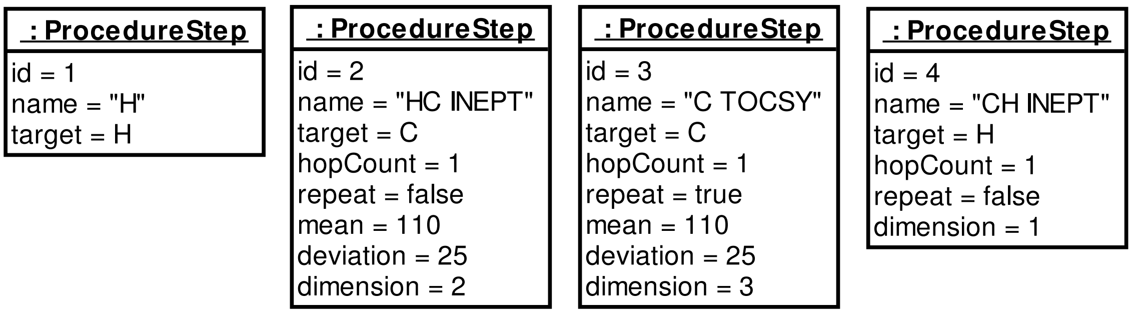
: ProcedureStep	: ProcedureStep	: ProcedureStep	: ProcedureStep
id = 1	id = 2	id = 3	id = 4
name = "H"	name = "HC INEPT"	name = "CC COSY"	name = "CH INEPT"
target = H	target = C	target = C	target = H
	hopCount = 1	hopCount = 1	hopCount = 1
	repeat = false	repeat = false	repeat = false
	mean = 110	mean = 110	dimension = 1
	deviation = 25	deviation = 25	
	dimension = 2	dimension = 3	

5.18 (H)CCH-TOCSY aliphatic



: ProcedureStep	: ProcedureStep	: ProcedureStep	: ProcedureStep
id = 1	id = 2	id = 3	id = 4
name = "H"	name = "HC INEPT"	name = "C TOCSY"	name = "CH INEPT"
target = H	target = C	target = C	target = H
	hopCount = 1	hopCount = 1	hopCount = 1
	repeat = false	repeat = true	repeat = false
	mean = 40	mean = 40	dimension = 1
	deviation = 35	deviation = 35	
	dimension = 2	dimension = 3	

5.19 (H)CCH-TOCSY aromatic



: ProcedureStep	: ProcedureStep	: ProcedureStep	: ProcedureStep
id = 1	id = 2	id = 3	id = 4
name = "H"	name = "HC INEPT"	name = "C TOCSY"	name = "CH INEPT"
target = H	target = C	target = C	target = H
	hopCount = 1	hopCount = 1	hopCount = 1
	repeat = false	repeat = true	repeat = false
	mean = 110	mean = 110	dimension = 1
	deviation = 25	deviation = 25	
	dimension = 2	dimension = 3	

6 Computer Aided Resonance Assignment

Computer Aided Resonance Assignment (short *CARA*) is on one hand the name of an optimized process model for the analysis and assignment of NMR spectra, and on the other hand the name of an interactive, graphical computer program supporting that process. This chapter gives an introduction into both the process and the computer program. It has the form of a tutorial, subdivided along to the main use cases of the process. The descriptions apply to CARA 1.1. We start with an overview explaining the general concepts.

6.1 Overview

The computer program CARA is a complete implementation of the conceptual model formally introduced in chapter 4. It should be regarded as a proof-of-concept of the model. CARA is a modern, efficient computer program featuring a state-of-the-art graphical user interface (GUI) with many usability features (like undo/redo) and a look and feel commonly accepted by the community. It is completely platform independent, offering identical features on all platforms. The program is currently deployed on Microsoft Windows (9x, ME, 2k, XP), Linux, Macintosh OS X, Sun Solaris and Silicon Graphics IRIX. Since CARA simply consists of a single executable file, installation is very easy (no libraries and the like are needed). Everything is written in ANSI C++ [Stroustrup 1997] (more than hundred thousand lines of code), adhering to modern software engineering standards. The program is therefore very fast, maintainable, extensible and adaptable to future developments. It can be downloaded for free from http://www.nmr.ch.

CARA is organized around a central *Explorer* window (see Figure 26). It is present during the whole session and enables access to all other tool windows of CARA. In contrast to previous solutions CARA doesn't follow the one-size-fits-all approach, but offers specialized environments for all major use cases. The user can even construct new environments if she wants to use the built in scripting language.

Figure 26 shows the nine standard tool windows (i.e. environments). All of them (besides NEASY, see below) follow the same presentation and usability concepts (i.e. their look and feel is very similar). Their difference lays in the specialization of their functionality.

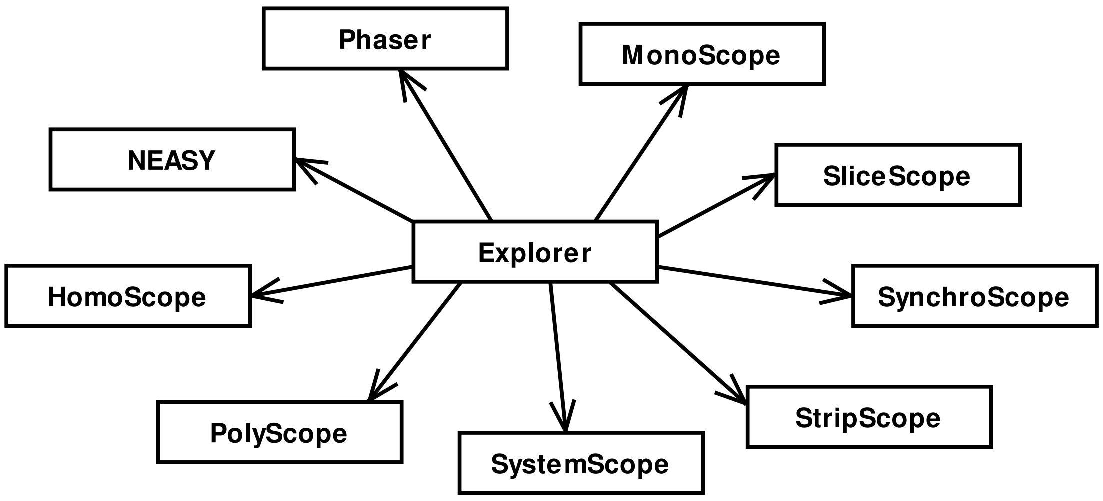


Figure 26: The CARA editors accessible with the Explorer

The following list gives a brief overview of the purpose of each environment.

1. *SynchroScope* is optimized to analyze HSQC/HNCA pairs (i.e. all spectra which follow the one-strip-per-residue-approach). It combines the plane view of the HSQC with a strip view of the HNCA.

2. *StripScope* is used to do 3D triple-resonance backbone assignment, i.e. to show selected strips, combine them to fragments and to map them interactively on the sequence.

3. *SystemScope* is the tool for managing and assigning the spins of a single spin system. It is used for side-chain assignment and optimized for the navigation in TOCSY type spectra. The tool makes use of pathway simulation.

4. *HomoScope* is optimized for homo-nuclear assignment of 2D spectra, but can also be used e.g. to check the completeness of a side-chain assignment using a 2D [^{1}H, ^{13}C]-HSQC (Vuister et al. 1992). It can handle both SpinLinks and pathway simulation.

5. *PolyScope* combines the features of HomoScope with the navigation concepts of SynchroScope. It can be used for 3D NOESY constraint gathering or special assignment strategies.

6. *MonoScope* is a flexible tool to explore two or higher-dimensional spectra. It can be used to create and manage peaklists and to perform peak volume determination of one or a series of spectra.
7. *SliceScope* is a simple viewer for 1D spectra.
8. *Phaser* can be used to adjust the phase of real and imaginary spectra.
9. *NEASY* is an emulation of a functional subset of XEASY [Bartels et al. 1995]. Its main purpose is to enable backward compatibility and to ease transition to CARA.

The central unit of work managed by CARA is a so called *Repository*. It contains all information needed for and coming up during an assignment project (besides the spectrum files, which are only referenced). Figure 27 gives an overview of the primary objects a *Repository* contains. Most of them were introduced and described in chapter 4.

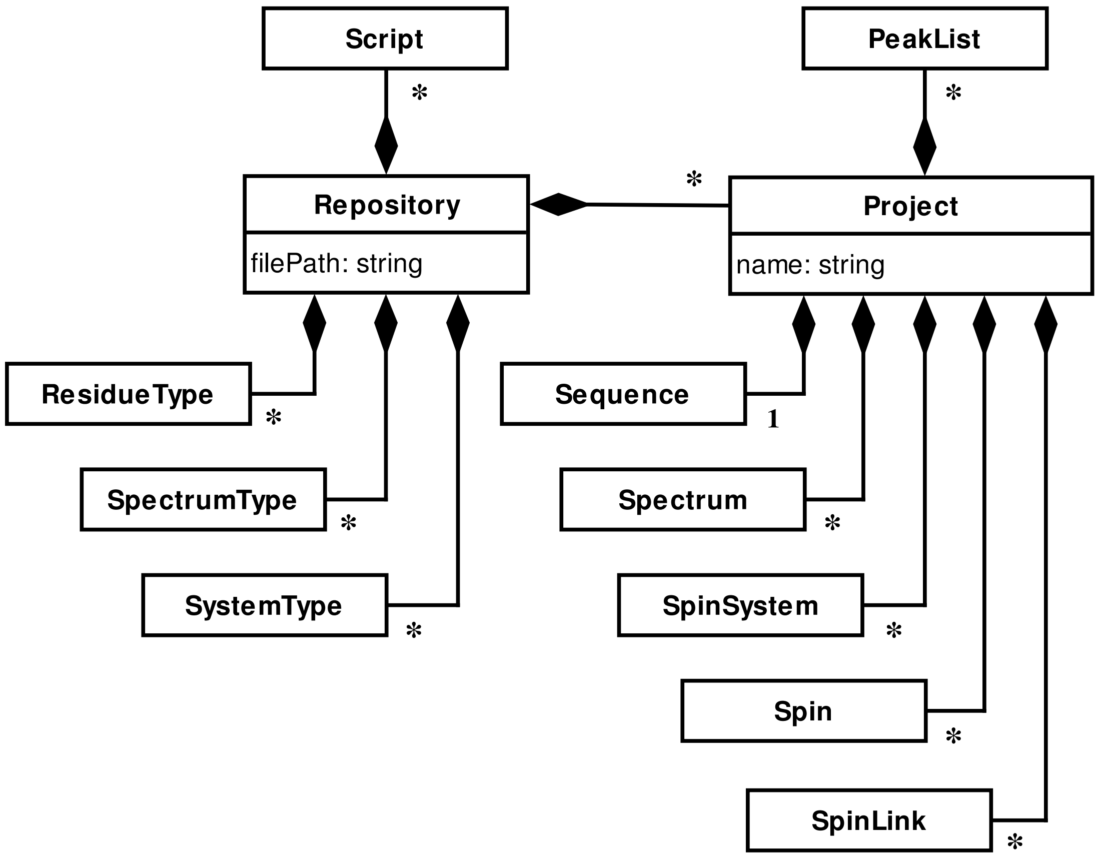
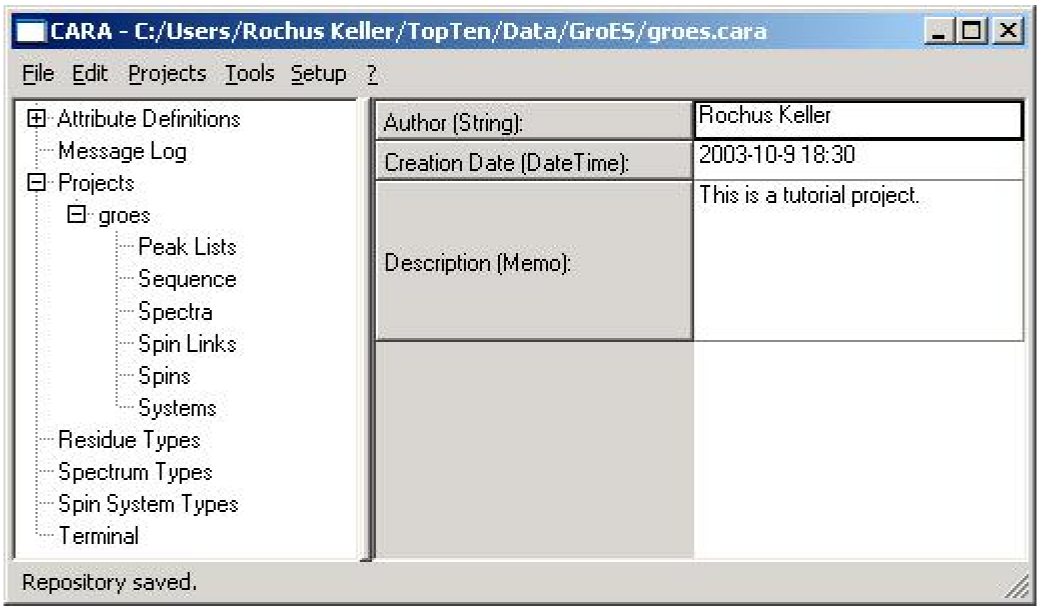


Figure 27: Structure of a Repository

It is important to notice that a *Repository* brings together all library and project data into a single file (omitting the versioning problems of previous solutions with separate library files). A *Repository* can contain more than one *Project* which is convenient when the work is subdivided into related sub-projects.



Author (String):	Rochus Keller
Creation Date (DateTime):	2003-10-9 18:30
Description (Memo):	This is a tutorial project.

Repository saved.

Figure 28: CARA Explorer

The main purpose of the Explorer (Figure 28) is to manage a *Repository*. The left side of the split bar shows a category tree and the right side is the pane where the content of the selected category is displayed (or the default pane as shown if a group category is selected). The following list gives a short overview of the categories.

1. Attribute Definitions: contains sub-categories (not shown) for each object type which can have dynamic attributes (see chapter 6.11).
2. Message Log: displays a list of system messages which should not be ignored.
3. Projects: this group contains all *Projects* contained in the *Repository*. In Figure 28 there is one *Project* called "groes".
4. Peak Lists: displays a list of all *PeakLists* of the project (if there are any).
5. Sequence: shows the *Sequence* associated with the project.
6. Spectra: displays the list of spectra associated with the project. The list is used to manage spectra and to open them using the tool windows.
7. Spins: this is the main list of all *Spins* and *SpinSystems* of the *Project*. *Spins* can either be managed from within this list or from within a tool window.

8. Spin Links: this pane contains the main list of all *SpinLinks* of the *Project*. It can be used to directly manage them (i.e. to create or delete them or change attributes).
9. Systems: this is the main list of all *SpinSystems* with their subordinate *Spins*. It can be used to directly manage *SpinSystems*, *Spins*, *SpinLinks* and all kind of assignments.
10. Residue Type, Spectrum Type, Spin System Type: list and manage *ResidueTypes*, *SpectrumTypes* and *SystemTypes*. The changes are immediately visible to all *Projects* of the *Repository*.
11. Terminal: this pane gives access to the scripting environment of CARA. It contains a command line and is used to manage *Scripts*.

Note that virtually all CARA windows and panes (i.e. different parts of the windows) support context menus, which pop up when the user presses the right mouse button (or the left mouse button while pressing the command key on Macintosh). Most functions of CARA are accessible that way. There are also some essential navigation shortcuts, which are not self-evident but important enough to be considered from the beginning (for a complete list see the annex on page 140).

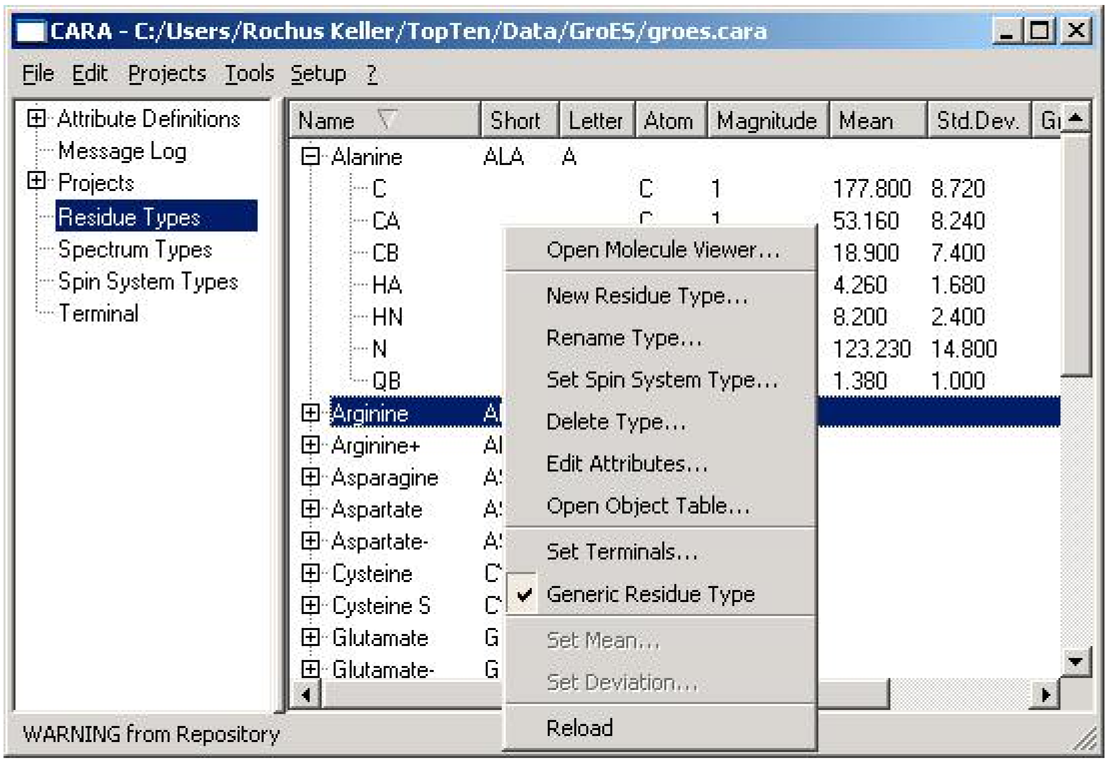
6.2 Setting up a Repository

A Repository is a file, residing somewhere in the file system. After starting CARA, the Explorer has automatically created a new, empty Repository. You can establish this state anytime yourself by executing the menu command *File/New*. A Repository can be saved to a file using the menu commands *File/Save As* or *File/Save*. The former is used to save the Repository to another file (which is the default behavior for new Repositories) and the latter updates the given file with the most recent changes. With *File/Open* you load a Repository from a file.

You can setup a Repository completely from scratch, but this seems only to be necessary if you want to analyze new molecule or spectrum types, for which no templates are available yet. For most users it will be much more convenient to start a new Repository from a template (menu *File/New from Template*). A template is nothing more than an ordinary Repository created by yourself or someone else. If you use it as a template, only its library part is copied into your new Repository (i.e. no project data). If you are using CARA for NMR based protein structure determination, your new Repository would then contain all amino acids, spin system types, scripts and common NMR experiments. So you could immediately start setting up your project (as described in chapter 6.3). For all other users we briefly explain how to survive without a template.

6.2.1 Managing ResidueTypes

At latest when you try to load a sequence into a project whose residue types are not known to CARA, you will get a friendly error message. This signifies that your sequence file either doesn't use the same residue type nomenclature like your Repository or the residue type is not yet defined. To create a new residue type, click on the *Residue Types* category in the Explorer. The pane shown in Figure 29 becomes visible (it might be empty or look different depending on how you started).



CARA - C:/Users/Rochus Keller/TopTen/Data/GroES/groes.cara

Name	Short	Letter	Atom	Magnitude	Mean	Std.Dev.	Gi
Alanine	ALA	A					
C			C	1	177.800	8.720	
CA			C	1	53.160	8.240	
CB					18.900	7.400	
HA					4.260	1.680	
HN					8.200	2.400	
N					123.230	14.800	
QB					1.380	1.000	
Arginine	A						
Arginine+	AI						
Asparagine	A:						
Aspartate	A:						
Aspartate-	A:						
Cysteine	C						
Cysteine S	C						
Glutamate	G						
Glutamate-	G						

WARNING from Repository

Figure 29: Explorer Residue Types pane

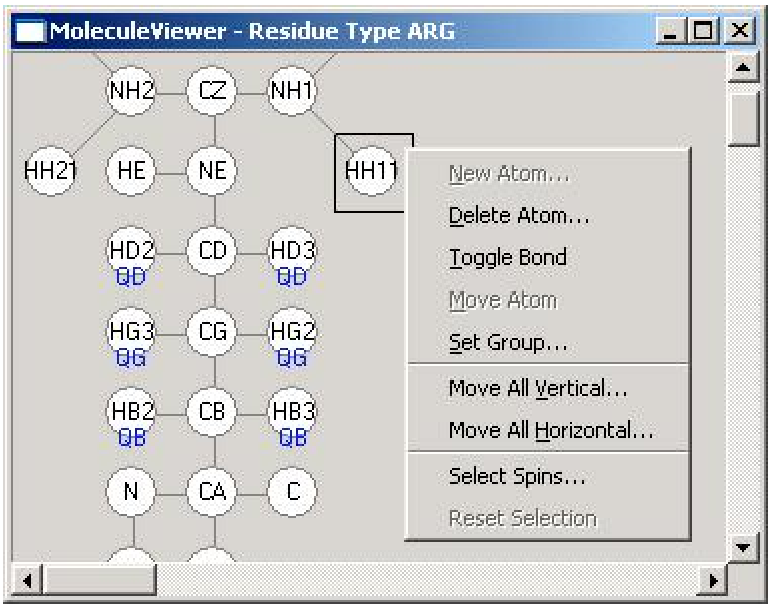
The figure also shows the context menu, which appears when the right mouse button is pressed (or the left button together with the command key on Macintosh OSX). You create a new residue type by executing the corresponding menu entry. A dialog box appears where you have to enter three alternative name versions (long, short and letter). A residue type is always referenced by its unique short name from everywhere within the repository. The long and letter versions are just for convenience. Consider that only the long name can be changed later. After creation the new type appears at the end of the list.

A residue type can only be deleted if it is not referenced (e.g. by a sequence). Optionally a spin system type can be assigned to the residue type to bias the sequential assignment of a spin system (see chapter 4.2.2 and 6.6).

The menu item *Set Terminals* controls by which atoms two residues are linked together (e.g. with *N* to the left and with *C* to the right in case of amino acids). The currently selected residue type can be declared to be used as the generic type for pathway simulation (in case a spin system has no assignment yet, see chapter 4.3.1). Both options affect the whole repository (i.e. all contained projects).

The options *Edit Attributes* and *Open Object Table* will be explained in chapter 6.11.

The molecule information of a new or existing type can be changed within the *Molecule Viewer* (Figure 30). The figure again shows the context menu.



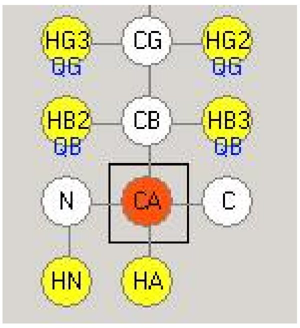


Figure 30: Molecule Viewer with Residue Type Arginine (left) and the effect of *Select Spins* on a CA with hopCount 3 on H (right)

New atoms are created by first positioning the black position rectangle using the mouse and then executing the *New Atom* command from the context menu. A unique tag and other model attributes then have to be entered into a dialog box. The attributes *group*, *mean* and *deviation* are optional and can be changed later. The group to which an atom belongs to, is drawn on the lower part of the circles. The bond between the atoms can be toggled on or off by the corresponding menu item (or by pressing the control key and then clicking on the target atom). The new atoms become visible in the explorer pane after executing *Reload*.

With the menu item *Select Spins* one can check the effect of a certain hop count to the given molecule (right part of Figure 30, see also chapter 4.3 for explanation).

6.2.2 Managing Spectrum Types

Before you can load an NMR spectrum into a project the corresponding spectrum type has to be defined. To create or change a spectrum type, click on the corresponding category in the Explorer. The pane shown in Figure 31 appears.

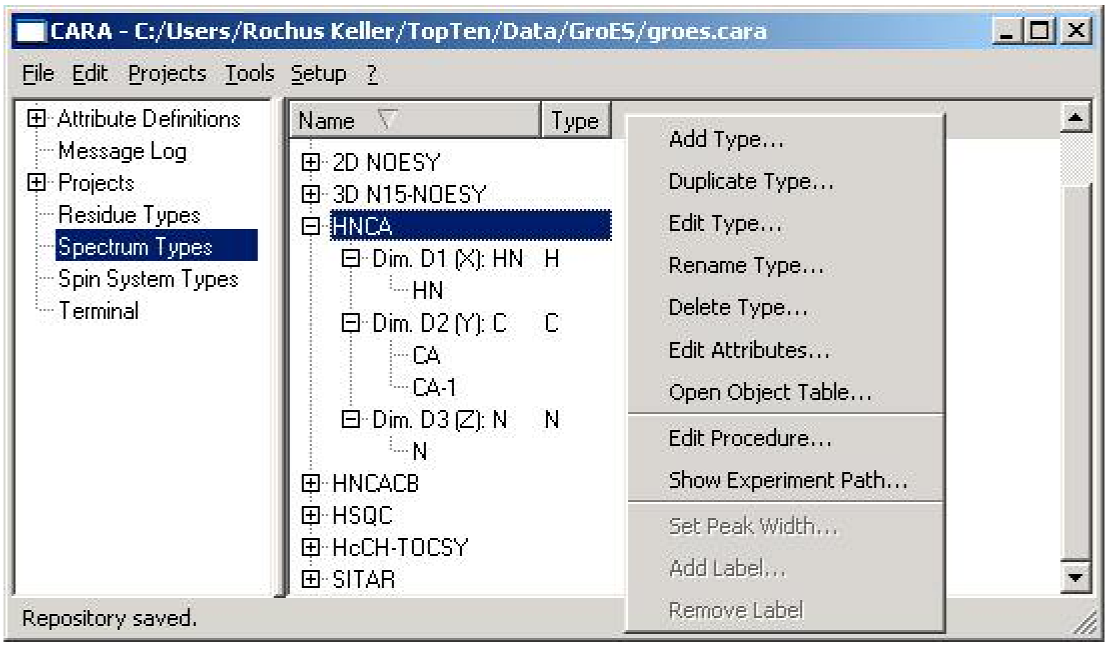
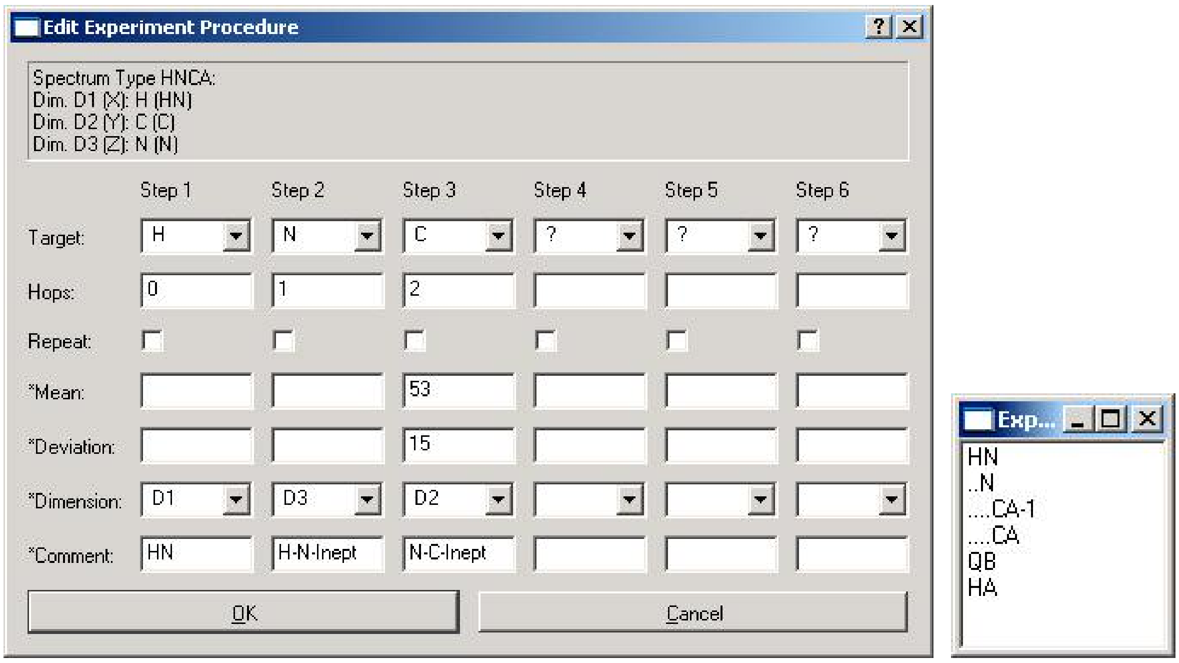


Figure 31: Explorer Spectrum Types Pane

With the context menu items *Add Type* or *Duplicate Type* a new spectrum type can be created. A unique name as well as the atom types of all needed dimensions have to be entered. Spectrum types are usually defined with the dimension order they are most likely to be presented on screen (e.g. HN=horizontal and C=vertical for a HNCA). Once created only the names can be changed later (but not the atom types or number of dimensions). Furthermore a type cannot be deleted as soon it is referenced by a spectrum.

For each dimension the spin labels one expects to see can be optionally declared. The selected HNCA in Figure 31 contains a unique *HN* and *N* label in dimension *X* and *Z* and the two labels *CA* and *CA-1* in dimension *Y*. Because *HN* and *N* are unique, CARA can automatically rotate and display HNCA spectra using SynchroScope and StripScope (see chapter 6.4).

For each spectrum type an experiment procedure can optionally be defined. Figure 32 shows the dialog for procedure editing. The periods of the NMR experiment are declared in order of their execution (see chapter 4.3). The *Dimension* attribute is then used to map them on the dimensions of the spectrum type (idicated in the upper pane of the dialog for convenience). Each dimension may be referenced exactly once. Only steps with atom types unequal to "?" are considered valid. All spectrum types examined by the author could be modeled by six steps only. CARA doesn't impose a restriction on the number of steps and the dialog could be easily extended if necessary.


Edit Experiment Procedure

Spectrum Type HNCA:
Dim. D1 (X): H (HN)
Dim. D2 (Y): C (C)
Dim. D3 (Z): N (N)

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Target:	H	N	C	?	?	?
Hops:	0	1	2			
Repeat:	☐	☐	☐	☐	☐	☐
*Mean:			53			
*Deviation:			15			
*Dimension:	D1	D3	D2			
*Comment:	HN	H-N-Inept	N-C-Inept			

OK Cancel

Exp...

HN
..N
....CA-1
....CA
QB
HA

Figure 32: Experiment Procedure of HNCA (left) and simulated magnetization transfer pathway of Alanine (right)

With the command *Show Experiment Path* the user has the possibility to validate whether the defined procedure renders the correct pathways. After selecting the residue type from a popup dialog, the simulated magnetization paths are presented as a tree (Figure 32, right). The algorithm has found the two paths *HN-N-CA* and *HN-N-CA-1*. The paths starting with *QB* and *HA* were cut off after the first step. If we had left out the declaration of mean and deviation for step 3, three new paths to *C*, *C-1* and *CB* would be detected (try it). The concepts of pathway simulation is explained in chapter 4.3.1.

6.3 Setting up a Project

If the repository has been successfully set up (by using a template or creating everything by yourself), one can start creating projects. To accomplish this the user activates the menu *Projects/New Project* from within the Explorer. First a unique project name has to be entered. CARA then asks the user, whether a sequence should be loaded. If the user decides not to load a sequence (e.g. if only amide exchange rates should be determined), the creation is finished. Otherwise the user can select a sequence file (in EASY [Eccles et al. 1991] format) from the file system.

Several variations of sequence files exist. The following table shows three variations of the first six residues of GroES. Each line corresponds to a residue.

Table 3: Sequence file variations, plain (left), with residue numbers (mid) and with assignments (right)

MET	MET	1	MET	1	420
ASN	ASN	2	ASN	2	382
ILE	ILE	3	ILE	3	399
ARG	ARG	4	ARG	4	306
PRO	PRO	5	PRO	5	
LEU	LEU	6	LEU	6	397

The plain variant is loaded without further notice (as long as each residue type is available). If CARA reads the version in the middle, it asks the user, whether the numbers represent residues or spin systems. In the latter case empty spin systems with the given number are created and initially connected to fragments and assigned to the sequence. The same thing happens if CARA reads the right variation and the user agrees to create spin systems.

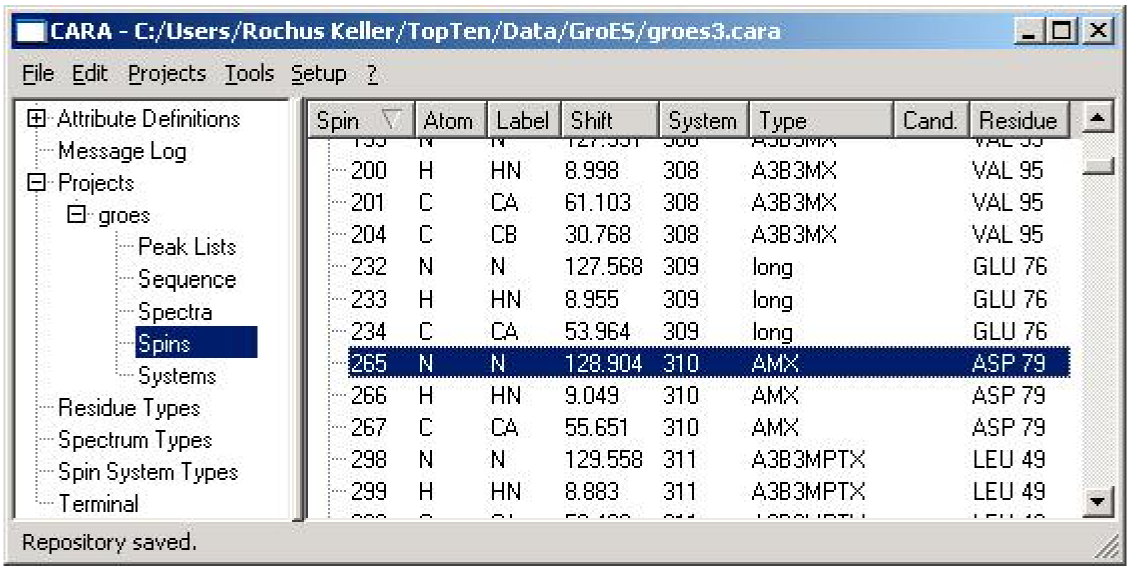
The user might optionally import an spinlist (using EASY proton list format) if one is available (i.e. corresponding to the sequence loaded). The user does so by selecting the project name in the category tree of the Explorer and activating *Project/Import/Atom List* from the menu. The user is then asked whether the assignment numbers reference residues or spin systems. This gives four different possibilities:

1. If the spin systems were created together with the sequence, and the spinlist references residues, then the atoms are imported into the spin systems, which assigned to the corresponding residues.
2. If no spin systems were created when importing the sequence, and the spinlist references residues, then new spin systems are created and assigned to the residues, which were referenced by the atoms (i.e. a system is only created if at least one atom references the residue). The atoms are imported into the new spin systems.
3. If spin systems were created together with the sequence, and the spinlist references spin systems, then the systems are only created if not yet existing, but not assigned to any residue. The atoms are imported into the referenced spin systems, whether they already existed or were created.
4. If no spin systems were created when importing the sequence, and the spinlist references spin systems, then all referenced systems are created, but not assigned to any residue. The atoms are imported into the new spin systems.

Table 4: Part of the imported atomlist (i.e. spinlist)

```
265 128.904 0.000 N      79
266   9.049 0.000 HN     79
267  55.651 0.013 CA     79
268 999.000 0.000 HA     79
270 999.000 0.000 CB     79
```

Only valid lines are imported from atomlists (i.e. where the PPM value is unequal to 999.000). The labels of the imported lines must be valid and also unique within the target spin systems. If the atomlist contains double spin numbers or the number is already occupied by a spin of the project, the user has the chance to ignore the doubles or abort the import. The import of further atomlists can happen at any time in the future, but CARA will display a warning message if spins already exist in the project. The imported spins and spin systems are listed in the *Spins* and *Systems* pane of the category tree in Explorer (see Figure 33). The panes are empty of course, if no atomlist was imported and no spins were created otherwise.



Spin	Atom	Label	Shift	System	Type	Cand.	Residue
200	H	HN	8.998	308	A3B3MX		VAL 95
201	C	CA	61.103	308	A3B3MX		VAL 95
204	C	CB	30.768	308	A3B3MX		VAL 95
232	N	N	127.568	309	long		GLU 76
233	H	HN	8.955	309	long		GLU 76
234	C	CA	53.964	309	long		GLU 76
265	N	N	128.904	310	AMX		ASP 79
266	H	HN	9.049	310	AMX		ASP 79
267	C	CA	55.651	310	AMX		ASP 79
298	N	N	129.558	311	A3B3MPTX		LEU 49
299	H	HN	8.883	311	A3B3MPTX		LEU 49

Repository saved.

Figure 33: A new project after importing an atomlist

After the sequence and perhaps the atomlist was imported, it is time to add NMR spectra to the new project. If the user clicks on the *Spectra* category in the Explorer, a pane listing all spectra of the project is displayed (initially empty). Within the spectrum pane, a context menu is available. The menu item *Add Spectrum* contains all spectrum types as sub items. If for example the user wants to load a HNCA spectrum, she has to activate the menu item *Add Spectrum/HNCA* (provided a spectrum type called *HNCA* exists). The spectrum file has then to be looked up using the file selector dialog.

CARA directly supports different spectrum file types (e.g. EASY, Bruker, etc.). Two file types ("CARA Spectrum *.nmr" and "EASY Spectrum *.param") are explicitly listed. The other formats can be accessed using the * or *.* file patterns. The supported file types can be directly used, i.e. there is no need to convert them (as is often the case with other software packages).

The dialog shown in Figure 34 can be used to load more than one file at the same time (multi selection is possible by pressing the shift or control keys while clicking the mouse on the file name).

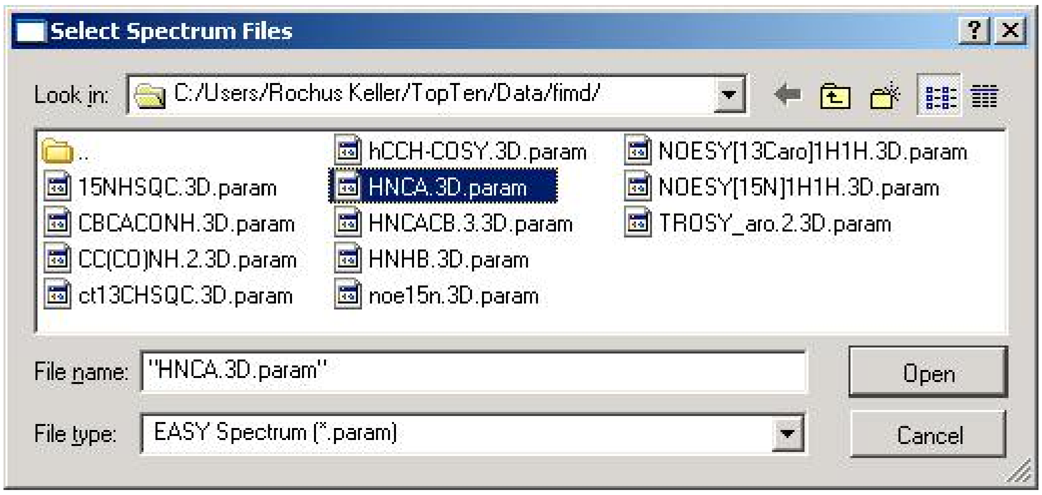


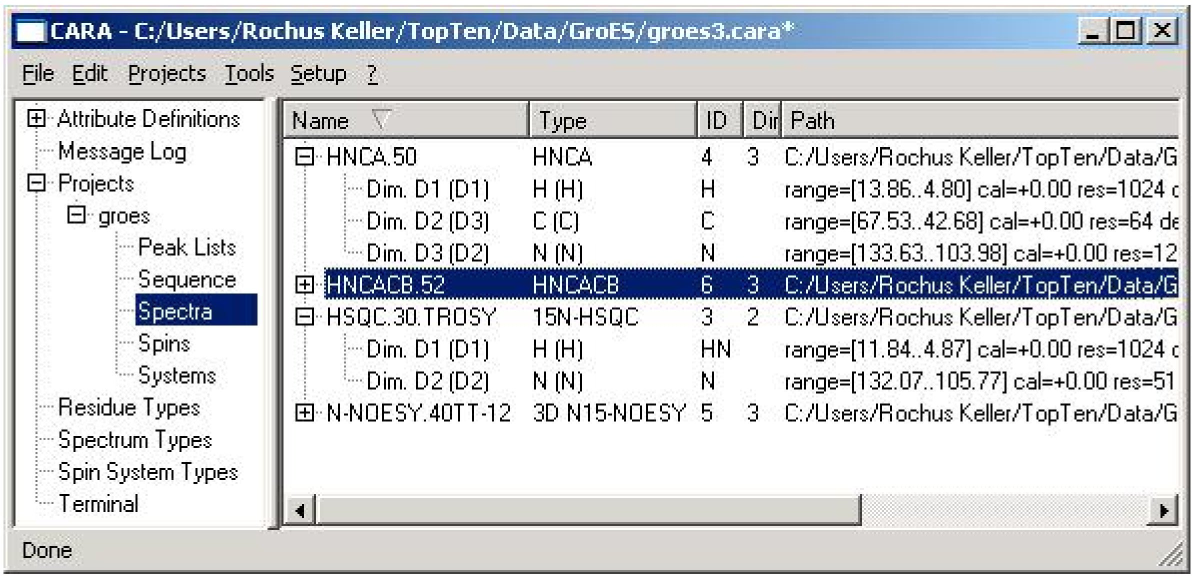
Figure 34: Spectrum file selector dialog

CARA tries to automatically rotate the dimensions of the loaded spectrum, so that they fit the given spectrum type. Depending on the spectrum file format, there are different hints about the atom types related with the dimensions. For example in a EASY param file (Table 5) the lines *Identifier for dimension* are usually used to assign an arbitrary name to the dimension. CARA uses the first letter of this name as the atom type symbol if available (i.e. *N* and *H* in the example).

Table 5: EASY param file of a HSQC spectrum

```
Version ....................... 1
Number of dimensions .......... 2
16 or 8 bit file type ......... 16
Spectrometer frequency in w1 .. 81.086098
Spectrometer frequency in w2 .. 800.133972
Spectral sweep width in w1 .... 26.301001
Spectral sweep width in w2 .... 6.974300
Maximum chemical shift in w1 .. 132.072830
Maximum chemical shift in w2 .. 11.842400
Size of spectrum in w1 ........ 512
Size of spectrum in w2 ........ 1024
Submatrix size in w1 .......... 64
Submatrix size in w2 .......... 128
Permutation for w1 ............ 2
Permutation for w2 ............ 1
Folding in w1 ................. RSH
Folding in w2 ................. RSH
Type of spectrum .............. ?
Identifier for dimension w1 ... N
Identifier for dimension w2 ... HN
```

There are spectrum file formats which don't provide enough information about atom types. CARA then tries to guess the atom type by comparing the mean of the given PPM range to a table of typical PPM ranges (e.g. -2..14 is interpreted as H, 10..100 as C and 100..200 as N). This can lead to confusion if for example only the PPM area of aromatic spins is recorded. In these cases, CARA will do the wrong guess and the user has to correct the mapping using the context menu function *Map to Type* (if CARA detects the uncertainty it automatically executes this function after loading). Once the mapping is adjusted, one doesn't have to bother anymore, since it is stored with the repository. After loading, the name of the spectrum can be changed using menu *Rename Spectrum*. A default name was automatically deduced from the file name by CARA. The changed name should be unique and informative, since it is used in the spectrum selection menus all-over the program.



Name	Type	ID	Dir	Path
HNCA.50	HNCA	4	3	C:/Users/Rochus Keller/TopTen/Data/G
Dim. D1 (D1)	H (H)	H		range=[13.86..4.80] cal=+0.00 res=1024 c
Dim. D2 (D3)	C (C)	C		range=[67.53..42.68] cal=+0.00 res=64 de
Dim. D3 (D2)	N (N)	N		range=[133.63..103.98] cal=+0.00 res=12
HNCACB.52	HNCACB	6	3	C:/Users/Rochus Keller/TopTen/Data/G
HSQC.30.TROSY	15N-HSQC	3	2	C:/Users/Rochus Keller/TopTen/Data/G
Dim. D1 (D1)	H (H)	HN		range=[11.84..4.87] cal=+0.00 res=1024 c
Dim. D2 (D2)	N (N)	N		range=[132.07..105.77] cal=+0.00 res=51
N-NOESY.40TT-12	3D N15-NOESY	5	3	C:/Users/Rochus Keller/TopTen/Data/G

Figure 35: Spectrum pane of the project after loading spectra

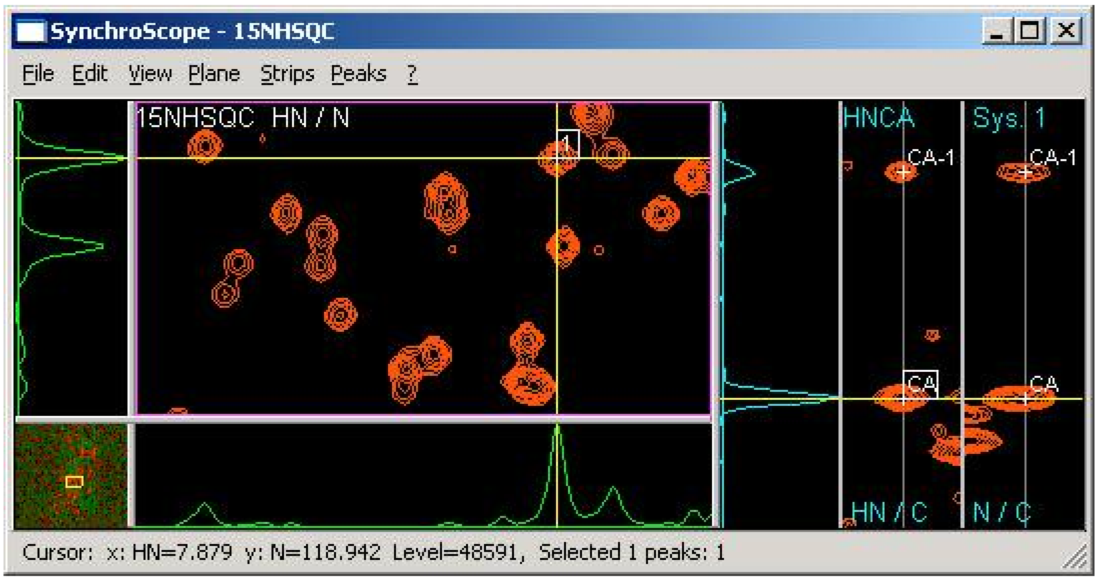
A spectrum can be removed any time, or the spectrum file can be replaced to redirect the the reference to another file (context menu items *Remove Spectrum* and *Replace Spectrum*). The latter can also be used if the spectrum file was renamed or moved to another directory (but don't do this while they're presented in a CARA window).

The project is now ready to be used. Most CARA tool windows (scopes) can be opened by first selecting a spectrum from the list (as in Figure 35) and then executing the corresponding context menu function (e.g. *Open MonoScope*, etc.). We will make extensive use of this menu items in the next sections.

6.4 Heteronuclear Backbone Resonance Assignment

Chapter 2.3 introduces general strategies for sequence specific assignment. In this chapter we will show how backbone resonance assignment is accomplished using double and triple-resonance spectra like [^{1}H,^{15}N]-HSQC, HNCA, HNCACB and ^{15}N-ed. [^{1}H,^{1}H]-NOESY. We assume that a project was set up as described in the previous chapter (i.e. the sequence and spectra were loaded, as shown in Figure 35). The spectra used in this and the next chapter come from the structure determination project of the protein FimD [Bettendorff, Pascal: unpublished data], which was one of the first projects accomplished using CARA.

The main CARA tools used for this process are SynchroScope, StripScope and optionally SystemScope. We start by selecting a [^{1}H,^{15}N]-HSQC spectrum in the spectrum pane of the CARA Explorer and executing the function *Open SynchroScope* from the context menu. If CARA complains you should check that the dimension spin labels of the spectrum type are properly declared (analogous to Figure 31).



Cursor: x: HN=7.879 y: N=118.942 Level=48591, Selected 1 peaks: 1

Figure 36: Identifying spin systems with SynchroScope

SynchroScope allows the simultaneous analysis of a 2D [^{1}H,^{15}N]-HSQC and any 3D spectrum with a unique spin label pair corresponding to the one of the 2D spectrum (HN and N in our case). CARA automatically finds and properly rotates all 3D spectra of the project, which fulfill this condition. They can be selected from the menu *Strips/Select Spectrum*. In Figure 36 we have selected a HNCA, from which a slice and two orthogonal strips are presented in the right part of the window. If you click in the HSQC plane, the cursor and with it the visible HNCA detail are changed. You can select the detail area of the plane by drawing a rectangle in the overview pane in the lower left of the window (by clicking, dragging and releasing the mouse). There are a plenty of shortcuts for zooming and scrolling available (please refer to the annex on page 140). Most panes also have their own context menu (which is accessible by right-clicking on the pane, or command-clicking on Macintosh).

It's now time to pick the first spin system. For this purpose, the cursor shall be positioned on an intensity peak in the HSQC plane, and then the command *Pick System* shall be selected from the context menu or the *Plane* menu at the top of the window. Like in XEASY, there is support for keyboard commands in CARA (e.g. press "PY" to pick a spin system). A new spin system with a unique identification number automatically provided by CARA appears.

If a new spin system was found, it is in most cases immediately possible to identify the CA and CA-1 within the HNCA strips (as a characteristic peak pair, the weaker one corresponding to CA-1). The identification is documented by picking and labeling the two spins (using the corresponding functions in the context menus of the strips). Also note that it is probably necessary to adjust the width of the strip display (menu *Set Strip Width*, e.g. 0.2 PPM for HN and 1.5 PPM for N).

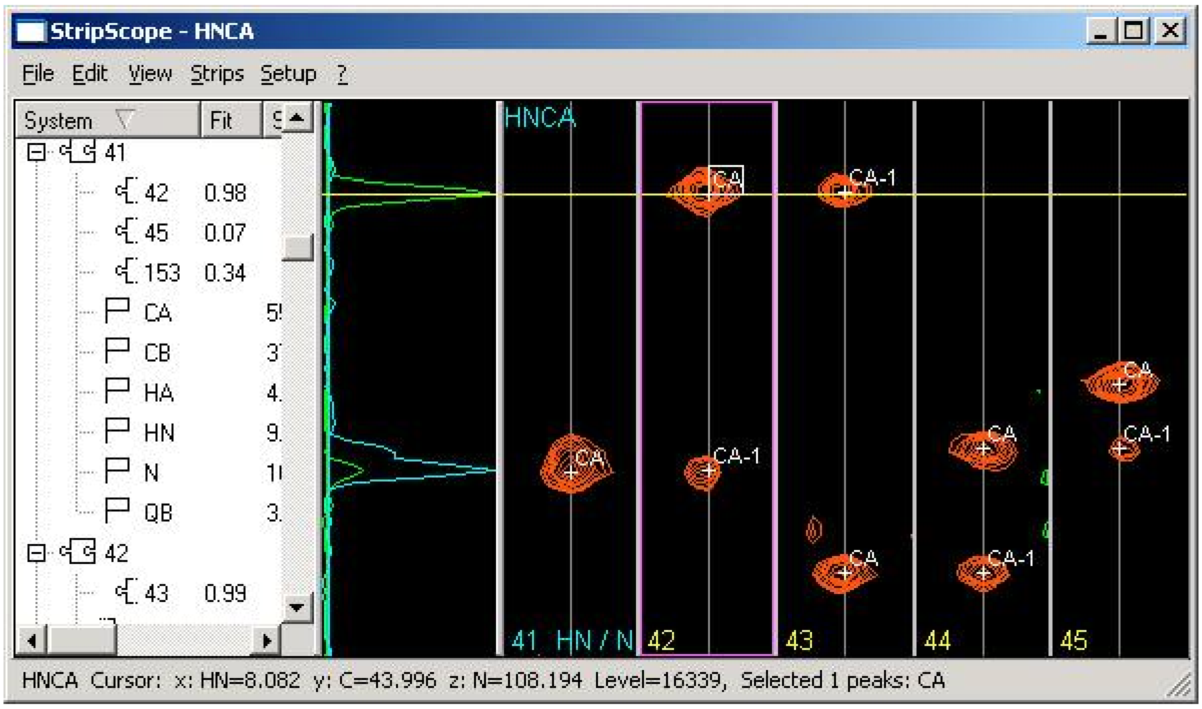
An important point was neglected up to now, the calibration of spectra. It can happen, that the HN/N plane of an HNCA has an offset compared to HSQC. You can correct this using the following steps:

1. Select a single spin system peak within the HSQC plane and position the cursor on a peak within the strips.
2. Uncheck the menu *View/Hide 3D Slices* and check the menu *Plane/Show 3D Plane*. The selected HNCA plane is now displayed.
3. Position the cursor on the peak maximum in the plane, shift-click the corresponding spin system peak (so it is selected) and execute *Plane/Calibrate From System*.

The spectrum (i.e. its PPM scale) has moved so the selected peak should now be on the intensity maximum. A similar procedure can be applied, if the strips of two 3D spectra have to be calibrated (e.g. to synchronize the *C* dimension of a HNCACB and a HNCA). To do this position the cursor to the intensity maximum within one of the strips, shift-click on the corresponding spin peak and execute *Strips/Calibrate Strip*. It can happen, that not all but only single peaks have non-systematic offsets between different spectra. In that case the user should not calibrate the whole spectrum, but adapt a single peak to a given spectrum using *Plane/Move System Alias* or *Strips/Move Spin Alias*.

Ideally it is possible to pick the *HN*, *N*, *CA* and *CA-1* (and mostly also the *CB* and *CB-1*) for all spin systems (besides Prolines) from within the SynchroScope. If this is accomplished, we can start to combine spin systems into fragments using StripScope. The window is opened by selecting the spectrum in the spectrum pane of the CARA explorer and executing the function *Open StripScope* from the context menu. If CARA complains you should check that the dimension spin labels of the spectrum type are properly declared (analogous to Figure 31).

StripScope shows five (up to ten) strip panes on the right, a slice pane in the middle and a spin system tree in the left. Again there are context menus associated with the panes. All spin systems containing spins whose labels correspond to the ones in the unique spin label set of the spectrum type can be displayed as strips. Using the menu *View/Select Strips* the user can control, which subset is shown (e.g. all possible strips, a certain fragment only, all possible successors/predecessors of a reference strip, etc.). The user can page through the selected subset with the menus *View/Next Page* or *View/Previous Page* (or by alternatively using the commands FS and BS).



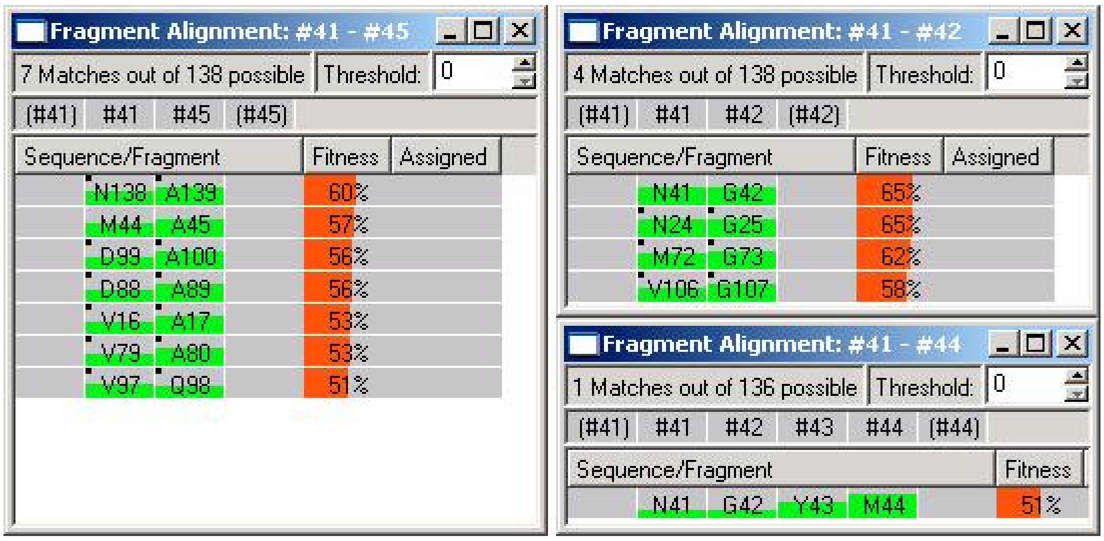
HNCA Cursor: x: HN=8.082 y: C=43.996 z: N=108.194 Level=16339, Selected 1 peaks: CA

Figure 37: Combining and mapping fragments with StripScope

As in SynchroScope CARA automatically finds and properly rotates all 3D spectra of the project, which have the same unique spin label set as the one opened in first place. The user can select from menu *View/Select Spectrum* which spectrum she wants to analyze (or by typing the commands NS or PS). The *Strips* menu enables the user to pick, label, move or delete the spins of the focus strip. All changes are immediately reflected in the spin system tree on the left.

CARA supports the fragment building process by an automatic strip matcher, which can be controlled by the *Setup* menu (the algorithm was introduced in chapter 4.2.1). The results of the algorithm are shown in the system tree (as split jigsaw pieces). In Figure 37 we can see that spin system 42 is a good candidate successor of system 41. The user can ask CARA to present all possible successors of system 41 as strips and then compare their slices (as shown in the figure). If she decides in favour of system 42, the menu command *Link to Reference* is executed from the context menu of the corresponding strip. This process is then continued to combine as many spin systems to fragments as possible. The user should set appropriate tolerance values (menu *View/Set Spin Tolerance*) to narrow the matches.

The fragment building is usually interleaved by runs of the mapping algorithm described in chapter 4.2.2. This supports for one part the decision about the placement of a fragment on the sequence (i.e. the sequence-specific assignment), but can also be used as a decision aid which fragments should be combined (see Figure 38).


Fragment Alignment: #41 - #45

7 Matches out of 138 possible | Threshold: 0

(#41) #41 #45 (#45)

Sequence/Fragment	Fitness	Assigned
N138 A139	60%	
M44 A45	57%	
D99 A100	56%	
D88 A89	56%	
V16 A17	53%	
V79 A80	53%	
V97 Q98	51%	

Fragment Alignment: #41 - #42

4 Matches out of 138 possible | Threshold: 0

(#41) #41 #42 (#42)

Sequence/Fragment	Fitness	Assigned
N41 G42	65%	
N24 G25	65%	
M72 G73	62%	
V106 G107	58%	

Fragment Alignment: #41 - #44

1 Matches out of 136 possible | Threshold: 0

(#41) #41 #42 #43 #44 (#44)

Sequence/Fragment	Fitness
N41 G42 Y43 M44	51%

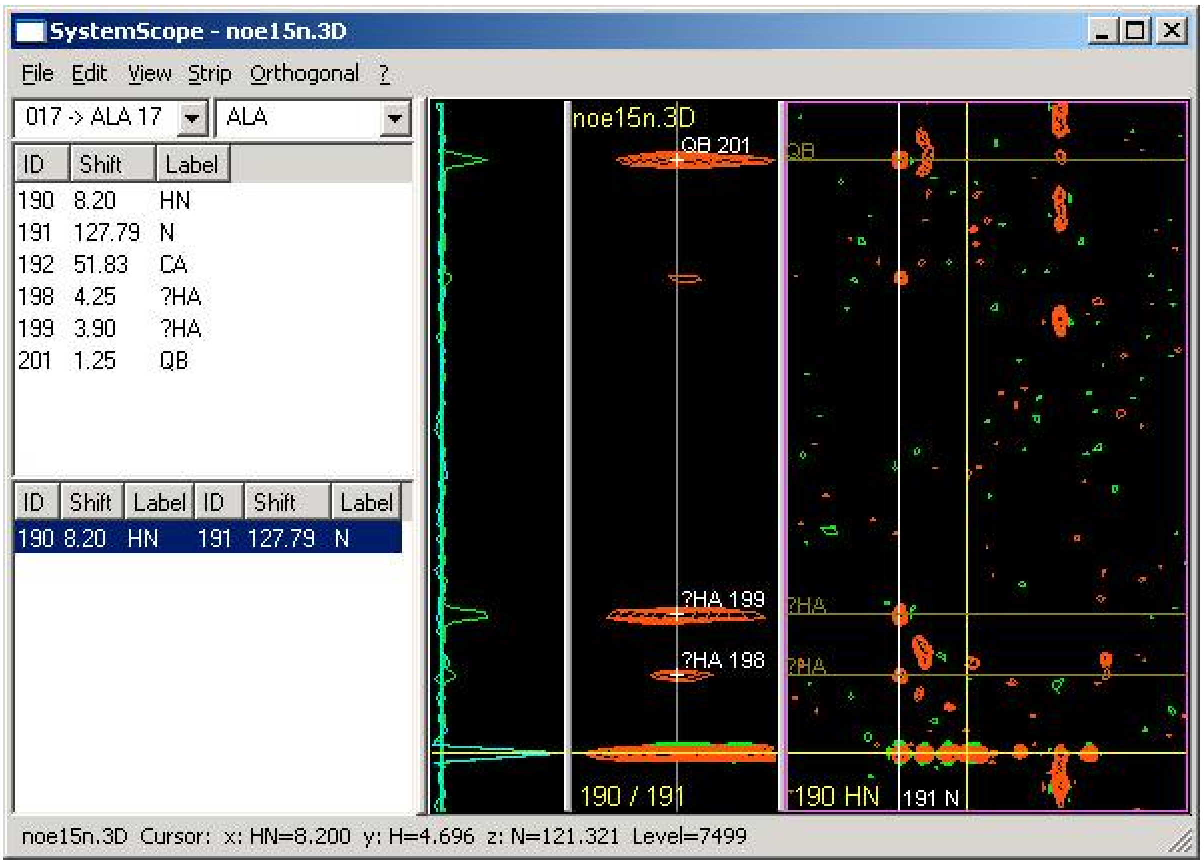
Figure 38: Sequence mapping of three fragment variations

The example shows that the fragment 41-45 has more ambiguity (i.e. less significance) than fragment 41-42. The decision becomes even easier after combining 41, 42, 43 and 44 into a fragment (because the spin systems were renumbered after final assignment for clarity reasons, we can immediately see that CARA made the right guess). In future versions of CARA this feature will be automated, so different fragment variations are automatically created and checked for their match to the sequence. Already today CARA can optionally cooperate with the program MAPPER [Güntert et al. 2000] (menu *Projects/Export/Mapper File* from within the Explorer) to get an optimal mapping considering all available fragments at once.

SynchroScope and StripScope can be used in parallel of course, i.e. there is no need to bring system and spin picking to an end before starting fragment building and assignment. There is no redundant information and each change in one window is immediately reflected in every other. It is even possible to open more than one StripScope or SynchroScope windows at the same time (e.g. if there are spectra with different sets of unique spin labels).

6.5 Heteronuclear Sidechain Resonance Assignment

The previous chapter showed how CARA can be used to do the sequence-specific assignment using triple-resonance spectra. The spin systems were combined to fragments and mapped to the sequence. We can therefore assume that we know at least the HN, N and CA chemical shifts of each residue, when the backbone assignment ends. This chapter gives a brief introduction how to use CARA for sidechain assignment. Again the spectra come from the FimD [Bettendorff, Pascal: unpublished data] project.


SystemScope - noe15n.3D

File Edit View Strip Orthogonal ?

017 -> ALA 17 | ALA

ID	Shift	Label
190	8.20	HN
191	127.79	N
192	51.83	CA
198	4.25	?HA
199	3.90	?HA
201	1.25	QB

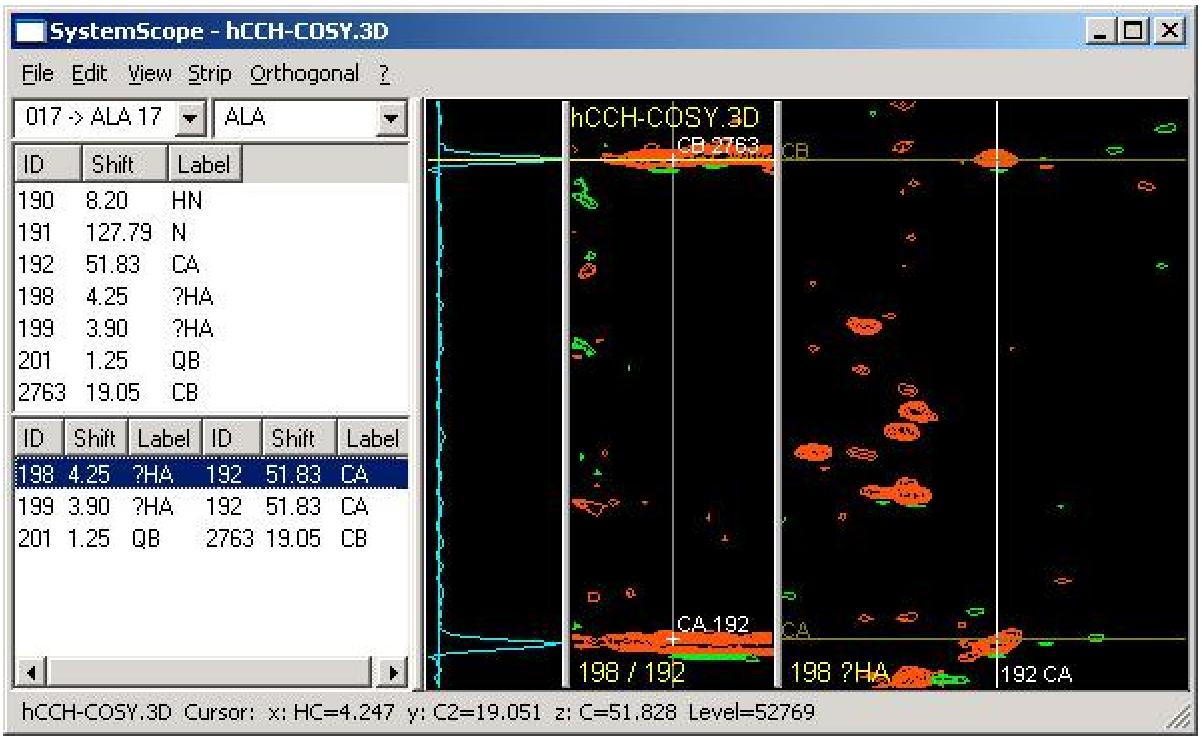
ID	Shift	Label	ID	Shift	Label
190	8.20	HN	191	127.79	N

noe15n.3D Cursor: x: HN=8.200 y: H=4.696 z: N=121.321 Level=7499

Figure 39: N/HN strip of ALA 17 in ^{15}N-ed. [^{1}H,^{1}H]-NOESY

The main tool for sidechain assignment is SystemScope. We will now start by selecting a ^{15}N-ed. [^{1}H,^{1}H]-NOESY spectrum in the spectrum pane of the CARA explorer and executing the function *Open SystemScope* from the context menu. SystemScope is a tool window consisting of four parts. The spin system to be analyzed is selected using the popup list in top left part of the window. All spins contained in the selected spin system are shown together with proposed strip positions (automatically inferred and updated to each spin system change by pathway simulation). The strip and its slice appear in the middle of the window as soon as the user selects one of the proposed strip positions and executes *Show Strip* from the context menu (or double-clicks on the list item).

The example strip initially shows four unpicked cross-peaks (see Figure 39). The one in the bottom is most likely caused by water and therefore not interesting. This can be confirmed by executing *Show Depth* from the context menu of the strip and watching the intensity spread along the N dimension of the orthogonal plane in the right part of the window. The top most peak of the strip can be picked and immediately identified to be QB by sequentially executing *Pick Spin* and *Label Spin* from the context menu (note that CARA allows to only select valid labels from the menu). There are two peaks remaining which both could be the expected HA. We pick both of them and execute *Force Spin Label* twice entering "?HA" in the label entry dialog (only one HA would be acceptable if the "?" was left out). The result should resemble Figure 39.



ID	Shift	Label
190	8.20	HN
191	127.79	N
192	51.83	CA
198	4.25	?HA
199	3.90	?HA
201	1.25	QB
2763	19.05	CB

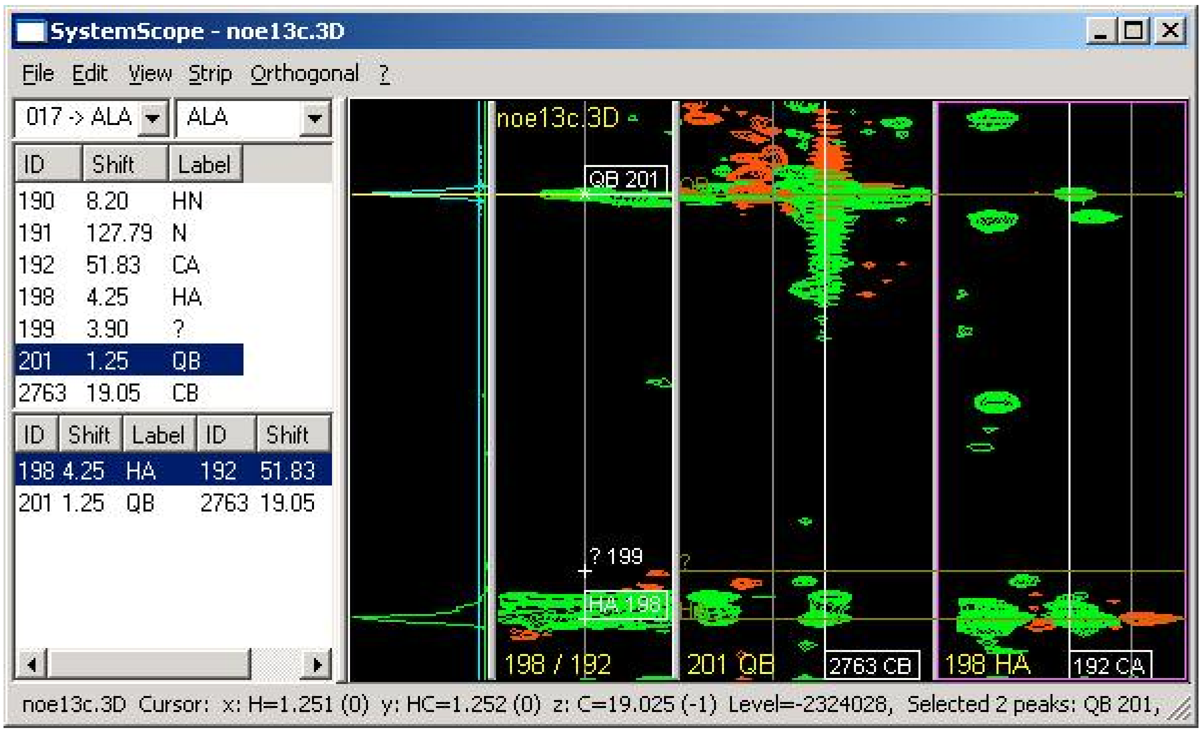
ID	Shift	Label	ID	Shift	Label
198	4.25	?HA	192	51.83	CA
199	3.90	?HA	192	51.83	CA
201	1.25	QB	2763	19.05	CB

```
hCCH-COSY.3D Cursor: x: HC=4.247 y: C2=19.051 z: C=51.828 Level=52769
```

Figure 40: HA/CA strip of ALA 17 in (H)CCH-COSY

Next a (H)CCH-COSY spectrum is opened using SystemScope. Initially two proposed strip positions are shown in the lower left list, one for each HA guess. The strip of spin 199 contains only noise, but the one other shows two peaks (see Figure 40, where CA 192 was already assigned). The CB can directly be picked and labeled (which is immediately reflected in the lists in the left part of the window). Since the correct HA is known now, the command *Accept Label* has to be executed (by selecting spin 198 in the list at the upper left side and using the context menu). The label of spin 199 is changed back to "?" and 198 is accepted to be HA (i.e. the "?" is removed).

Another way to find the CB spin makes use of a $^{1}H/^{13}C$ based 3D NOESY or TOCSY spectrum. Up to now we used the orthogonal pane in the right part of the window only to display the depth plane corresponding to the strip, i.e. the z/y plane at the strip position x_0. In Figure 40 for example the orthogonal pane shows the plane along both C dimensions at the strip position given by spin 198. But SystemScope also allows to use another than the strip position as the origin of the z/y plane.



ID	Shift	Label
190	8.20	HN
191	127.79	N
192	51.83	CA
198	4.25	HA
199	3.90	?
201	1.25	QB
2763	19.05	CB

ID	Shift	Label	ID	Shift
198	4.25	HA	192	51.83
201	1.25	QB	2763	19.05

noe13c.3D Cursor: x: H=1.251 (0) y: HC=1.252 (0) z: C=19.025 (-1) Level=-2324028, Selected 2 peaks: QB 201,

Figure 41: Detecting CB in the orthogonal plane with origin QB

In Figure 41 the spins HA and QB were first selected in the strip. Then *Show Orthogonal* was executed from the context menu. Two C/C planes appeared within the orthogonal pane, one at the position of QB and the other at the position of HA (i.e. the orthogonal pane was automatically split).

Since the spectrum was recorded with folded signals, the option *Show Folded* has to be enabled from the *View* menu (so the plane shows a shifted or mirrored copy of the spectrum, when an area outside the original sweep width is selected). The light vertical lines in the two C/C planes represent the borders of the original sweep area of the spectrum. The cursor position reported in the status bar is extended by the "quadrant number" in brackets (i.e. the ordinal offset from the original sweep area).

As expected the plane at origin HA already displays the CA spin (inferred by pathway simulation). If the CB wasn't identified yet, it could now be picked and labeled within the plane at QB, using the items *Pick Spin* and *Label Spin* from the context menu. Due to pathway simulation the menu only contains the expected labels.

The *Show Orthogonal* function is particularly useful for analyzing 3D TOCSY spectra. Starting e.g. from a CA/HA based strip, all other C/H pairs of the amino acid are visible as a characteristic "TOCSY tower" in the strip pane. By switching between the TOCSY and COSY spectra, the assignment is straight forward. If the user is not sure about which spins are expected to see in the selected strip, she can check the *Label Spin* menu (which always contains the label set corresponding to the selected strip position, inferred by the pathway simulation), or execute the *Show Spin Path* function in the list of proposed strip positions. The user can then make her way through the whole spectrum by executing *Show Orthogonal* for all spins of the selected strip. For each spin its immediate neighbors can usually be identified in the orthogonal plane by comparing the characteristic pattern of the TOCSY tower (or by directly picking them in case there are no ambiguous signals visible). Whenever a new spin is picked and labeled, the two lists in the left part of the window (showing all spins and proposed strip positions of the selected spin system) are immediately updated to reflect the current state of the assignment.

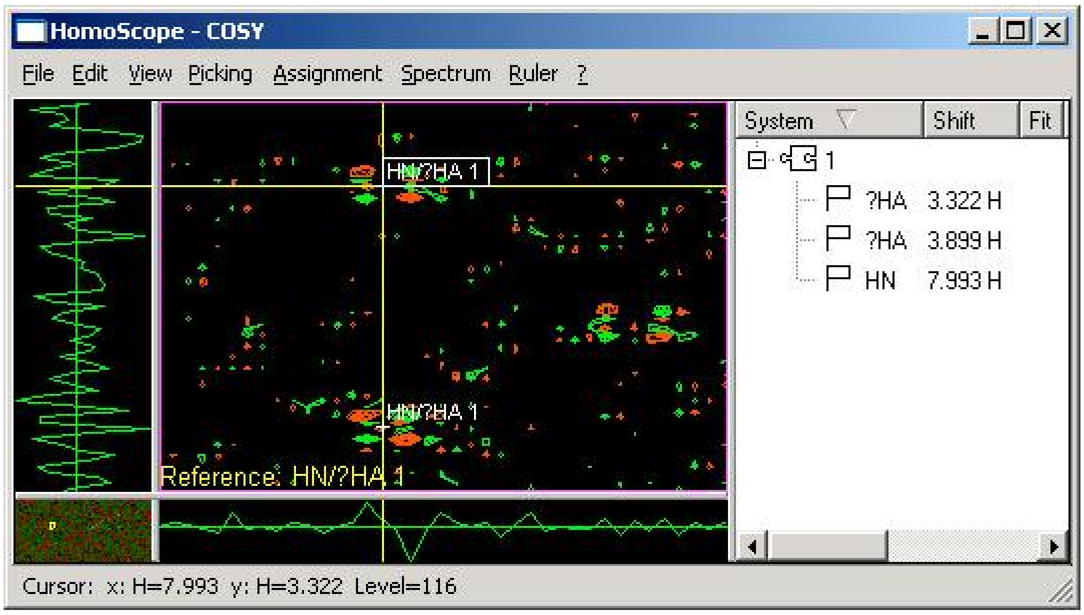
The completeness of a sidechain assignment can be checked for example by monitoring whether all peaks on a [^{1}H,^{13}C]-HSQC spectrum are identified. This is possible, if the [^{1}H,^{13}C]-HSQC is simultaneously displayed in a HomoScope window.

The assignment process shown is executed for each residue of the sequence. Finally there should be a complete spin list for each spin system, where each spin carries a label of an atom of the corresponding residue type. The atomlist can then be exported by either using the menu *Projects/Export/Atom List* from within the Explorer, or by selecting a spectrum in the *Spectra* pane of the Explorer and selecting *Export Atom List* from the context menu. In the latter case, the alias chemical shifts of the selected spectrum are used.

6.6 Homonuclear Assignment

Homonuclear assignment is rarely used today, since it is mainly useful for small macromolecules. This chapter briefly shows, how sequence-specific assignment can be accomplished in CARA using 2D COSY, TOCSY and NOESY spectra. The only tool needed is HomoScope. The following example nicely shows the effect of peak inference. The spectra come from the ER23 project [Damberger, Fred: unpublished data].

After the project is created (as described in chapter 6.3) and all needed spectra are loaded, the user should select a COSY spectrum in the corresponding Explorer pane and then execute *Open HomoScope* from the context menu. The window opens and we can start picking new spin systems in the amid/alpha region of the spectrum (upper left part) using the menu *Picking/Pick New System*. The labels of the peak can then be set using menu Picking/Label Peak (in the given region the labels HN/HA can immediately be assigned).



System	Shift	Fit
1		
?HA	3.322 H	
?HA	3.899 H	
HN	7.993 H	

Cursor: x: H=7.993 y: H=3.322 Level=116

Figure 42: Picking a Glycine in a 2D [^{1}H,^{1}H]-COSY spectrum

The pattern in Figure 42 is typical for a Glycine. The second peak is picked by simply placing the cursor on it and activating *Picking/Extend Vertically* , which doesn't create a new spin system but extends the previously selected reference spin system by a new spin. The horizontal spin of the new peak already carries the HN label (because it is identical to the horizontal spin of the reference spin system). Since we actually know its a Glycine, we could right know classify the spin system as of AX type, but we postpone this for didactic reasons. Instead we label both vertical spins as ?HA (spin systems allow to carry as many equal labels as needed as long they are in draft state, notified by the "?" symbol). The menu View/Show List displays a tree list in the right part of the window showing all spin systems and their spins (see Figure 42). We see that CARA inferred this two peaks from three spins only.

If we show the whole spectrum (menu *View/Fit Window*), seven additional peaks become visible which were automatically inferred. The spin system pattern can be made clearer by creating horizontal and vertical rulers along all peaks (by selecting them and executing menu *Ruler/Add Horizontal Ruler* and *Add Vertical Ruler*). The result should look like Figure 43. CARA in fact inferred nine peaks from three spins only. If any of these spins is moved, all peaks are moved accordingly. In peaklist oriented programs each peak was an independent entity and thus had to be moved explicitly. The same applies to label and other changes (see below).

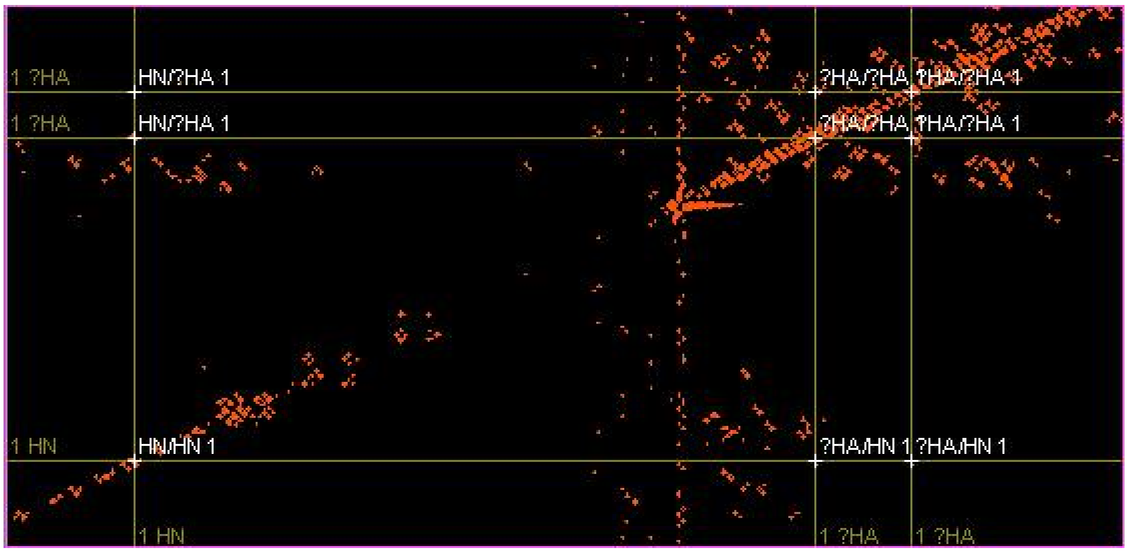
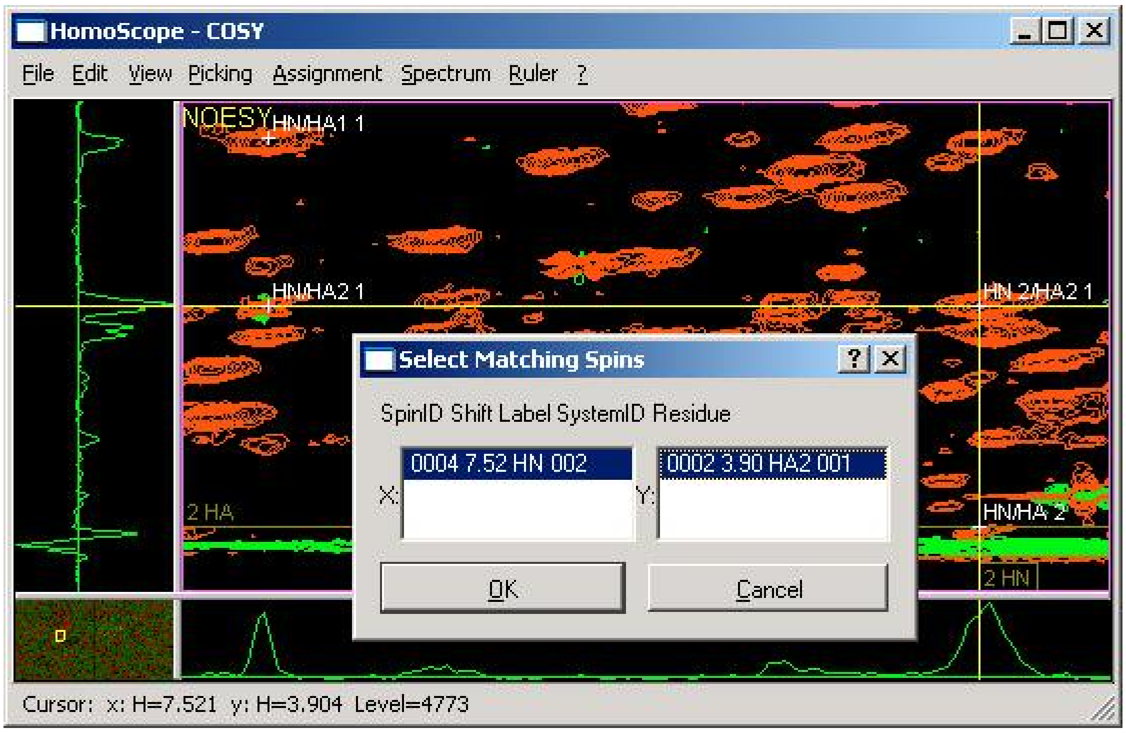


Figure 43: Detail of spin system pattern with rulers (positive contours only)

Now its time to set the spin system type. This can be done by either selecting one of the peaks and then executing the menu *Assignment/Set System Type*, or by selecting the system in the tree list and activate the function from its context menu. The popup list of the dialog should contain the AX system type (if the repository was properly set up). Then the two ?HA spins are relabeled as HA1 and HA2 either using the *Picking* menu or *Label Spin* in the tree list. The label list should contain the labels appropriate to a Glycine. The label change is immediately reflected on each peak where it occurs. If peaks temporarily disappear in the plane (because the labels don't fit the system type), continue in the tree list. If now the *Show Alignment* function is executed for the spin system, all Glycines of the sequence are proposed with equal significance (i.e. it is not yet possible to uniquely assign the spin system).

The process continues as shown by picking and classifying all spin systems using the COSY and optionally the TOCSY spectrum. After or - if possible - during this process the user would try to connect the spin systems to fragments using the HA_{i-1}/HN_i connectivities in NOESY (one can switch between spectra using the *Spectrum* menu or the commands *NS/PS*). For this purpose the cursor is placed on the peak and the menu *Picking/Propose Peak* is executed. A dialog appears showing all spins around the cursor position (see Figure 44). If too many spins are displayed, one should narrow the tolerance using menu *Picking/Set Verti. Tolerance*.



SpinID Shift Label SystemID Residue

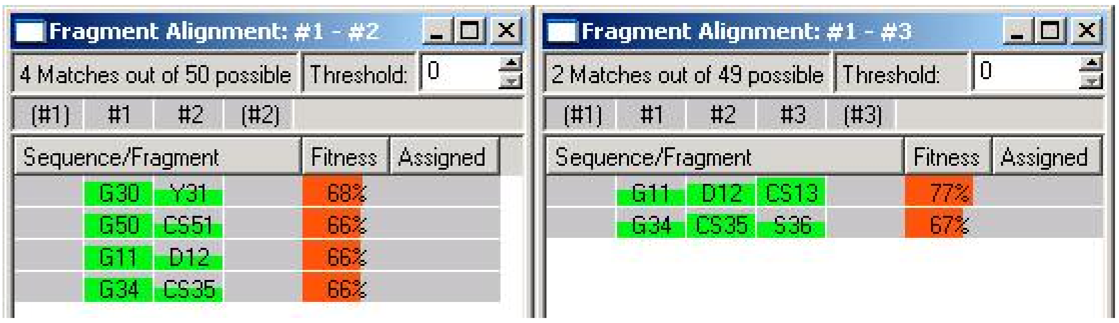
X: 0004 7.52 HN 002

Y: 0002 3.90 HA2 001

Cursor: x: H=7.521 y: H=3.904 Level=4773

Figure 44: Identifying HA_{i-1}/HN_i connectivities in NOESY

If the user has made her choice, the cross-peak representing the connectivity appears, but the spin systems are not yet linked together. This can be done by selecting the cross-peak and then executing the menu *Assignment/Link Systems*. CARA shows a dialog listing both fragment possibilities: *1-2* or *2-1*. The first is the one we want. This process is continued as long as possible. At any point the *Show Alignment* function can be executed. Both the spin system types and the Hydrogen chemical shift statistics are taken into account by CARA to calculate the alignment (see chapter 4.2.2). The first line in the right part of Figure 45 is already the correct solution and could immediately be assigned using the *Assign* context menu from within the alignment window. The figure also documents the influence of the fragment length on the number and assessment of possible mappings (i.e. another mapping was considered the right one in case of fragment length two).


Fragment Alignment: #1 - #2

4 Matches out of 50 possible | Threshold: 0

(#1) #1 #2 (#2)

Sequence/Fragment		Fitness	Assigned
G30	Y31	68%	
G50	CS51	66%	
G11	D12	66%	
G34	CS35	66%	

Fragment Alignment: #1 - #3

2 Matches out of 49 possible | Threshold: 0

(#1) #1 #2 #3 (#3)

Sequence/Fragment			Fitness	Assigned
G11	D12	CS13	77%	
G34	CS35	S36	67%	

Figure 45: Fragment alignment of length 2 (left) and length 3 (right)

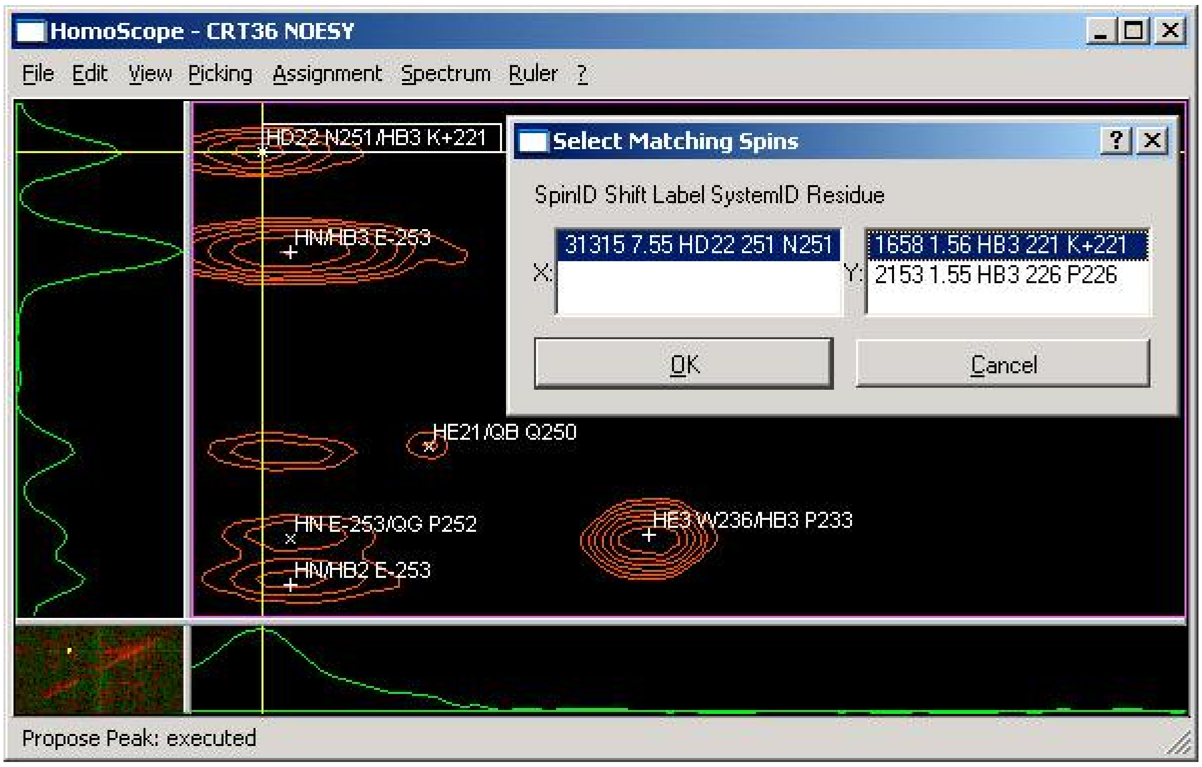
6.7 Constraint Gathering

With the advent of algorithms like ATNOS [Herrmann et al. 2002a] it should be no longer necessary to manually do constraint gathering and peak volume determination. The process shown in this chapter is still applicable for special cases, for which these kinds of algorithms fail, or to review and correct the output of such algorithms.

In the chapters 2.4 and 4.4 the concept of distance constraints was already introduced. Depending on whether the process is based on homonuclear or heteronuclear spectra, either the HomoScope or PolyScope tool window is used. As usual the tools can be opened by selecting a spectrum in the spectrum pane of the Explorer and executing the corresponding command from the context menu.

The same concepts apply to both versions of the process (homo- and heteronuclear): all spins identified during backbone and sidechain assignment are immediately visible as cross-peaks, since they can be inferred by the pathway simulation introduced in chapter 4.3.1. The remaining intensities of the spectrum, which have not yet been assigned, are thus expected to be the interesting distance constraints (i.e. the arbitrary inter-nuclear relations representing structural information).

The following example is a continuation of the homonuclear assignment shown in the last chapter, this time showing spectra from the CRT36 project [Ellgaard et al. 2002]. A 2D [^{1}H,^{1}H]-NOESY spectrum is selected in the spectrum pane of the Explorer. The command *Open HomoScope* is then executed from the context menu. The window opens, displaying all cross-peaks calculated by peak inference. During constraint gathering no new spins are created, but existing spins are linked together. The user places the cursor on the intensity peaks not already picked. The menu *Picking/Propose Peak* opens a dialog box displaying all existing spins having a position corresponding to the cursor position (see Figure 46). The spins are selected using Eq. 6 (see chapter 4.2.1) and displayed in descending order of the agreement (i.e. the most likely candidates are on top of the list). The tolerance value can be set using menu *Picking/Set Verti. Tolerance* (which is valid for both dimensions since they have equal atom types).



SpinID Shift Label SystemID Residue

X: 31315 7.55 HD22 251 N251

Y: 1658 1.56 HB3 221 K+221
2153 1.55 HB3 226 P226

Propose Peak: executed

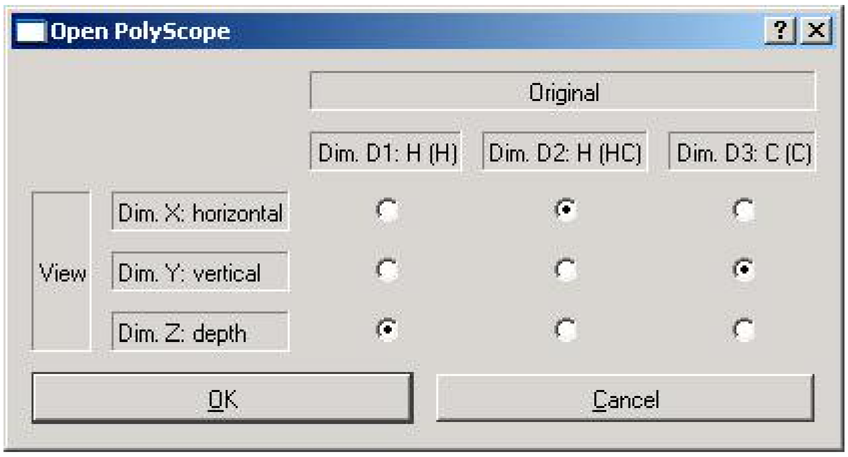
Figure 46: Selecting a spin pair corresponding to a cursor position

The dialog shown in the figure gives two alternative spins to choose from for the vertical dimension. The two spins with id numbers 31315 and 1658 were selected by the user and are in the following connected by a spin link, which is then displayed as a cross-peak. The user can recognize from the labels whether the cross-peaks connect intra- or inter-residual spins. Usually only inter-residual spins are connected by links, since the intra-residuals can be inferred by pathway simulation.

If there are no more unpicked intensity peaks left within the interesting areas of the spectrum, the process is finished. At this point a peaklist can be generated and either be saved to a file (using menu *File/Export/Peaklist*) or directly transferred to

MonoScope for peak volume determination (using *File/Export/Peaklist to MonoScope*, see next chapter).

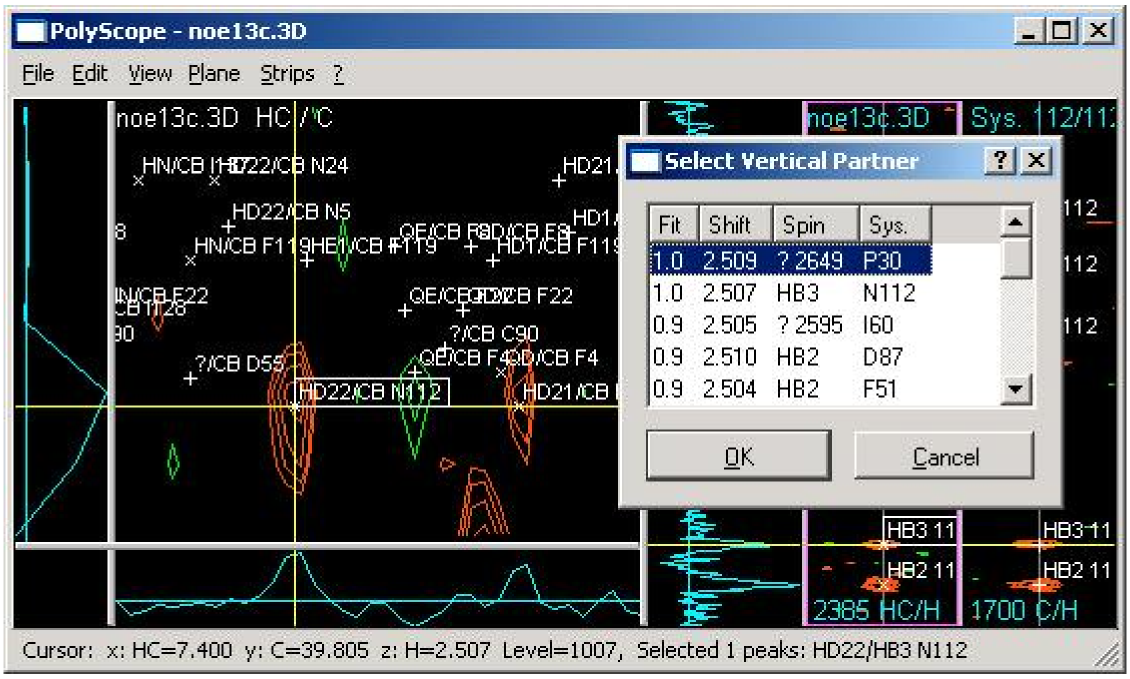
In the 3D case the analysis works similar. The distance contraints are identified using the PolyScope tool instead. The ^{13}C-ed. [^1H,^1H]-NOESY has probably to be rotated to present the $^1H/^{13}C$ correlation in the plane and the NOESY dimension in the strip.


Open PolyScope

		Original		
		Dim. D1: H (H)	Dim. D2: H (HC)	Dim. D3: C (C)
View	Dim. X: horizontal	○	●	○
	Dim. Y: vertical	○	○	●
	Dim. Z: depth	●	○	○

Figure 47: Common dialog to change the rotation of the dimensions

Up to this point all spectra were directly opened, assuming that the spectrum is displayed using the dimension order specified by its spectrum type. The spectrum is as usual selected in the spectrum pane of the Explorer. The command *Open PolyScope (rotated)* is then executed from the context menu. The dialog of Figure 47 appears. The identification of NOESY peaks then works in a similar way described in the homonuclear example (using the menu command *Strips/Propose Spin*). Figure 48 shows a ^{13}C-ed. [^1H,^1H]-NOESY of the FimD [Bettendorff, Pascal: unpublished data] project. The strips display all spins of the spin system selected in the plane. The cursor is then positioned on an intensity peak within the strip. If the command *Propose Spin* is executed, CARA presents a list of all spins having chemical shift positions corresponding to the cursor position (sorted in descending order of the agreement calculated with Eq. 6).



Fit	Shift	Spin	Sys.
1.0	2.509	? 2649	P30
1.0	2.507	HB3	N112
0.9	2.505	? 2595	I60
0.9	2.510	HB2	D87
0.9	2.504	HB2	F51

```
Cursor: x: HC=7.400 y: C=39.805 z: H=2.507 Level=1007, Selected 1 peaks: HD22/HB3 N112
```

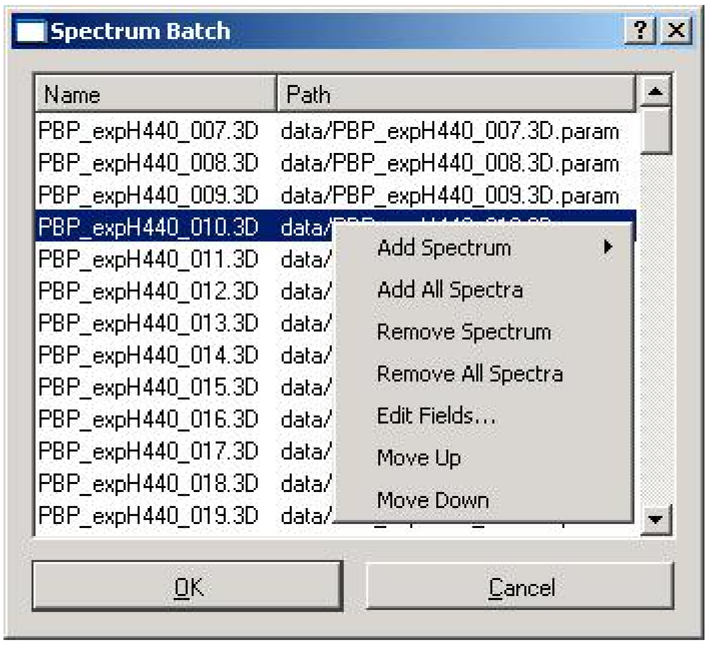
Figure 48: Identifying peaks in a ^{13}C-ed. [^{1}H,^{1}H]-NOESY using PolyScope

The user can select a spin from this list, which is then displayed as a cross-peak in the strip. The selection of the correct spin is difficult if there are many spins at the same chemical shift. Often the decision is only possible during structure calculation. Algorithms like CANDID [Herrmann et al. 2002b] are able to handle these *ambiguous distance constraints* and their output can again be used in CARA (so the user does no longer have to guess about the assignments).

6.8 Peak Volume Determination

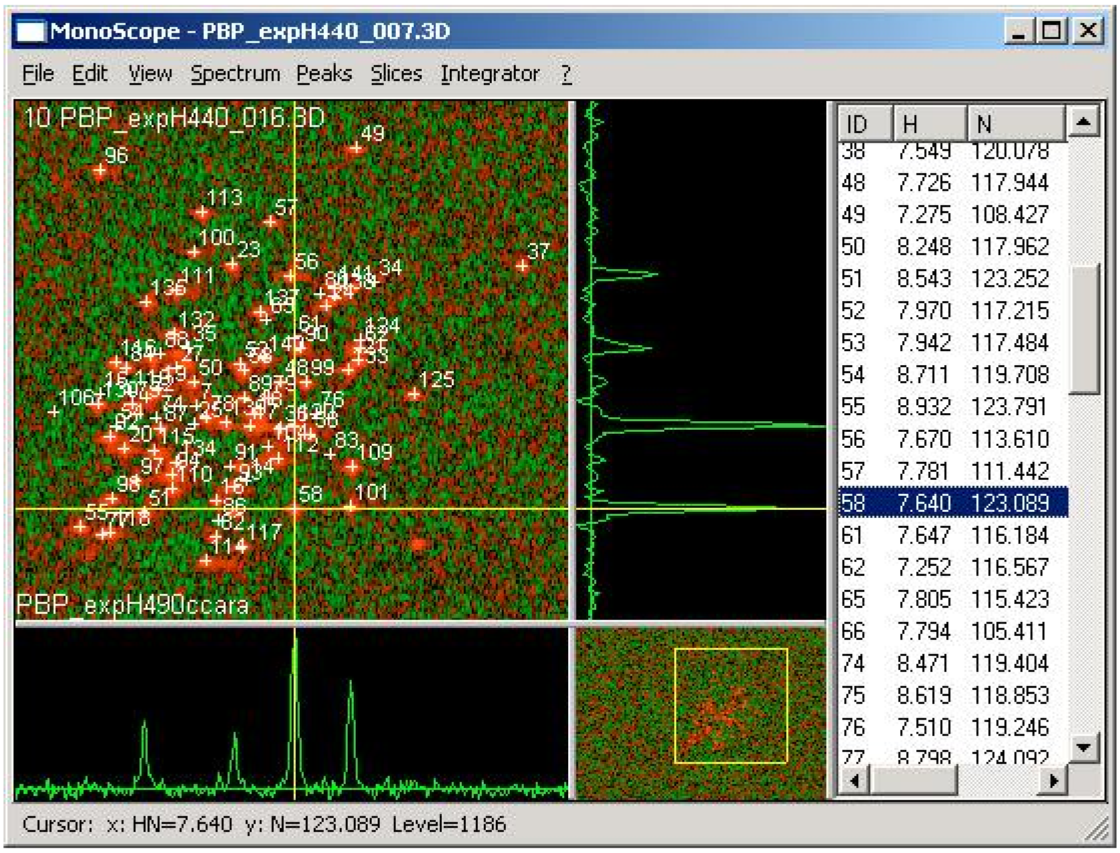
In this section we give a short overview on how to use CARA for peak volume determination. We take the D_2O exchange of Pheromone binding protein from Bombyx mori at pH 4.5 as an example [Lee et al. 2002]. The same procedure can be applied to NOESY peak volume determination with the difference of having only one spectrum instead of a series of spectra.

The spectra of the exchange series can be imported to the *Project* with the file selector dialog in one single step if they are all located in the same directory. Next, the first spectrum of the series is opened with MonoScope and the corresponding peaklist is imported. Figure 49 shows how the spectra are then added to the ordered batch list of the peaklist. Because the order of the spectra corresponds to the sort order of their names, this can also be done with one mouse click (menu item *Add All Spectra*).



Name	Path
PBP_expH440_007.3D	data/PBP_expH440_007.3D.param
PBP_expH440_008.3D	data/PBP_expH440_008.3D.param
PBP_expH440_009.3D	data/PBP_expH440_009.3D.param
PBP_expH440_010.3D	data/
PBP_expH440_011.3D	data/
PBP_expH440_012.3D	data/
PBP_expH440_013.3D	data/
PBP_expH440_014.3D	data/
PBP_expH440_015.3D	data/
PBP_expH440_016.3D	data/
PBP_expH440_017.3D	data/
PBP_expH440_018.3D	data/
PBP_expH440_019.3D	data/

Figure 49: Spectrum Batch List

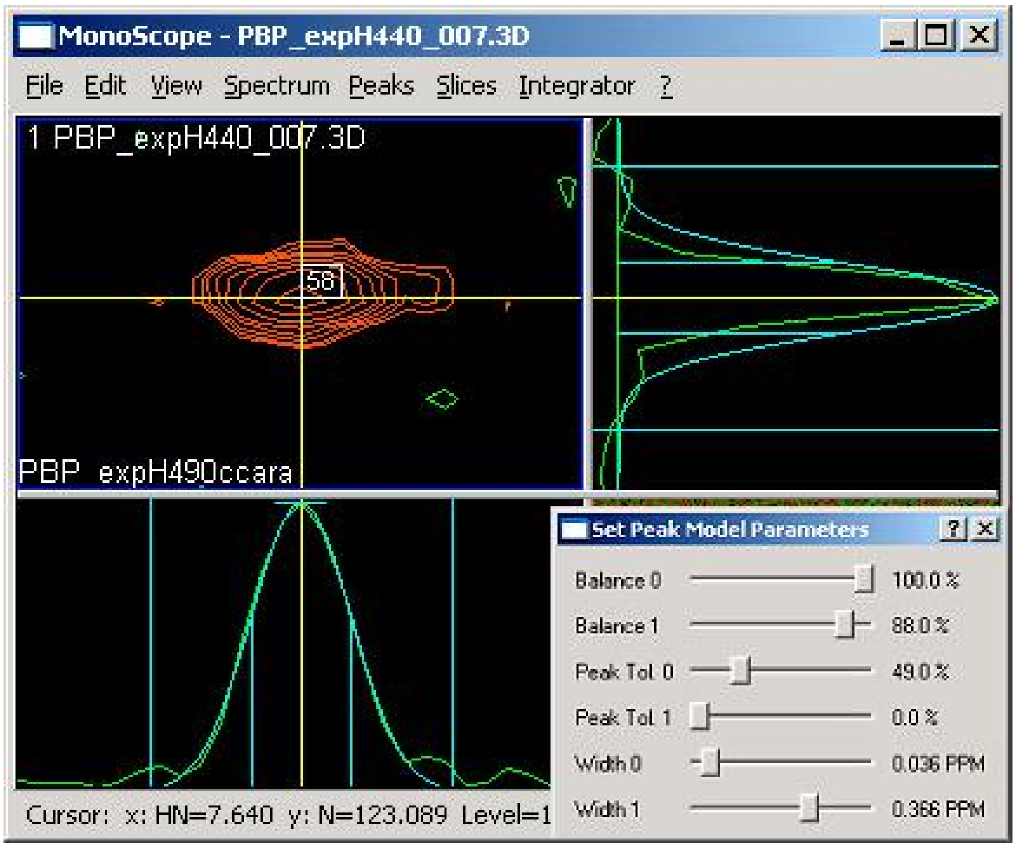

MonoScope - PBP_expH440_007.3D
File Edit View Spectrum Peaks Slices Integrator ?
10 PBP_expH440_016.3D
PBP_expH490ccara

ID	H	N
38	7.549	120.078
48	7.726	117.944
49	7.275	108.427
50	8.248	117.962
51	8.543	123.252
52	7.970	117.215
53	7.942	117.484
54	8.711	119.708
55	8.932	123.791
56	7.670	113.610
57	7.781	111.442
58	7.640	123.089
61	7.647	116.184
62	7.252	116.567
65	7.805	115.423
66	7.794	105.411
74	8.471	119.404
75	8.619	118.853
76	7.510	119.246
77	8.798	124.092

Cursor: x: HN=7.640 y: N=123.089 Level=1186

Figure 50: MonoScope with peaklist, spectrum, slices and overview

The first spectrum in the batch list is used as a reference to load the peaklist to be integrated. Figure 50 shows the tool window, where all peaks can be verified within all spectra of the batch list. The program offers efficient commands to quickly navigate to the interesting parts of a spectrum. It is also possible to move or pick additional peaks, if necessary. Peak positions can be individually adapted to each spectrum using the *Move Peak Alias* command from the *Peaks* menu. This feature allows the program to follow the *trace* of a peak along all spectra of the batch list.



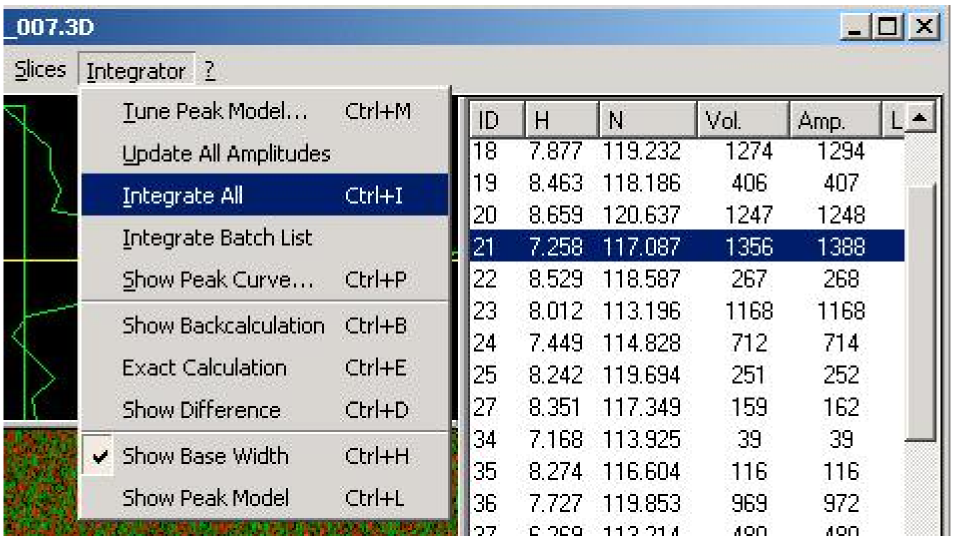
Balance 0 100.0 %
Balance 1 88.0 %
Peak Tol 0 49.0 %
Peak Tol 1 0.0 %
Width 0 0.036 PPM
Width 1 0.366 PPM

Cursor: x: HN=7.640 y: N=123.089 Level=1

Figure 51: Adjusting model parameters of a selected peak. The model is plotted on top of the original slices

The user interactively tunes the model parameters (line width, Gauss/Lorentz balance, peak tolerance, etc.) by selecting representative peaks and moving the sliders as shown in Figure 51. Every change immediately affects the displayed model curve. The HN dimension in the figure has already been adjusted by the user. The N dimension still needs attention. When the parameters are suitably adjusted, the volume determination can be started using the *Integrator* menu (see Figure 52).

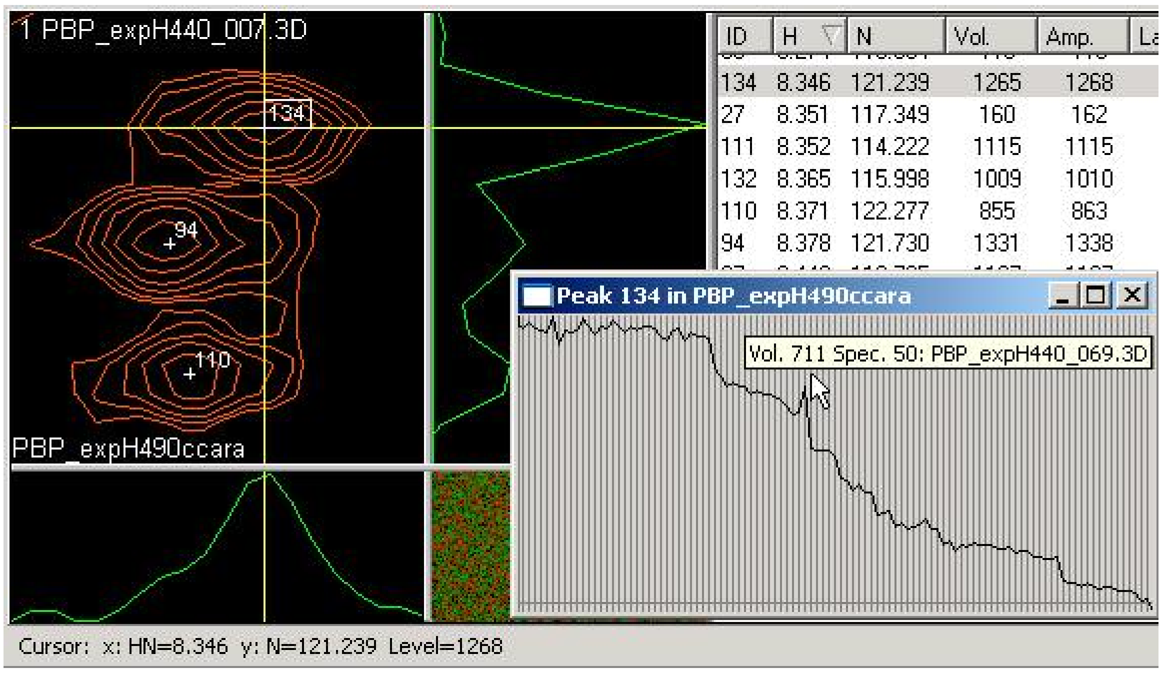
It is possible to integrate a single spectrum or alternatively the whole batch list in one pass. The volume determination of the batch list of Pheromone Binding Protein (107 spectra) took less than 10 seconds on a laptop (Windows XP, 2.4 GHz). The results of the volume determination are immediately visible in the peaklist (Vol. column).



ID	H	N	Vol.	Amp.	L
18	7.877	119.232	1274	1294	
19	8.463	118.186	406	407	
20	8.659	120.637	1247	1248	
21	7.258	117.087	1356	1388	
22	8.529	118.587	267	268	
23	8.012	113.196	1168	1168	
24	7.449	114.828	712	714	
25	8.242	119.694	251	252	
27	8.351	117.349	159	162	
34	7.168	113.925	39	39	
35	8.274	116.604	116	116	
36	7.727	119.853	969	972	

Figure 52: The Integrator menu and the integrated peaklist (columns *Vol.* and *Amp.*)

The user can display the exchange curve along the spectra of the batch list for each peak. Irregularities are thus easily recognized and assessed to the originating spectrum. Figure 53 shows spectrum 1 and the overlapping peaks 134, 94 and 110. Additionally he exchange curve of peak 134 is presented. The peak volume assigned to spectrum 50 seems to be irregular (information for each point of the curve can be indicated by use of a so called *ToolTip*). The user will visit the corresponding spectrum, do the appropriate adjustments and then restart the volume determination. Because volume determination nearly happens in real-time, the user can apply an incremental and iterative optimization approach.



ID	H	N	Vol.	Amp.	La
134	8.346	121.239	1265	1268	
27	8.351	117.349	160	162	
111	8.352	114.222	1115	1115	
132	8.365	115.998	1009	1010	
110	8.371	122.277	855	863	
94	8.378	121.730	1331	1338	

Peak 134 in PBP_expH490ccara

Vol. 711 Spec. 50: PBP_expH440_069.3D

Cursor: x: HN=8.346 y: N=121.239 Level=1268

Figure 53: Three overlapping peaks and the exchange curve shown for one of them

The user can assess the adequacy of the model parameters and the quality of the volume determination by comparing the backcalculated and difference spectrum with the original one (Figure 54). Eq. 14 shows how to backcalculate a spectrum using the available data (model, volumes and positions). The spectra are calculated in real-time and immediately reflect changes to the database.

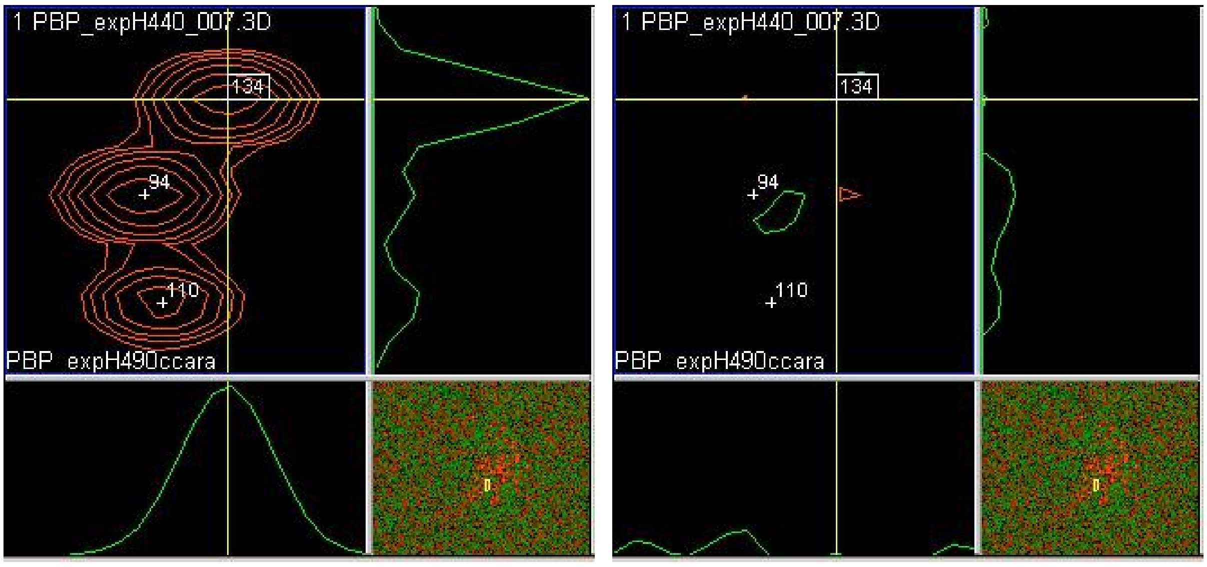


Figure 54: The backcalculated spectrum (left) corresponding to Figure 53 and the difference spectrum (right)

In the right part of Figure 54 the peaks have nearly disappeared, i.e. there is little intensity remaining. In case a peak was covered before or accidentally not picked, it would now appear in the difference spectrum. The user could pick the peak, reintegrate and immediately see the effect in the spectrum. This incremental and iterative process would continue until the user was satisfied by the result.

6.9 Phasing

As part of the processing the phase of an NMR spectrum has to be adjusted to get pure absorptive signals (as described in chapter 2.2). Most common processing programs are controlled by a command language (i.e. using a terminal screen) and cannot graphically present spectra by themselves. If the user wants to visually assess the effect of the parameter settings, she has to make use of other programs offering this capability. CARA has a dedicated tool window, which allows the user to load all real and imaginary parts of a spectrum and to interactively adjust the phase angles along all dimensions, having immediate feedback of the adjustments.

The corresponding tool in CARA is called Phaser and can be accessed by menu *Tools/Phase Spectrum* from within the CARA Explorer. After executing the menu item, the user has to select a spectrum file from the file system representing the real part. If done so the Phaser opens showing the selected spectrum. The user has then to explicitly open the imaginary part spectra, one for each dimension and corresponding to the real part spectrum. The File menu contains tree items *Open Imag. Dim X*, *Y* and *Z* by which the imaginary spectrum files can be loaded. For a 2D spectrum only X and Y are enabled and the user has thus to select two imaginary spectrum files, so she has finally loaded three spectra (*rr*, *ir* and *ri*).

Figure 55 shows Phaser in action (the spectrum comes from the CRT100 project [Bettendorff, Pascal: unpublished data]). The user has already adjusted the pivot point (menu *Slices/Set Pivot*). This is the position in the spectrum where the linear phase *Phi 1* has no influence (visualized by the darker of the two cursor lines).

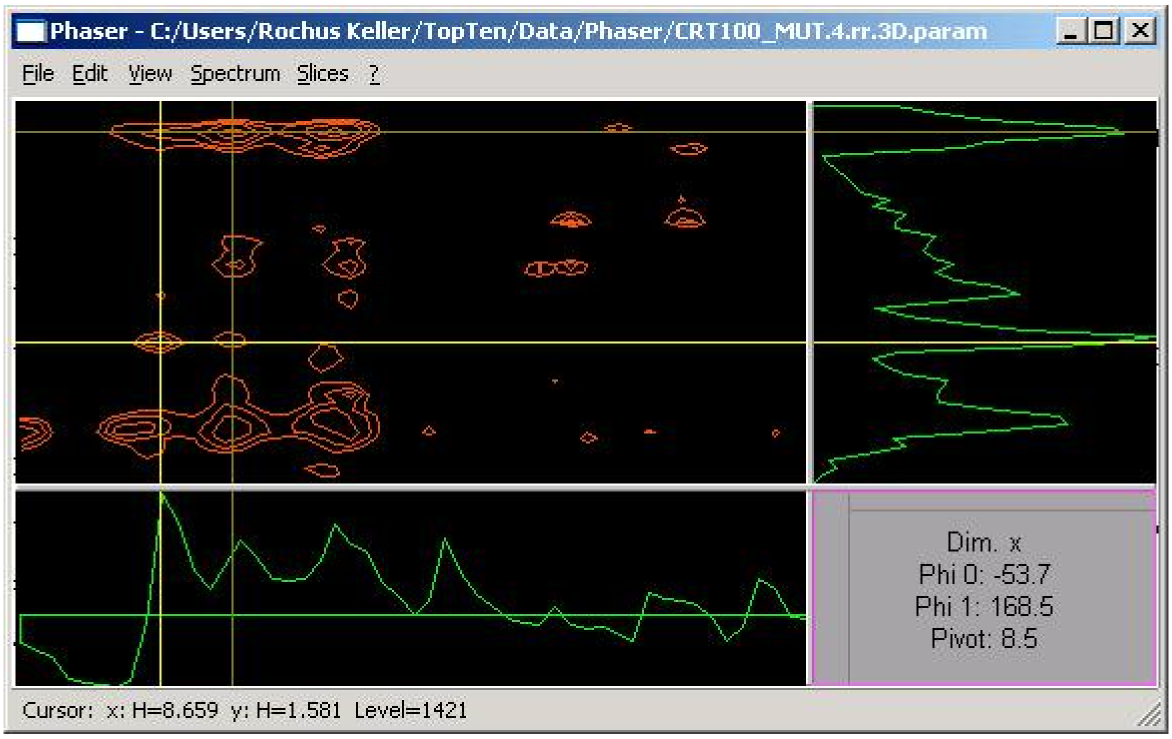

Phaser - C:/Users/Rochus Keller/TopTen/Data/Phaser/CRT100_MUT.4.rr.3D.param
File Edit View Spectrum Slices ?
Dim. x
Phi 0: -53.7
Phi 1: 168.5
Pivot: 8.5
Cursor: x: H=8.659 y: H=1.581 Level=1421

Figure 55: Using Phaser with a 2D spectrum

The phases can be changed in different ways, either by dragging the mouse in the lower right pane (the *control panel*, where *X* direction corresponds to *Phi 0* and *Y* to *Phi 1*, and each can be separately or both simultaneously changed), by pressing the cursor keys (horizontal for *Phi 0* and vertical for *Phi 1*) or by explicitly entering the numbers in a dialog box (menu *Slices/Set Phase*). In all cases the effect of the new angles is simultaneously calculated and the spectrum display accordingly changed (the slice display is even animated during the mouse drag to ease visual optimization). There is only one dimension active at any time. The control panel displays the active dimension together with its current phase and pivot values. The active dimension is changed by either clicking into a slice pane or by the menu item *Slices/Use Dim. X*, *Y* or *Z*. The user would usually select a prominent peak in one region of the spectrum, use it as the pivot and adjust *Phi 0* for all dimensions. She would then select another prominent peak and compare the two regarding *Phi 0* (by optionally superimposing their slices). If necessary the second peak can be corrected in regard of *Phi 1*. This procedure is usually repeated with different peaks in different areas of the spectrum, until all of them look evenly absorptive. The phase angles can then be read out to adjust the processing software in use.

6.10 Printing

Scientists spend a major part of their time writing reports and papers. An assignment has usually to be documented and presented on the basis of the used spectra. It is therefore essential that a software package offers an efficient way to not only render information interactively to screen, but also to a printable form. Since a printed document obeys other formatting requirements than a screen display, it is not appropriate to just print out the contents of the tool windows.

CARA follows a WYSIWYG (*what you see is what you get*) approach in that it allows to render the contents of a tool window to a generic print preview window, where the user can adjust the look of the presentation by various interactive parameters. The user would work as usual with one of the tool windows and then execute the menu item *File/Print Preview* as soon as she has zoomed into the area to be presented. The window of Figure 56 appears (the spectra come from the GroES project [Fiaux et al. 2002]).

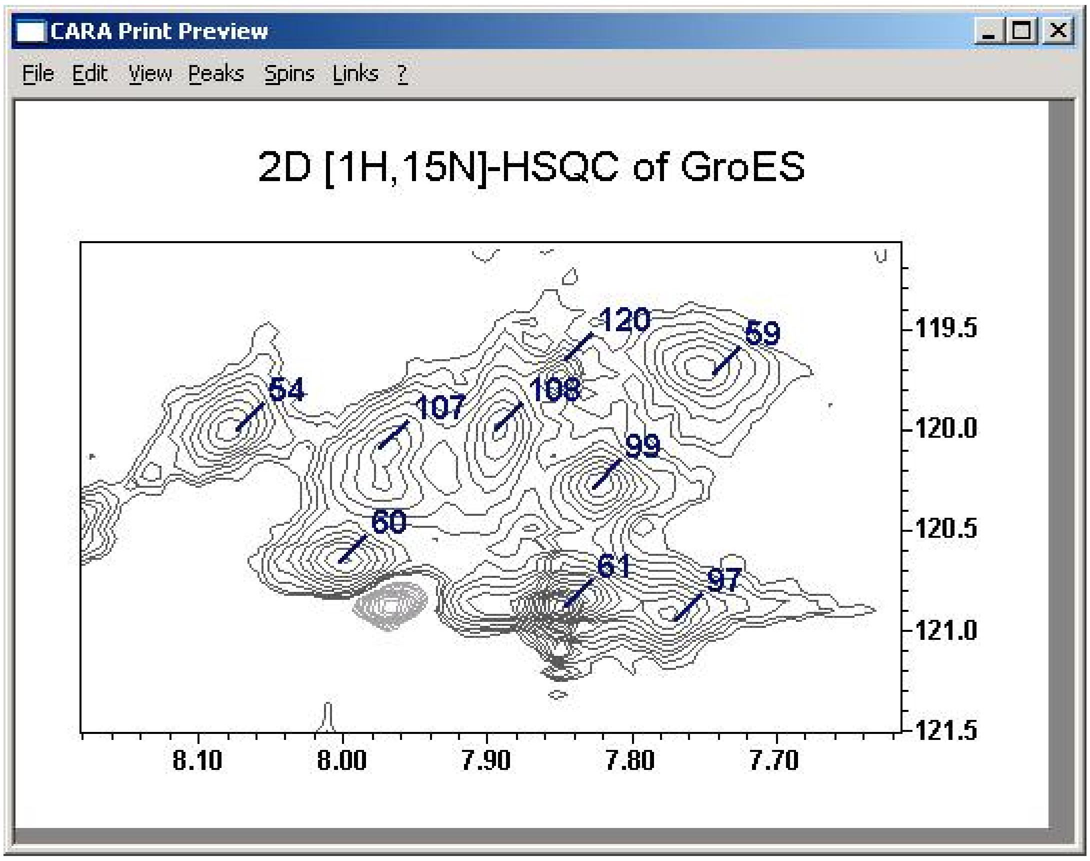


Figure 56: Print Preview window showing HN/N cross-peaks

There is a menu bar and also a context menu available. The user adjusts the page size and orientation using the *View* menu. The print output is cut at the borders of this virtual page. Each change of any of the fifty parameters is immediately visible. Some features (such as the placement and resolution of the scale numbers) are automatically adjusted by the software for convenience. Parameter settings can be saved to or loaded from configuration files using the menu commands *File/Save Settings* and *File/Load Settings*. If the user is satisfied with the look she can send the page to a printer or save it to a file using the menu *File/Print*. The printed page looks exactly like the one displayed on screen (provided the printer can display colors). It seems appropriate for casual presentations to directly use the printed page. For paper publications instead, it usually makes sense to enhance the drawings by use of dedicated picture editing packages (which are able to load the postscript output of CARA).

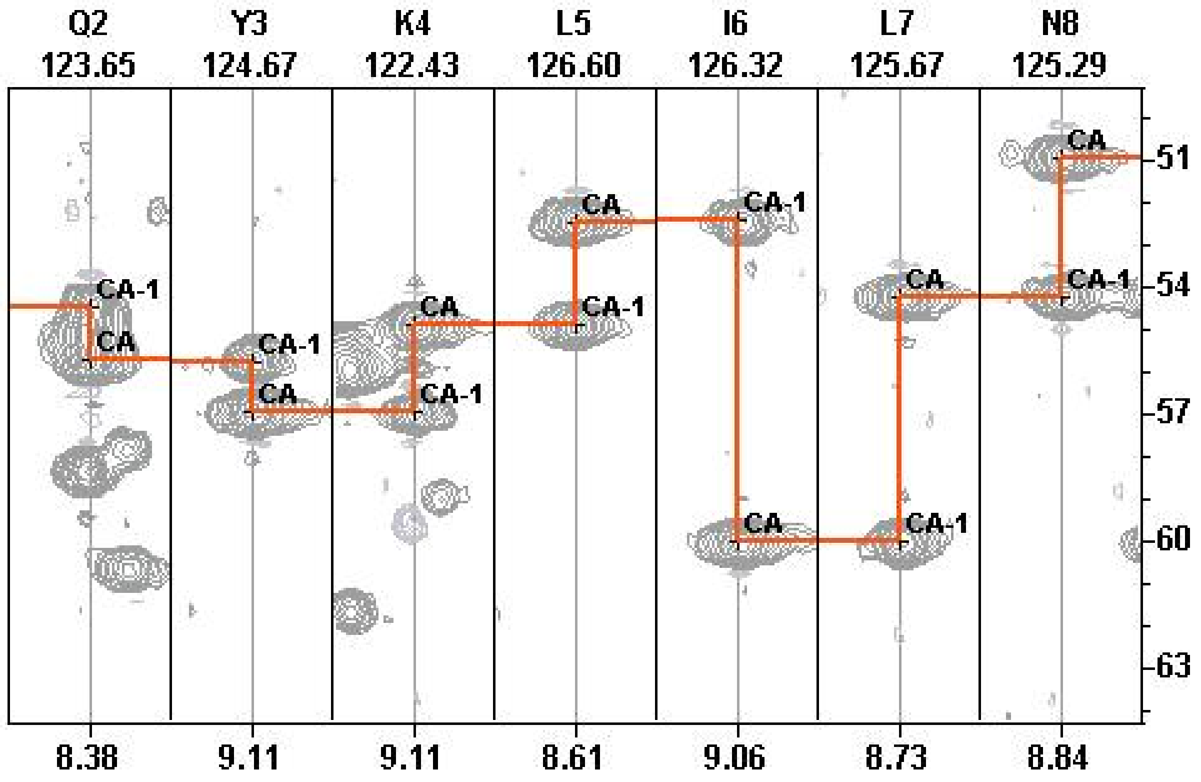
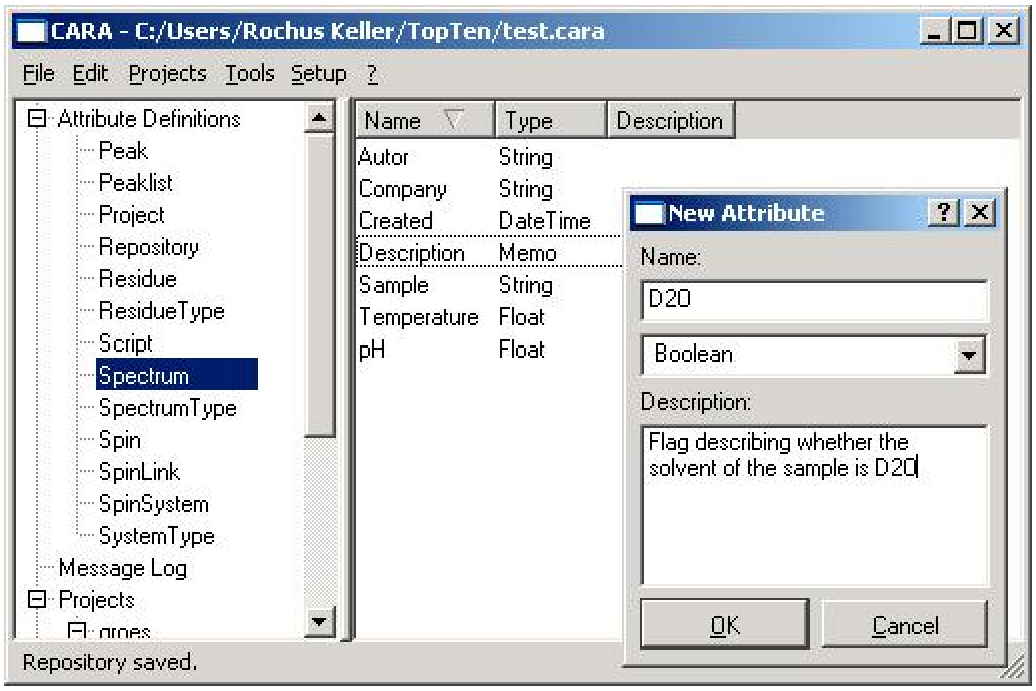


Figure 57: Print out of an assigned HNCA strip fragment

6.11 Extensibility by Dynamic Attributes

Each primary CARA object is extensible on the fly by new attributes by the user. The *Attribute Definitions* category of the CARA Explorer lists all object classes, which can be extended (Figure 58). The attributes pane on the right lists all attributes of the selected object class. The user executes the *New Attribute* command of the context menu of the pane, which opens a dialog box, where an attribute name, one of the supported data types and an optional description of the attribute can be entered. The attribute definitions are then part of the repository and immediately applicable.



Name	Type	Description
Autor	String	
Company	String	
Created	DateTime	
Description	Memo	
Sample	String	
Temperature	Float	
pH	Float	

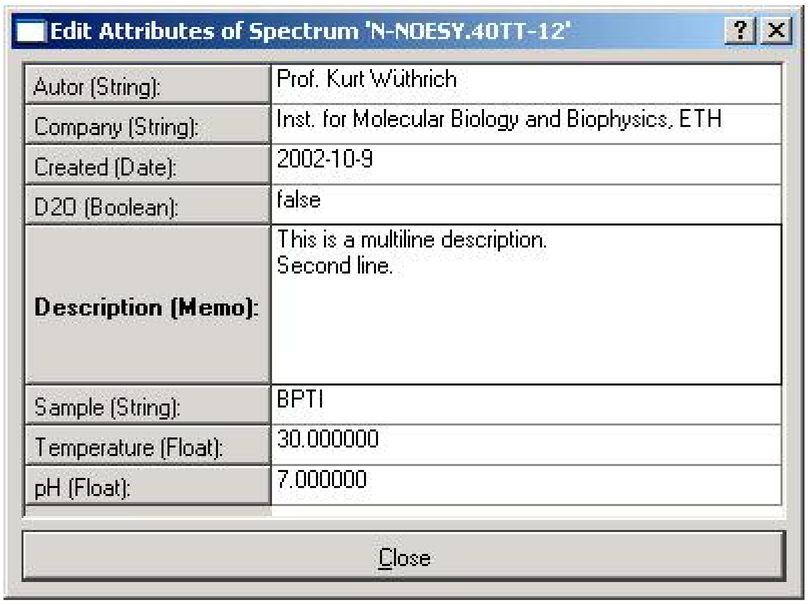
Name: D2O

Boolean

Description: Flag describing whether the solvent of the sample is D2O

Figure 58: Extending CARA object classes with new attributes

As soon as attributes are defined, the user has the option to select an arbitrary object (e.g. a spectrum within the spectrum pane as shown in Figure 35) and execute the menu command *Edit Attributes*. The attributes of most objects are edited using the dialog shown in Figure 59. The repository and project objects are privileged in that their attributes are directly editable from within a pane of the Explorer (see Figure 28). The height of the fields can be adjusted using the mouse.


Edit Attributes of Spectrum 'N-NOESY.40TT-12'

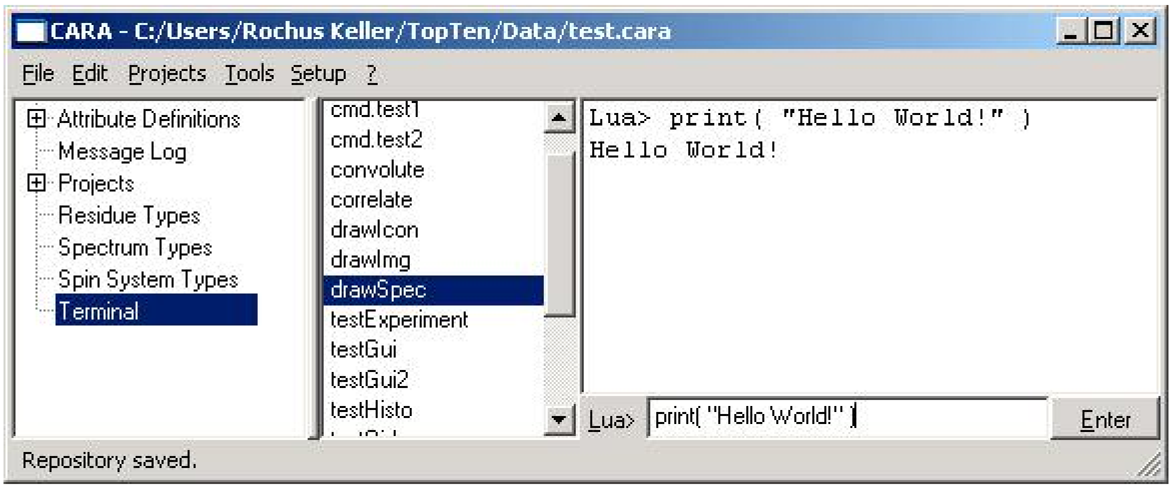
Autor (String):	Prof. Kurt Wüthrich
Company (String):	Inst. for Molecular Biology and Biophysics, ETH
Created (Date):	2002-10-9
D2O (Boolean):	false
Description (Memo):	This is a multiline description. Second line.
Sample (String):	BPTI
Temperature (Float):	30.000000
pH (Float):	7.000000

Figure 59: Dialog for editing dynamic attributes

Dynamic attributes are a nice way for the user to interact with an algorithm implemented by a CARA script (see next chapter). A script is able to read and write the same attributes that the user can edit in the attributes dialog. An algorithm could calculate something and store the results as attributes where the user can inspect them. On the other hand the user could control the parameters of an algorithm by presetting certain attributes. Since the user only can see the attributes declared within the definition pane, an algorithm could easily hide private attributes from the user, but storing them anyway as a part of the repository. The next chapter shows how even structured attributes (i.e. like a record in a database) can be created and accessed by scripting. A repository can be extended this way by custom objects.

6.12 Scripting and Automation

CARA has a built in scripting language. It is useful to implement custom algorithms, data structures, input/output formats or user interfaces. Scripting in CARA is based on the programming language Lua [Ierusalimschy et al. 2003], which is for several reasons regarded by the author as being best suited for this purpose. Technically Lua is the combination of a very small and extremely efficient virtual machine kernel with a lean, easily learnable programming language, which has similarities to the Pascal family of programming languages. The training period to become productive is usually quite short (about one day) for a user even with no computer science background. The execution performance of the resulting scripts is very high [Computer Language Shootout, http://dada.perl.it/shootout/craps.html]. This gives Lua a strong advantage compared to other scripting technologies like Perl, Python or Tcl (which are more widely known instead and used by other NMR software packages). Lua itself is completely application neutral and only equipped with a minimal but robust set of standard library functions. The full benefit of Lua is realized if it is embedded in a host program like CARA.



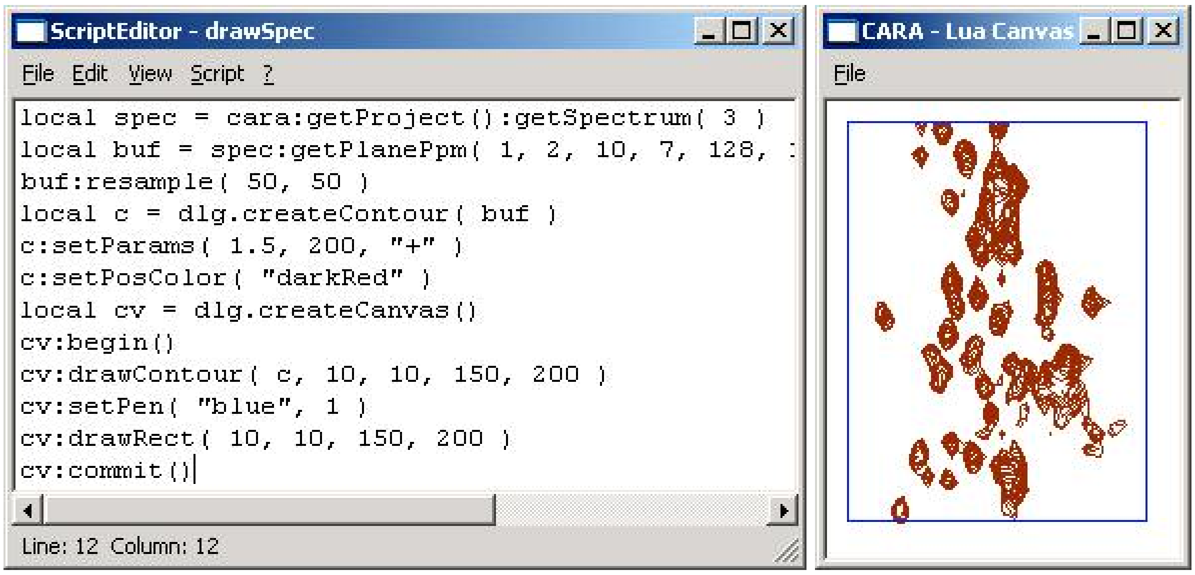
```
Lua> print( "Hello World!" )
Hello World!
```

Figure 60: The terminal pane of the CARA explorer

CARA/Lua consists of a full fledged programming environment with terminal, editors and persistent module management (Figure 60). Most objects managed by CARA can be accessed within the script by means of a large application programming interface (API). The API consists of about fifty object types and five hundred procedures which are documented in a separate programmers manual [Keller 2003].

The execution and development of scripts is straight forward. The user has the possibility to either directly input and execute Lua statements from within the terminal (as shown in Figure 60), or she can create a new named script, which can then be edited using a dedicated scripting editor (Figure 61). The script becomes part of the repository for later reuse. Even the deployment of scripts by means of the template concept (described in chapter 6.2) is possible, so the user doesn't have to write all functions from scratch, but can base its new algorithms on a library loaded from a template.

Let's assume the user has created a new script called *drawSpec* (executing the command *New* from the context menu of the script list in the terminal pane). As soon as a unique name is entered, the script editor opens. The user can then directly start to type the script. When this is done, the menu commands *Script/Check Syntax* and *Script/Execute Script* are executed to check and run the script. Syntax errors are written to the status line of the editor and also to the terminal pane. The script presented in Figure 61 only consists of twelve lines of code. Nearly all lines are calls to the CARA API. The first line opens a spectrum from the repository. Then a plane is cut out of a spectrum and stored in a buffer, before changing its resolution to 50 samples in each dimension. A contour view is then created and configured. Finally a canvas window is opened and everything is painted into it. The right part of the figure shows what happens, if this script is executed.



```
local spec = cara:getProject():getSpectrum( 3 )
local buf = spec:getPlanePpm( 1, 2, 10, 7, 128, :
buf:resample( 50, 50 )
local c = dlg.createContour( buf )
c:setParams( 1.5, 200, "+" )
c:setPosColor( "darkRed" )
local cv = dlg.createCanvas()
cv:begin()
cv:drawContour( c, 10, 10, 150, 200 )
cv:setPen( "blue", 1 )
cv:drawRect( 10, 10, 150, 200 )
cv:commit()
```

Figure 61: A short script to draw a part of a (left) spectrum and its output (right)

Scripting in CARA has many powerful features. As already mentioned there are robust standard libraries (with functions for string handling, mathematics and many other things) and access to all CARA objects. Additionally CARA/Lua offers a comprehensive user interface library to build interactive applications, an object database to create complex, persistent data structures, and a library to easily parse and save XML [Holzner 2001] files.

Algorithm 3: Executing a pathway simulation

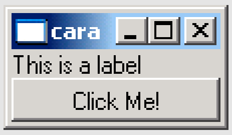
```
local nmr = spec.createExperiment(
  cara:getSpectrumType( "HNCA" ),
  cara:getResidueType( "ALA" ) )
```

```
local path = nmr:getPath( 1 )
for i,j in pairs( path ) do print( j ) end
-- Output:
HN
CA
N
```

Algorithm 3 shows again how to access the CARA API, this time to instantiate an NmrExperiment (see chapter 4.3) and to execute a pathway simulation for an Alanine in a HNCA spectrum. One of the resulting paths is printed using a loop statement.

Algorithm 4: Creating a button window with an event handler

```
local box = gui.createVBox()
gui.createLabel( box, "This is a label" )
local button = gui.createPushButton( box, "Click Me!" )
button:setCallback( gui.event.Clicked,
  function() print( "Button clicked!" ) end )
box:show()
-- Output:
```



In Algorithm 4 a vertical box is created and filled with a label and a button (the resulting window is also shown). The *Clicked* event of the button is then associated with a handler function printing a text to the terminal as soon the button is clicked.

Algorithm 5: Creating and using persistent record objects

```
local rec = cara:createRecord()
print( rec:getId() )
cara:setAttr( "myRecord", rec )
rec:setAttr( 1, "Value 1" )
rec:setAttr( "2", "Value 2" )
print( cara:getAttr( "myRecord" ):getAttr( 2 ) )
```

```
rec:setAttr( "subRecord", cara:createRecord() )
rec:getAttr( "subRecord" ):
  setAttr( "Name", "Rochus Keller" )
print( cara:getAttr( "myRecord" ):
  getAttr( "subRecord" ):getAttr( "Name" ) )
cara:setAttr( "Author", "Rochus Keller" )
print( cara:getAttr( "Author" ) )
print( cara:getAttr( "Creation Date" ) )

-- Output:
5
Value 2
Rochus Keller
```

Algorithm 5 shows how a script can create a persistent record object. It is uniquely identified by an ordinal number automatically assigned by CARA. A reference to this record is then stored in the dynamic attribute "myRecord" of the Repository object by the script. The record can have attributes by itself. The example shows how an attribute can be indexed by either a number or a string. Records can even be nested, i.e. a reference to a record can be stored in an attribute of another record (as done with the attribute "subRecord").

The last chapter shows how a user can present and edit dynamic attributes using the attributes dialog. The last example showes how the algorithm can access the "Author" and "Creation Date" attributes of the repository. If the number attribute "myRecord" was also declared in the repository object type (see Figure 58), the user would see it containing record reference and an ordinal number. She could then change or delete it and thus interact with the algorithm. The user can declare the needed attributes in the definition pane of the Exporer (see Figure 58) and has full control about which attributes are visible in the dialog (see Figure 59) or only accessible by scripts.

7 The CARA Spectrum Format

7.1 Problem

An NMR spectrometer records FID signals in the time domain. The signals originate from a probe head and pass an analog/digital conversion. The converted digital signal is furthermore processed and turned into a frequency domain spectrum, which is then analyzed using dedicated software like CARA. In practice several spectrum file formats have been established, some of them more than twenty years ago. They were mostly developed by scientists as a supposedly neglected side effect of some scientific projects, sometimes without taking into account the respective achievements of computer science. The projects finished, but the spectrum file formats remained to stay.

The following chapters describe a spectrum file format which meets all important requirements; it is efficient and feasible to handle and to integrate into a software package like CARA.

7.2 Requirements and State of the Art

A spectrum file formats has to fulfill a bunch of requirements. Besides the fundamental requirement of being able to carry multidimensional spectrum data, there are also important ergonomic and technical requirements, often neglected by the available formats.

NMR spectra are in fact valuable assets of an enterprise or a scientific group (even if they are not always treated accordingly). Their creation needs a lot of time, money and know-how, and they are the foundation of long going analysis projects and the proof of scientific conclusions. It is therefore worthwhile to put them under configuration control, together with all other relevant project documents. An NMR spectrum thus becomes one of several formal products documenting the results of a project. In a modern scientific environment, all documents should be kept in a document management system (DMS), taking care of access control, version management, traceability and archiving.

All DMS can handle configuration units consisting of a single file. Some can combine more than one files into a compound document, which needs additional configuration effort by the user. To be compatible with all DMS, an NMR spectrum should consist of a single file with minimum size (or at least be compressible).

In earlier times, when no dedicated editors were available, users depended on the possibility to edit spectrum parameter files with a common text editor. That's why many of the available spectrum formats divide a spectrum into different parameter and data files (which complicates spectrum file handling). Some programs even stipulate a naming scheme for the files and the directories containing them. The parameters are normally kept in human readable text files, whereas most data file formats are binary. With today's tools it shouldn't be necessary for a user anymore to manually edit spectrum files. The format should rather impede manual changes, because they hold dangers of corrupting the spectrum. Today it also seems unacceptable to dictate uncommon naming schemes or directory structures to the users.

A spectrum file format should be directly readable by a software in a efficient way, without being forced to initially copy or decode the whole file. The binary format should also be easily interpretable to reduce the risk of programming errors. For historical or implementation reasons of their original host applications, many spectrum formats still use complex compression or data layout schemes (e.g. sub-matrices and separation of mantissa and exponent). On today's computers with their high processing power and large memory sizes, these schemes are not expected to be necessary anymore, even more if the software makes use of modern operating system features like memory mapped files.

A scientific group usually works with different operating systems and processor architectures at the same time. It happens that a spectrum file resides on a server and is used concurrently by a program running on Windows, Linux and Solaris. Therefore the spectrum file formats must be completely platform independent (i.e. indifferent to byte ordering). For performance reasons, the number formats of the sample values must be of constant size over the whole spectrum (i.e. constant block format). One could be tempted to meet these requirements by even coding the sample values in human readable ASCII numbers, but the size overhead would be tremendous.

For the purpose of archiving and internet distribution it is advantageous if spectrum formats are compressible (i.e. the file size is reducible by algorithms like ZIP). Binary formats tend to produce very stochastic (i.e. uncompressible) byte streams (in contrast to text formats), e.g. a 16 bit EASY and a 32 bit Bruker spectrum file can only be compressed by 10% maximally (e.g. using the program WinZIP). The next chapters show, that a compression factor of at least 50% is achievable (without using lossy coding techniques).

Finally the spectrum file format should - even uncompressed - not take up more storage space than needed for the information it contains. Many formats use 32 bit numbers (e.g. Bruker), although the common NMR spectrometers produce numbers with 12 to 18 bits of information only. It is thus difficult to comprehend, why the EASY spectrum format does a compression of 32 bit to 16 bit numbers. The next chapters show, that uncompressed 16 bit numbers are adequate for most cases, and compression is only necessary if 8 bit numbers are used.

7.3 Information Flow

Figure 62 shows a conceptual model of the signal flow from the probe head of the NMR spectrometer to the spectrum file. The RF signal from the probe head as first to be demodulated and amplified. Due to the Nyquist theorem all frequencies above $f_s/2$ have to be removed (where f_s is the sampling frequency of the analog/digital converter) by a low-pass filter to avoid (uncontrolled) aliasing. Most NMR spectrometers make use of oversampling, so the slope of low-pass filter doesn't have to be very steep, which reduces phase distortions and self-oscillation of the filter. Due to oversampling, the digitized signal has a much higher sampling rate than required. The signal is thus reconstructed by a digital FIR filter (which doesn't suffer from the restrictions of an analog filter) and then resampled according to $f_s/2$. The filter can be tuned by the spectroscopist to control the amount of aliased frequencies which should be visible in the spectrum. After reconstruction the digital time domain signals are further processed and converted to the frequency domain (as described in chapter 2.2). The spectrum is then coded (sometimes using data compression techniques) to conform the rules of the spectrum file format.

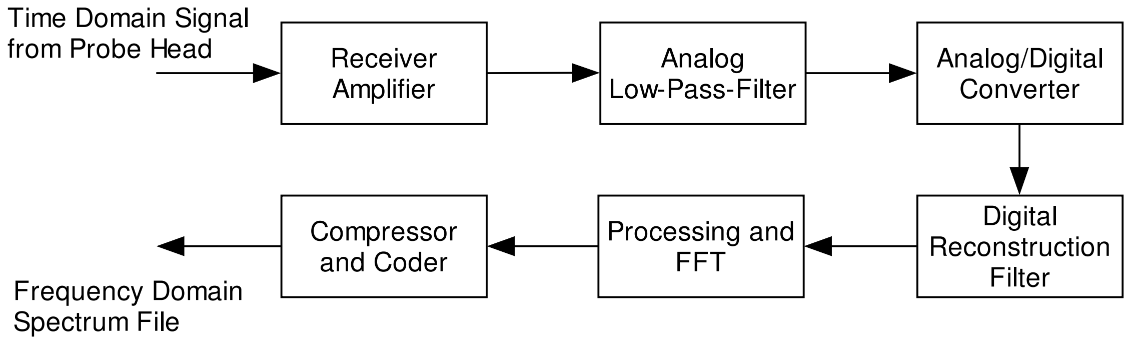


Figure 62: Conceptual signal flow from probe head to spectrum file

All NMR spectrometers known to the author make use of a 16 bit analog/digital converter. The resulting digitized signal thus has a maximum information content of 16 bits. This amount is further reduced by three effects:

1. The receiver amplifier has a limited signal to noise ratio, which limits the useable amplitude range already within the analog part of the signal flow.
2. The quantization of the analog signal introduces a quantization noise inversely proportional to the resolution.
3. In practice only a part of the available amplitude range of the converter is used. The spectroscopists tend to leave enough headroom to avoid clipping. The more conservative the amplifier gain is set, the more bits of the converter are wasted and the higher is the amount of quantization noise.

So the effective information content of the digitized signal is clearly lower than 16 bits in practice. There are several ways to improve this. Oversampling can increase the information content by a few bits. Another way would be the application of noise-shaping techniques to reduce the quantization noise (not done in current NMR spectrometers). But these are only slight improvements, so the signal to noise ratio and the resolution of the analog/digital converter are still of higher significance. This means that even if the calculations of the following processing steps are done with 32 bit floating point precision, the effective information content will remain around 16 bits. There is obviously no need to store sample values with a 32 bits precision in the spectrum file.

As shown in the next chapter the precision of the spectrum files can even be reduced to a range of 6 to 9 bits per sample in practice without losing important information. Additional reduction is possible if a compression scheme is applied. Data compression is either lossy or loss-less. A loss-less compression scheme could for example make a statistical analysis of the digital signal and assign the most frequent amplitude values to the shortest code words (as done e.g. in Huffmann or ZIP compression). This doesn't work well for noise-like data sets. A lossy compression scheme can be more rigorous, because it only allows an approximate and not a precise reconstruction of the original signal. A naive approach could for example simply reduce the number of bits per sample value (e.g. from 16 to 8 bits), which would regularly increase the quantization error over the whole amplitude range. In practice this scheme is optimized by choosing an error distribution taking into account the characteristics of the signal. In XEASY [Bartels et al. 1995] (and also in CARA) a logarithmic compression scheme (and thus error distribution) is used.

In the next chapter the amplitude distribution of typical spectra is studied to find an optimal distribution.

7.4 Amplitude Dynamics of NMR Spectra

NMR spectra tend to have a very similar amplitude distribution, independent of the experiment type or other parameters (e.g. number of samples). Each of the following figures shows histograms of different spectra. All histograms are normalized to y=1.0 (vertical) and show the same amplitude range 0..1'500 along the x axis (horizontal). Only positive amplitudes are shown (A_{max} is the maximum positive amplitude found in each spectrum). The spectra were independently measured by different people as part of different projects using different instruments over two years.

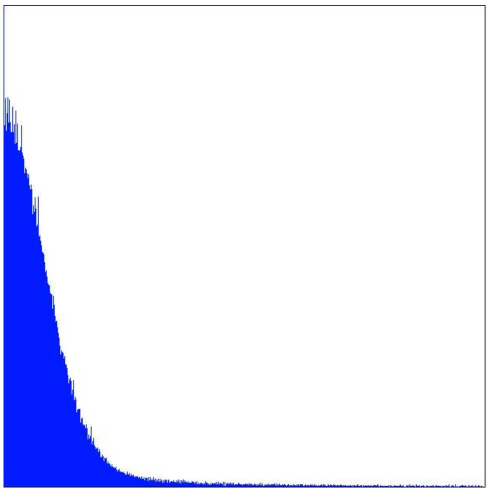
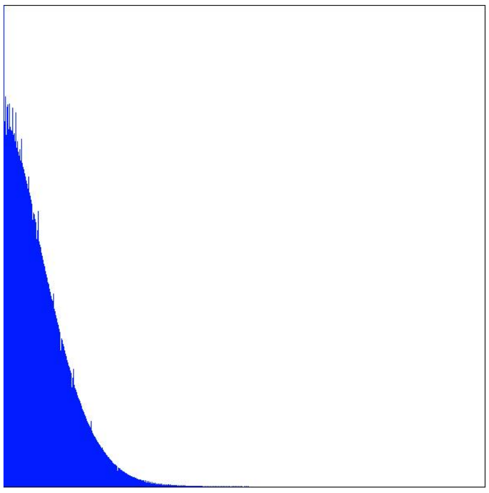



Figure 63: Detail of normalized Histogram A=0..1'500, GroES Trosy-HSQC, A_{max}=281'778 (left), GroES ^{15}N-ed. [^{1}H,^{1}H]-NOESY, A_{max}=1'236'186 (right)

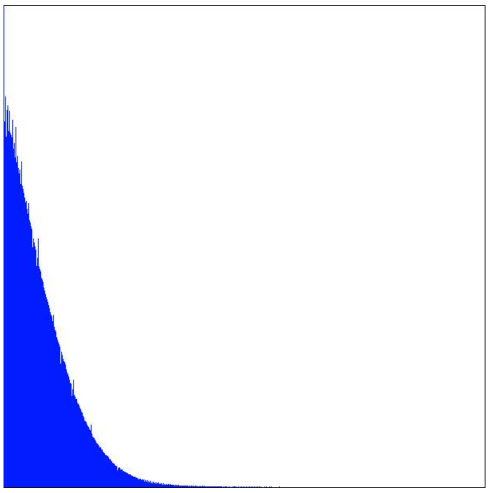
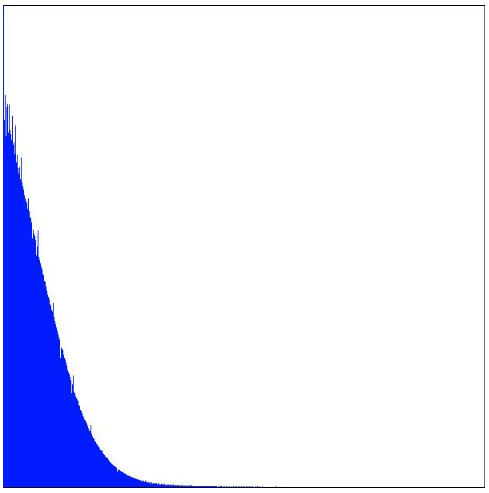



Figure 64: Detail of normalized Histogram A=0..1'500, GroES HNCA, A_{max}=819'611 (left), GroES HNCACB, A_{max} =812'413 (right)

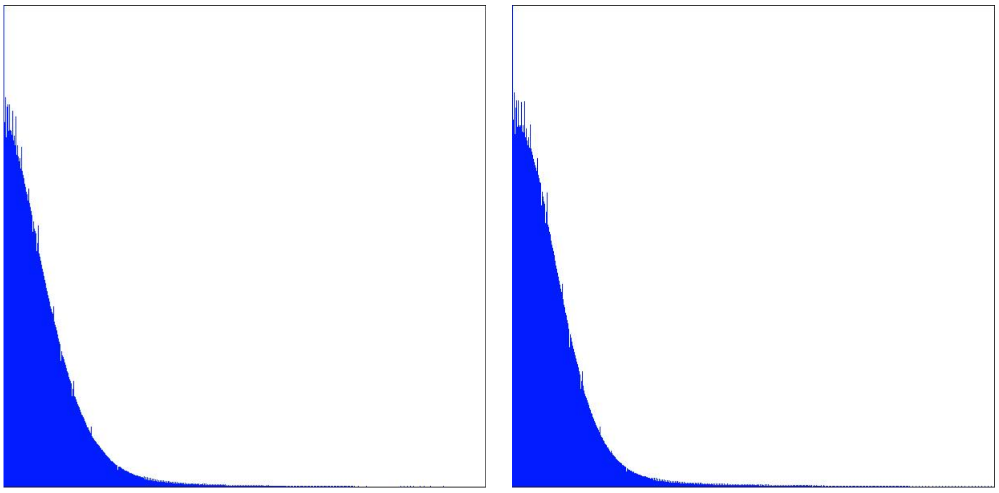


Figure 65: Detail of normalized Histogram A=0..1'500, FimD HNCA, A_{max}=1'511'705 (left), FimD HNCACB, A_{max} =51'998 (right)

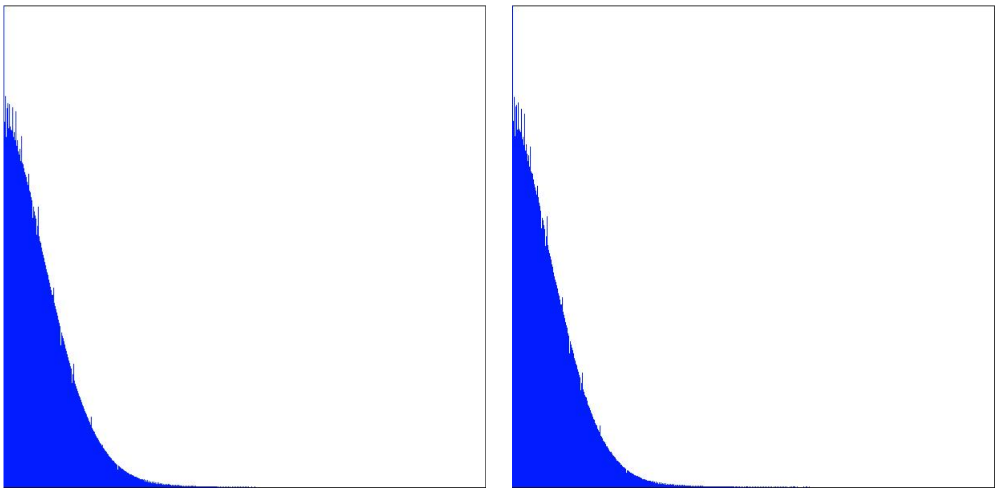


Figure 66: Detail of normalized Histogram A=0..1'500, FimD ^{15}N-ed. [^{1}H,^{1}H]-NOESY, A_{max}=1'614'542 (left), FimD ^{13}Caro-NOESY, A_{max}=1'835'372 (right)

It can immediately be seen that all histograms show very similar gaussian distributions (only the positive part has been shown in the figures). The deviation of all distributions is around 250, even if the maximal amplitudes of the different spectra are completely uncorrelated. A gaussian distribution corresponds to a *white noise*, i.e. a noise with a linear frequency distribution. The noise is probably produced by the receiver amplifier and thus independent of the experiment parameters.

The histograms substantiate that the major part of the spectrum contains noise only. But even the remaining energy is not useful in every case. In HSQC, HNCA and HNCACB experiments the maximum amplitude largely corresponds to the useful amplitude range of the detected signals. In contrast to that in the NOESY experiments the amplitude range of the useful signals is usually only one percent of the maximum amplitude (due to water and diagonal signals), thus wasting 99% of the available amplitude resolution. If such a spectrum was coded using a 16 bit word per sample value (only considering positive values), only the first 9 bits of each word would contain useful information (or even less because at least 2 to 3 bits contain noise only). For these cases it would completely make sense to clip the unused amplitude range and dedicate the available resolution to only useful information.

7.5 CARA Spectrum Format Specification

CARA saves each spectrum in a single file together with all meta information. The format supports different resolutions and coding schemes. Sample values can be stored with either 32, 16 or 8 bit precision. The meta information is divided into elements with fixed and variable sizes (i.e. a text element usually has a variable size). All elements with fixed sizes together constitute the file header, followed by the spectrum data (i.e. intensity matrix) and in the end the elements with variable sizes. All data are stored in a platform independent binary format (using big-endian byte ordering). The intensity matrix is serialized along dimension 1, 2, ... rendering value streams of the form $s_{1,1}$, ... , $s_{m,1}$, $s_{1,2}$, ... , $s_{n,2}$, ... (where $s_{i,j}$ are the sample values and *m*, *n*, ... are the number of samples along the dimensions 1, 2, ...). There is no notion of sub-matrices as e.g. in XEASY.

The following table gives a formal definition of the CARA spectrum file format with all its currently supported variations. The data type *uint* represents an unsigned integer, whereas *float* represents a 32 bit IEEE 754 floating point value (i.e. the native C/C++ float type), both stored with **big-endian byte ordering**. The type *string* is a variable length array of 8bit ANSI characters terminated by a zero byte (value 0).

Table 6: Field list of a CARA spectrum file

Field	*Type*	*#Bytes*	*Description*
cookie	byte array	8	Constant value "caraspec" to recognize the spectrum type
version	uint	4	Constant value "3" (CARA 1.0)
offset1	uint	4	Byte offset to intensity matrix

Field	*Type*	*#Bytes*	*Description*
offset2	uint	4	Byte offset to variable size meta information block (starting with "name")
date1	uint	4	Creation date (number of days since Sep. 14. 1752)
time1	uint	4	Creation time (milliseconds since midnight)
date2	uint	4	Change date (number of days since Sep. 14. 1752)
time2	uint	4	Change time (milliseconds since midnight)
posMax	float	4	Positive amplitude maximum (clipped)
negMax	float	4	Negative amplitude maximum (clipped)
posNoise	float	4	Positive estimated noise level
negNoise	float	4	Negative estimated noise level
posPeak	float	4	Positive amplitude maximum (unclipped)
negPeak	float	4	Negative amplitude maximum (unclipped)
format	byte	1	Code representing the format of the sample values: 0...symmetric 8bit log-compressed 1...adaptive 8bit uncompressed 2...adaptive 16bit uncompressed 3...floating point IEEE 32bit uncompressed 4...symmetric 8bit gauss-compressed 5...adaptive 8bit log-compressed
dimCount	byte	1	Number of dimensions of the spectrum
per dimension			
atom	byte	1	Atom type code (ordinal number according to periodic system): 1...H, 2...He, 3...Li, 4...Be, 5...B, 6...C, 7...N, 8...O, ...
ppm1	float	4	PPM value of sample index 1
ppmN	float	4	PPM value of sample index N
count	uint	4	Sample count N of dimension
fold	byte	1	Code representing the folding along this dimension: 0...unfolded 1...translated (e.g. RSH) 2...mirrored (e.g. TPPI)
MHz	float	4	Spectrometer frequency along the dimension in MHz

Field	*Type*	*#Bytes*	*Description*
data	byte array	*	Intensity matrix (type and #bytes dependent on value of field *format*)
start of variable size block			
name	string	*	Name of spectrum
specType	string	*	Name of spectrum type
desc	string	*	Description of spectrum
author	string	*	Authors of spectrum
com	string	*	Company or institution holding copy right
per dimension			
label	string	*	Name of dimension
dimDesc	string	*	Description of dimension
per attribute			
attrName	string	*	Name of attribute
attrType	string	*	Attribute type (e.g. "String", "Long", etc.)
attrValue	string	*	Value of attribute as string (printf format)

All uncompressed, integer-based values are calculated according to Eq. 15, where p is the original value of the intensity matrix, $v_{\max}$ is the maximum value of the code word (255 for 8 bits and 65'535 for 16 bits), p_{pos} is the positive and p_{neg} the negative maximum amplitude (corresponding to posMax and negMax in Table 6). p is always clipped to the range given by p_{neg} and p_{pos}. The resulting value v is then written to the file.

Eq. 15

$$v = \frac{(p - p_{\text{neg}}) \cdot v_{\max}}{p_{\text{pos}} - p_{\text{neg}}}$$

The coding is either *symmetric* or *adaptive*. The symmetric variation dedicates half of the resolution of the code word to the positive and the other half to the negative amplitude range. The adaptive variation distributes the resolution proportional to the width of the positive and negative amplitude range. Eq. 16 is used to calculate the code value representing the zero amplitude.

Eq. 16

$$v_{\text{zero}} = \left\lfloor 0.5 + v_{\max}\left(1 - \frac{-p_{\text{neg}}}{p_{\text{pos}} - p_{\text{neg}}}\right)\right\rfloor$$

CARA can apply a logarithmic compression scheme, which takes into account the amplitude distribution of a typical NMR spectrum (i.e. lower amplitudes are better resolved than higher ones). The positive and negative amplitude ranges are separately treated. Eq. 17 shows the concept, where $p_{\max}$ either corresponds to p_{pos} or p_{neg} and vpart is either $v_{zero} - 1$ or $v_{max} - v_{zero}$. For $p < 1$ or $p > -1$ the value v becomes 0. The positive amplitude range is mapped to $v = 1..v_{zero} - 1$, whereas the negative amplitude range is mapped to $v = v_{zero}...v_{max}$. Again p is clipped to the range given by p_{neg} and p_{pos}.

Eq. 17

$$v = v_{\text{part}} \frac{\ln(|\,p\,|)}{\ln(|\,p_{\max}\,|)}$$

The gaussian compression scheme tried to further adapt the available resolution to the histogram of a typical spectrum, but didn't render significantly better results than logarithmic compression. Eq. 17 is even more effective and easier to implement than the scheme used by XEASY [Bartels et al. 1995] (which has a larger coding error, less resolution and problems with high amplitude values). A decoder can be implemented very efficiently using a lookup table.

All schemes implemented in CARA allow the user to use other values for p_{neg} and p_{pos} than the ones found in the spectrum, i.e. to clip the stored amplitudes to a certain range. This makes it possible to dedicate the available resolution to the interesting amplitude ranges (e.g. clipping away the water line in a NOESY or the negative side in a HSQC). The use of the available storage and the runtime access to the data become more efficient.

The *adaptive 16 bit uncompressed* scheme is suited to be compressed by loss-less algorithms like ZIP, rendering compression rates of about 50%. The spectrum files thus have the same size like the 8 bit formats, but have to be decompressed before they can be used. The compression rate is possible, because the sample values are stored as integers instead of floating point values, and - due to the typical amplitude distribution - the probability is very high that one byte per sample is zero.

8 Achievements and Contributions

This chapter summarizes the achievements of this PhD project with focus on the personal intellectual contributions of the author (explicitly required by the PhD examiners).

The achievements of this PhD and - at the same time - the intellectual contributions of the author are fully described by this PhD thesis, especially by the chapters 3, 4, 5, 6, 7 and 9, as well as appendices B, C and D.

The conceptual model introduced in chapter 4 is a direct result of the analysis, prototyping and evaluation cycles conducted by the author. It partly improves on ideas by others, but mostly introduces genuine ideas by the author.

Especially the following are genuine ideas by the author (a detailed description can be found in chapter 4):

1. Replacement of the concept of peaks (predominant for the last twenty years) by the more flexible concept of *Spins*, *SpinAlias*, *SpinLabels* and *SpinSystems*. Use of smart *SpinLabels* to tag *Spins* projected from potentially neighbouring SpinSystems.
2. More efficient way of spectrum folding and folded *Spins* representation.
3. Modeling molecules using generic meta classes *ResidueType*, *Atom* and *AtomGroup*, thus not predetermining the model to protein applications.
4. Representation of pseudo atoms by *AtomGroups*, which can be used as generic substitute for *Atoms*, before stereo-specific assignment is known.
5. Directly associating statistic information with *Atom* and *Residue*. The statistics is not predetermined to ^{13}C. *Residue* level statistics override *Atom* level statistics.
6. Categorization of *SpinSystems* and *ResidueTypes* by *SystemTypes*, thus allowing sequence mapping using *SystemType* correspondence (useful for homonuclear assignment described in [Wüthrich 1986]).
7. New variations of algorithms for strip matching and sequence mapping. Modeling fragments by explicit chains of *SpinSystems*.

8. Modeling sequence-specific assignment by connecting the *SpinSystems* to *Residues* and implicitly the *SpinLabels* to the *Atoms* of the *ResidueType* by string matching.
9. Modeling NOE constraints using *SpinLinks* and *LinkAliasses*, as an extension to *Spins* and *SpinSystems* (thus replacing peak lists for constraint management).
10. Modeling NMR experiments as either a sequence of generic selection operators, or as sets of expected labels per dimension. Meta tagging NMR spectra with a *SpectrumType* (without predetermination of supported spectra).
11. Magnetization transfer pathway simulation and peak inference.
12. Specification of typical NMR experiments using the new *SpectrumType* formalism.
13. Robust, very efficient volume determination algorithm using a linear equation system and a uniform peak model (independent of spectrum resolution). Quality assessment by calculating the difference of real spectrum and a back-calculated spectrum.

These ideas were a necessary precondition to satisfy the requirements and recommendations formulated in chapters 1.1 and 3.7, and thus to achieve a scientific progress.

Other algorithms have been described to match strips to fragments [Bartels et al. 1995], to map fragments to the amino acid sequence [Güntert et al. 2000], or to determine peak volume [Glaser, R., www.molebio.uni-jena.de/~rwg/ spscan/]. The strip matching algorithm introduced in this thesis differs from the others by use of smart spin labels, fuzzy comparison operators (based on simple triangle functions) and weighted fitness values. The fragment mapping algorithm introduced in this thesis differs from others by use of fuzzy operators (based on simple triangle functions), weighted fitness values and integral use of generic statistic values associated with freely designable molecule types (thus not restricted to ^{13}C or amino acids). The volume determination algorithm introduced in this thesis differs from others by not directly depending on spectrum quality or resolution (thus avoiding problematic intensity decomposition). The volume is calculated as a superposition of uniform, mathematical peak models instead. The real spectrum is only used in a second step for quality assessment by comparing with the back-calculated spectrum.

The conceptual model has been developed in the context of protein liquid-state NMR. Recent applications in other groups have shown that the model (and the prototype) is generalizable enough to also being reused for other NMR experiments originally not analyzed by the author, or other application domains (like solid-state NMR) or molecule types (like nucleic acids or combinations of amino and nucleic acid chains).

The model has been evaluated and refined along a series of prototype implementations. All prototypes have been implemented as a proof of concept (and as such a side effect of the PhD) by the author himself. The following list gives an overview of the prototypes and implementation years:

1. Spectroscope I: 2000
2. Spectroscope II: 2001
3. SpinBase: 2001
4. Spectroscope III: 2001
5. Backbone: 2001 - 2002
6. Compare: 2001
7. Neasy: 2002
8. Backbone2: 2002
9. Integrator: 2002
10. Sidechain: 2002-2003
11. Slicer: 2002
12. Compiler: 2002
13. Aida/Cara 1.0: 2003
14. Cara > 1.0: since 2004

All prototypes natively run on all major platforms. The implementation details (e.g. software architecture, design and source code) are owned by the author and not part of the PhD thesis.

The specific use case, user interface and usability designs of the prototype windows (as described in chapters 6 and appendix C) also contain many genuine contributions by the author. Window systems, contour plots, strips and slices have already been used in other programs (e.g. XEASY [Bartels et al. 1995] or Sparky [Goddard et al. 2000]). The specific window layout, tailoring of use cases to program functions, simultaneous view of different spectra, and the combination of different view types in the same window, have been optimized by the author as part of this project over several years, and largely affect efficiency, usability and acceptance of the program. Especially the concepts of SynchroScope, PolyScope and SystemScope (chapter 6) are genuine contributions of the author, not yet seen in other programs.

SystemScope became the tool of choice for semi-automatic side-chain assignment in many scientific groups.

Yet another genuine contribution of the author are the NMR spectrum format described in chapter 7, and the repository XML format described in appendix B. Part of the latter has been adopted by the forthcoming application RADAR [Herrmann et al., unpublished].

Science is a joint effort. NMR was completely new to the author when he started his PhD at the institute of Molecular Biology and Biophysics in the group of Prof. Kurt Wüthrich. The group was very helpful and beared with the author when he became acquainted with NMR. Even if the author hasn't been employed with ETH, he has generously been given access to ETH resources and lecture courses, which was indispensable to conduct the PhD project. The group (especially Dr. Fred Damberger, Dr. Peter Güntert and Pascal Bettendorff) additionally supported the project by evaluating the concepts and prototypes developed by the author, and also by providing the author with valuable feedback and comments, allowing him to refine his concepts and to converge the analysis and prototyping activities towards an optimal solution. At a later stage of the project the evaluation was extended to the groups of Prof. Konstantin Pervushin, Prof. Roland Riek and Prof. Beat Meier. These cooperations not only provided the author with additional feedback and new insights into NMR, but also made it possible to generalize the concepts to other working cultures and even new application domains (e.g. solid-state NMR).

A PhD is not only science. It wouldn't have been possible to successfully complete this PhD project without the dedicated and persistent effort of Prof. Nikolaus Amrhein and the Biology Department of ETH.

9 Outlook

CARA so far has proved to be useful, making the process of spectrum analysis, resonance assignment, constraint gathering and volume determination more efficient. Not only a program, but a whole software infrastructure has been created, on top of which new features can be built.

The next big thing will be the replacement of the current, hard coded scope windows by a scope toolkit (a kind of "SuperScope"), in which only the panes or pane layers are hard coded, and then to rebuild the current scopes out of this new toolkit, using Lua as the glue language. Users can then either use these pre-defined scopes or easily build their customized versions. This new concept also allows to automate menus and the like by custom Lua code, or to save a complete application setup as an executable Lua procedure (which then can rebuild the application state when executed).

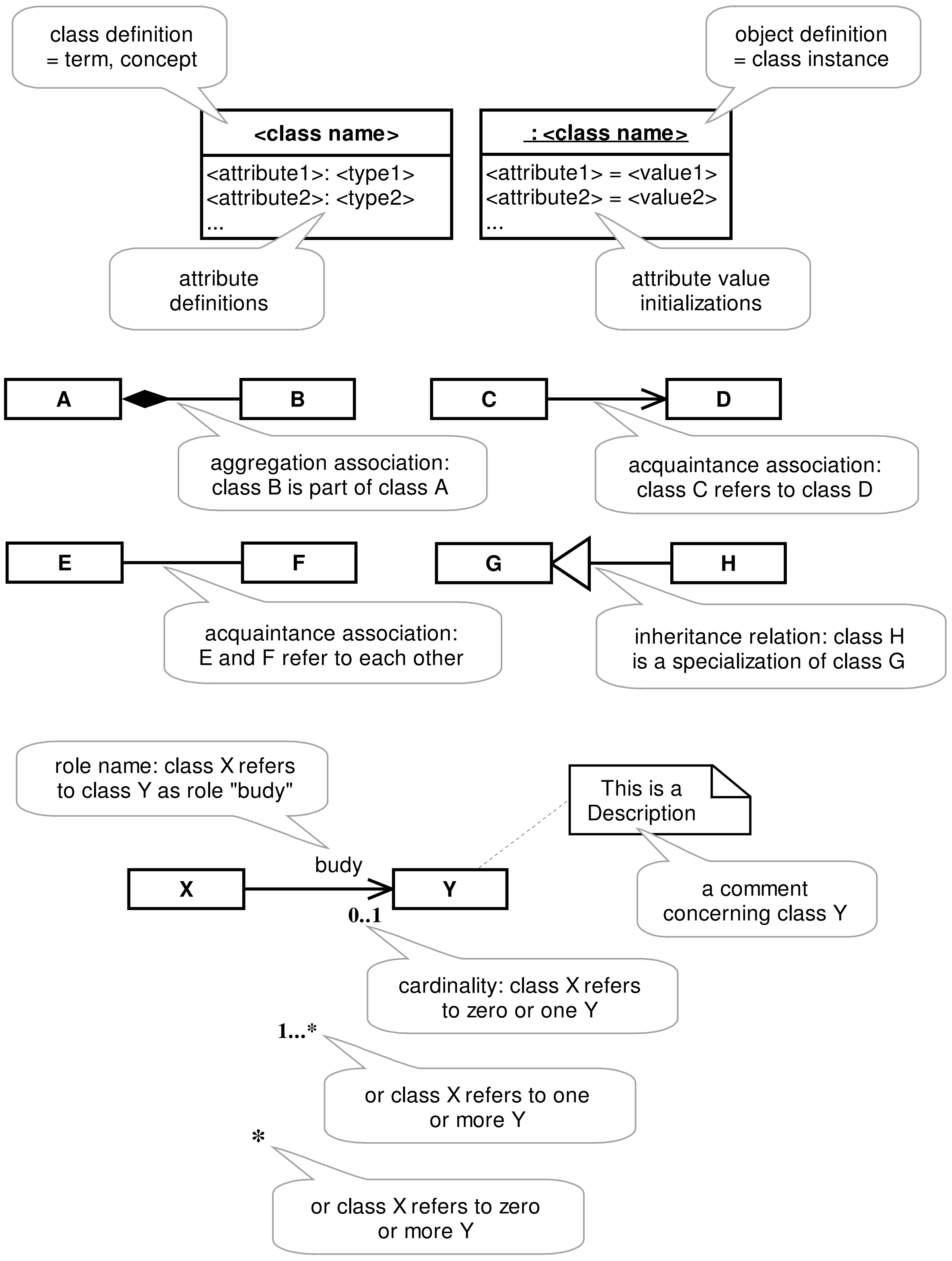
CARA is then ready for further experiments. The author would like to extend the volume determination algorithm of chapter 4.5.1 to be useful as an automatic peak picker (by minimizing the remaining intensity of the difference spectrum). Further algorithms will be developed to automate the backbone assignment. One idea is to calculate all significant fragment variations and validate them against the sequence using carbon shift statistics. Alternatively one could do the same with all combinations of fragments with length three and then apply a threshold accepting or genetic optimization to them.

In the future CARA could establish as an application integration platform, where all kinds of algorithms can be embedded as a plug-in. The scripting and XML features of CARA could then be used to build graphical user interfaces for these algorithms and to control them using asynchronous inter-process communications on the base of XML web service technology. The core algorithms of PROSA [Güntert et al. 1992], DYANA [Güntert et al. 1997] or RADAR [Herrmann et al., unpublished] could then be either published as a shared library or a web service by their authors, and would no longer depend on a command line interface. Nevertheless the development of the different components could still be independent, allowing useful cooperations with even other algorithm's authors, if they adapt their programs to be compatible to this technology.

This integration would enable a tight coupling of all processes, so one could think of not only mapping fragments to the sequence, but already in this early step start calculating structural conformers and checking for their convergence. If new measuring equipment and new experiment types become available, one could think of semi-automatic algorithms directly looking for NOESY constraints without first assigning all individual resonances and then immediately calculating structures. These could be looped back to validate the NOESY analysis.

Another interesting thing would be the introduction of fuzzy peaks to replace the current concept of PPM position as scalar numbers. Each peak picking up to now is a "hard decision" out of a noisy fact base, inherently leading to a loss of information, which has later to be compensated for by use of tolerance ranges and agreement functions. If the whole process chain would make use of fuzzy peaks straight through, the convergence of the minimization algorithms could be enhanced, or the optimization could already include the assignment process (as mentioned before to start structure calculation before the assignment is completed).

Appendix A: A Brief UML Reference


class definition = term, concept
object definition = class instance
<class name>
<attribute1>: <type1>
<attribute2>: <type2>
...
: <class name>
<attribute1> = <value1>
<attribute2> = <value2>
...
attribute definitions
attribute value initializations
A
B
C
D
aggregation association: class B is part of class A
acquaintance association: class C refers to class D
E
F
G
H
acquaintance association: E and F refer to each other
inheritance relation: class H is a specialization of class G
role name: class X refers to class Y as role "budy"
This is a Description
budy
X
Y
0..1
a comment concerning class Y
cardinality: class X refers to zero or one Y
1...*
or class X refers to one or more Y
*
or class X refers to zero or more Y

Appendix B: CARA Repository Format

CARA makes use of the XML standard [Holzner 2001] to define its repository file format (and also some other formats or the XML objects accessible by Lua). The capability of XML to handle hierarchic structures allows a direct mapping of the classes introduced in chapter 4 on XML elements. The names of the elements and attributes are chosen in favor of an optimal space/readability trade-off.

An XML based storage format has several advantages compared to a proprietary binary or text format. Binary formats do not allow the user to see and optionally change the repository files, as is the case with e.g. XEASY default files or Felix project files. If these files get corrupted somehow, they usually can no longer be opened by the original programs and there is no way to repair them. XML files on the other hand are easier to handle than proprietary text formats, because there are many XML parsers available as free software libraries. Other programs thus can easily reuse CARA repository files (i.e. read and write them).

This chapter defines the repository format used by CARA. All elements and attributes are specified using a syntax similar to XML schema (which it is understandable by readers with XML experience).

Table 7: Repository format specification (Version 26)

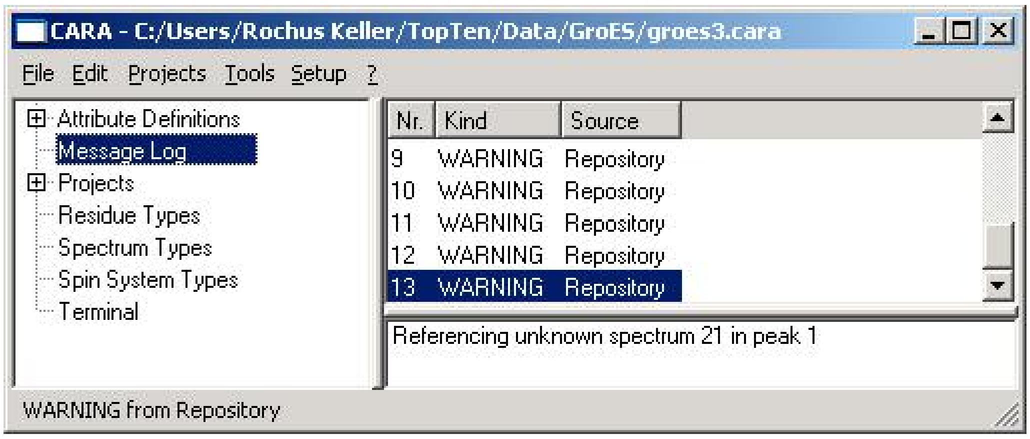
```
elem name="repository"
| attr name="version" type="uint"
| attr name="creator" type="string"
| attr name="modified" type="string"
| attr name="author" type="string"
| attr name="template-by" type="string"
| elem max="*" name="odef"    -- object definition
| | attr name="class" type="string"
| | elem max="*" name="fld"   -- field definition
| | | attr name="name" type="string"
| | | attr name="type" type="typeCode"
| | | attr name="desc" type="string"
| elem name="library"
| | attr name="nterm" type="atomTag" -- n-terminus
| | attr name="cterm" type="atomTag" -- c-terminus
| | attr name="gensys" type="string" -- residue type
| | elem max="*" name="spectrum-type"
| | | attr name="name" type="string"
| | | elem max="*" name="dim"
| | | | attr name="name" type="string"
| | | | attr name="atom" type="atomType"
| | | | attr name="width" type="ppm"  -- peak width or 0
| | | | elem max="*" name="label"
| | | | | attr name="tag" type="atomTag"
| | | | | attr name="off" type="int"
| | | | | attr name="final" type="labelState"
| | | elem max="*" name="step"        -- procedure step
| | | | attr name="text" type="string"
| | | | attr name="atom" type="atomType"
| | | | attr name="dim" type="int"
| | | | attr name="hops" type="int"
| | | | attr name="rep" type="bool"   -- TOCSY repeat
| | | | attr name="mean" type="ppm" optional
| | | | attr name="dev" type="ppm" optional
| | elem max="*" name="systype"   -- spin system type
| | | attr name="id" type="uint"
| | | attr name="name" type="string"
| | | attr name="mdl" type="string"  -- residue type
| | elem max="*" name="residue-type"
| | | attr name="name" type="string"
| | | attr name="short" type="string"  -- primary key
| | | attr name="letter" type="string"
| | | attr name="systype" type="uint"  -- spin system type
| | | elem min="0" max="*" name="group"
| | | | attr name="name" type="atomTag"
| | | | attr name="x" type="uint"
| | | | attr name="y" type="uint"
| | | elem max="*" name="atom"
| | | | attr name="name" type="atomTag"
| | | | attr name="type" type="atomType"
| | | | attr name="num" type="uint"  -- magnitude
| | | | attr name="mean" type="ppm" optional
| | | | attr name="dev" type="ppm" optional
| | | | attr name="x" type="uint"
| | | | attr name="y" type="uint"
| | | | attr name="group" type="atomTag" optional
| | | elem max="*" name="bond"
| | | | attr name="from" type="atomTag"
| | | | attr name="to" type="atomTag"
| elem name="project"
| | attr name="name" type="string"
| | attr name="next" type="uint"  -- next spectrum id
| | elem max="*" name="width"     -- strip width
| | | attr name="atom" type="atomType"
| | | attr name="ppm" type="ppm"
| | elem max="*" name="tol"       -- strip matching tol.
| | | attr name="atom" type="atomType"
| | | attr name="ppm" type="ppm"
| | elem name="sequence"
| | | elem max="*" name="residue"
| | | | attr name="id" type="int"
| | | | attr name="type" type="string"  -- residue type
| | | | elem max="*" name="param"
| | | | | attr name="atom" type="atomType"
| | | | | attr name="mean" type="ppm"
| | | | | attr name="dev" type="ppm"
| | elem max="*" name="spectrum"
| | | attr name="type" type="string"
| | | attr name="id" type="uint"
| | | attr name="name" type="string"
| | | attr name="path" type="string"
| | | elem name="level"
| | | | attr name="pmax" type="float"  -- pos. peak amp.
| | | | attr name="pnoise" type="float"-- pos. noise level
| | | | attr name="nmax" type="float"  -- neg. peak amp.
| | | | attr name="nnoise" type="float"-- neg. noise level
| | | | attr name="thres" type="float"-- contour threshold
| | | elem name="cal"       -- calibration offset
| | | | attr name="dim" type="uint"
| | | | attr name="off" type="ppm"
| | | elem name="map"       -- dimension mapping
| | | | attr name="view" type="uint"
| | | | attr name="spec" type="uint"
| | elem name="spinbase"
| | | elem max="*" name="spinsys"     -- spin system
| | | | attr name="id" type="uint"
| | | | attr name="ass" type="int" optional -- residue
| | | | attr name="systype" type="uint" optional
| | | elem max="*" name="link"        -- fragment
| | | | attr name="pred" type="uint" -- system id
| | | | attr name="succ" type="uint" -- system id
| | | elem max="*" name="spin"
| | | | attr name="id" type="uint"
| | | | attr name="atom" type="atomType"
| | | | attr name="home" type="uint"  -- spectrum id
| | | | attr name="tag" type="atomTag" optional
| | | | attr name="off" type="int" optional
| | | | attr name="final" type="labelState" optional
| | | | attr name="sys" type="uint" optional-- spin system
| | | | elem max="*" name="pos"        -- alias position
| | | | | attr name="spec" type="uint" -- spectrum id
| | | | | attr name="shift" type="ppm"
| | | elem max="*" name="pair"        -- spin link
| | | | attr name="lhs" type="uint"  -- spin id
| | | | attr name="rhs" type="uint"  -- spin id
| | | | elem name="inst"              -- spin link alias
| | | | | attr name="spec" type="uint"  -- spectrum id
| | | | | attr name="amp" type="float"
| | | | | attr name="vol" type="float"
| | | | | attr name="mdl" type="uint"  -- model id
| | elem name="peaklist"
| | | attr name="id" type="uint"
| | | attr name="name" type="string"
| | | attr name="home" type="uint"  -- spectrum id
| | | elem max="*" name="dim"
| | | | attr name="atom" type="atomType"
| | | elem max="*" name="model"
| | | | attr name="id" type="uint"
| | | | attr name="name" type="string"
| | | | attr name="kind" type="modelCode"
| | | | elem max="*" name="param"  -- model dimension
| | | | | attr name="gain" type="float"
| | | | | attr name="bal" type="float"
| | | | | attr name="width" type="ppm"
| | | | | attr name="tol" type="ppm"
| | | elem max="*" name="peak"
| | | | attr name="id" type="uint"
| | | | attr name="home" type="uint"  -- spectrum id
| | | | attr name="tag" type="string"
| | | | attr name="color" type="uint"
| | | | elem name="pos"              -- position alias
| | | | | attr name="spec" type="uint"  -- spectrum id
| | | | | attr name="amp" type="float"
| | | | | attr name="vol" type="float"
| | | | | attr name="mdl" type="uint"  -- model id
| | | | | elem max="*" name="dim"      -- position vector
| | | | | | attr name="pos" type="ppm"
| elem max="*" name="script"
| | attr name="name" type="string"
| | elem name="code"
| | | simpleContent
| | | attr name="lang" type="langCode"
| elem max="*" name="color"          -- color definition
| | attr name="code" type="uint"
| | attr name="name" type="string"
| elem name="database"
| | elem max="*" name="obj"
| | | attr name="oid" type="uint"
| | | elem name="fld"                -- dynamic attribute
| | | | simpleContent
| | | | attr name="name" type="string"
| | | | attr name="type" type="typeCode"
```

The definition of Table 8 can be additionally applied to the following elements: repository, script, residue-type, systype, spectrum-type, project, residue, spectrum, spinsys, spin, pair, peaklist and peak.

Table 8: Generic fld element to store dynamic attributes

```
elem name="fld"
| simpleContent
| attr name="name" type="string"
| attr name="type" type="typeCode"
```

A repository file is validated by CARA when loaded. Any violation of the given specification is reported to the user, i.e. only valid repositories are accepted. CARA also does extensive semantic testing, e.g. check for uniqueness of id numbers, range and format conformance of values, existence of referenced objects, availability of resources, and the like. For some cases the repository is accepted even if there were minor semantic violations. A message is displayed in the status bar in these cases and the user can look up the warning in the *Message Log* category of the Explorer (see Figure 67).



```
Nr. Kind     Source
9   WARNING  Repository
10  WARNING  Repository
11  WARNING  Repository
12  WARNING  Repository
13  WARNING  Repository

Referencing unknown spectrum 21 in peak 1

WARNING from Repository
```

Figure 67: A warning from repository validation

Appendix C: CARA Usability Concepts

Design Qualities

The usability design of CARA meets the following qualities:

1. CARA is optimized for both occasional and regular users. Nearly all functions can be accessed either from descriptive menus (context menus or menubar) or by shortcuts (e.g. CTRL-S to save, CTRL drag to scroll, etc.) or commands (e.g. PP to pick a peak). Menu items which are not available in a certain context are disabled (i.e. not selectable by the user). Names and gestures are standardized and applicable the same way in all windows.
2. CARA tries to never waste screen space. Whatever can be easily implied from context is not explicitly printed all-over the windows (e.g. PPM scales and legends are mostly implicit, but positions are printed in the status bar when moving the cursor).
3. CARA does automatic screen layout. The user can change the distribution of window space on the fly by dragging split bars.
4. CARA tries to minimize the cost for the user to execute the most frequent functions, e.g. slice windows automatically scale to the maximum amplitude, contour levels can be automatically calculated to optimally fit the zoom area, etc.
5. CARA is mode-less. Some programs put the user interface into certain modes (e.g. by explicitly entering the zoom or move mode in Sparky or XEASY). The usual interactions are then interpreted in a way depending on the current mode (i.e. they don't always render the same behavior). In contrast to that CARA gives immediate access to all functions by means of menu functions or shortcuts without the necessity to take care of mode selection, which is much more efficient for experienced users (and at last we expect all users to be experienced after a brief learning period).

6. CARA subdivides each window into different panes which can independently have keyboard focus. The pane having the focus can be recognized by the purple frame rectangle. Many functions and shortcut target the focus pane (e.g. the cursor shortcuts).
7. CARA supports an incremental undo and redo feature for most functions. If a user has executed a function by mistake or trial, she can re-establish the state before execution by activating the undo feature. The number of undo steps is configurable. Undone functions can again be redone.
8. Multi-dimensional information is simultaneously visible from different perspectives (e.g. the slices at the cursor position are always visible, etc.).
9. CARA reduces the complexity of assignment by taking care of global consistency and interlinking of information, so the user doesn't have to always keep every possible connection in his mind, but can concentrate on a certain clear detail, without loss of general validity.
10. CARA is large, but can be incrementally conquered by the user. After learning a few general principles (e.g. like the zoom and navigation shortcuts), most features are evident or at least ignorable up to the point when the user wants to use them. There is no need to first learn the complete program, and since each major use-case has its own environment window, there is little danger to get lost (as for example in XEASY or even Microsoft Word, where one window incorporates each conceivable function, and an inadvertent shortcut execution can lead to nirvana).

Navigation Gestures

The following gestures and shortcuts are generally applicable in all CARA windows. Some functions are supported by more than one gesture to easy operability.

Function	*Gesture*
Zoom in	Press the CTRL and SHIFT keys, click the left mouse button and draw the rectangle around the zoom region by dragging the mouse.
Zoom in	Press the CTRL and SHIFT keys and double click the left mouse button on the center of the zoom region. A zoom factor of two is applied along all dimensions.

Function	*Gesture*
Zoom in	Press the CTRL and SHIFT key and then either the UP or LEFT cursor key to zoom in along the Y or the X axis.
Zoom to Area	If an overview pane is visible (e.g. like in the lower left part of HomoScope), the user can click in it to center the zoom area around the mouse position. The user can also click and drag another zoom area rectangle.
Zoom out	Press the CTRL, SHIFT and ALT keys and double click the left mouse button on the point from which you want to zoom out.
Zoom out	Press the CTRL and SHIFT key and then either the DOWN or RIGHT cursor key to zoom out along the Y or the X axis.
Scroll	Press the CTRL key, click the left mouse button on the starting position and drag the mouse to the end position
Scroll	Press the CTRL key and then one of the cursor keys (LEFT, RIGHT, UP DOWN) to move the spectrum 20 points per key press in that direction. If you press ALT at the same time, the spectrum moves by only one point.
Page	Press the PAGE UP and PAGE DOWN keys to move the view up or down by 75% of the visible height. If you press CTRL at the same time, PAGE UP moves the view to the left and PAGE DOWN to the right.
Move Cursor	Position the mouse and click the left mouse button. The cursor (i.e. the yellow ruler) is then placed at the mouse position and the PPM coordinates are written to the status line.
Move Cursor	Press one of the cursor keys (LEFT, RIGHT, UP DOWN) to move the cursor one point per key press in the given direction. If you press SHIFT at the same time, the cursor moves by 20 points. The new position is printed to the status line.
Select Peaks	Press the SHIFT key, click the left mouse button and draw the rectangle around the peaks you want to select by dragging the mouse. During the drag the distances are printed to the status line in PPM and Hz. When the mouse is released, the status line lists the selected peaks.

Function	*Gesture*
Select Peak	Press the SHIFT key and click on the peak you want to select using the left mouse button. If more than one peak is located under the mouse position, another one is selected with each further click (the current one is printed to the status line). If the ALT key is pressed instead of the SHIFT key, the cursor is moved without unselecting the peak (doesn't work on all Unix platforms)
Open Popup Menu	Press and release the right mouse button in a pane featuring a context menu (or the left button while pressing command on Macintosh). The context menu is opened at the mouse position. On Windows there is also a special key on the keyboard to open the context menu in the upper left of the pane.
Set Focus	If you click in a pane using the mouse (left or right button), then the keyboard focus is automatically transferred to this pane (indicated by a purple frame around the pane). The focus can also be cyclically changed by pressing the TAB key (optionally pressing SHIFT to reverse direction). The keyboard entries are handled by the focus pane (i.e. the availability of shortcuts is dependent on which pane has the focus)
Change Active Window	On Windows and some Unix variations (e.g. Linux) the top-most window can be cyclically changed by pressing the ALT and TAB keys (optionally pressing SHIFT to reverse direction).

Mnemonic Commands

CARA supports character commands to ease the transition from XEASY. The user can directly type the commands from within most windows. The input is written to the status line of the window. It behaves like a normal command line, i.e. the user can use the backspace key to delete input characters. The space key is used to separate parameters (if more than one is expected). If the command is recognized by CARA, it is written out in plain text together with the expected parameters (e.g. GS for "Goto System [Long] <enter>), or directly executed (e.g. FP for "Forward Plane"). The command "?" prints a list of all commands supported by the window to the message log accessible from the Explorer.

Appendix D: CARA High-Level Architecture

This chapter gives a brief overview of the conceptual high-level architecture of the CARA prototype implementation. The prototype is a proof-of-concept of the conceptual model introduced in chapter 4 and as such a side effect of the project. The actual design and implementation are not part of the PhD. The user interface and operational concepts are described in chapter 6.

As a result of requirements analysis, chapter 3.7 recommends an extensible workbench approach, providing a graphical user interface with state-of-the-art usability features. Such a kind of application tends to be inherently complex. Therefore, the development of CARA followed proven software engineering and architectural concepts, even if it was intended to be just a prototype implementation. In contrast to XEASY, which virtually is one large module, CARA follows a layer and component based architecture with a high degree of modularity, promoting dedicated module responsibility and component reusability. The software architecture of CARA can essentially be seen as three layers integrated with a common object infrastructure.

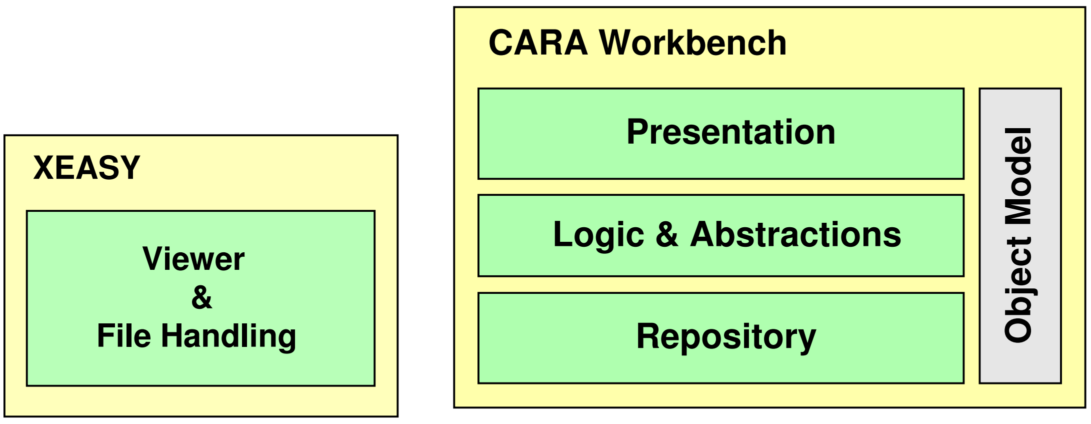


Figure 68: CARA layered architecture compared to XEASY

CARA follows a hierarchic Presentation Abstraction Control (PAC) architecture [Buschmann et al. 1996]. The presentation layer consists of all interactive windows (scopes) and visual components (e.g. for slice, intensity and contour display). The visual components (which are PAC agents actually) have strong resemblance to the ones also found in Geographic Information Systems (GIS), in that multiple visual and interaction layers with a common coordinate system (PPM in this case) are stacked on top of each other.

The analysis objects of the conceptual model specified in chapter 4 are directly mapped to the Logic & Abstractions layer. This layer also contains the undoable transactions for all model updates and the magnetic pathway simulation and peak inference engine. The Repository layer is responsible for the management of the persistent application objects. It also contains all data format handlers and the spectrum engine. A repository contains one or more project partitions and a common library partition reused by all projects (see Figure 69).

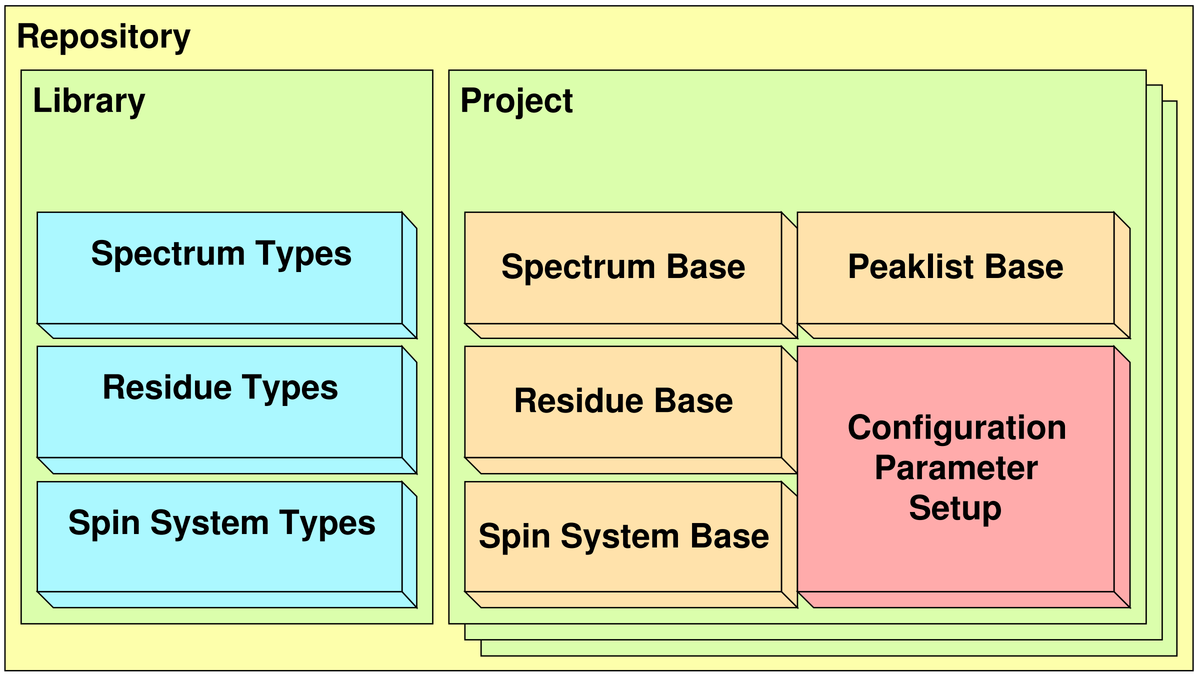


Figure 69: Repository and persistent application objects

Finally CARA builds on top of a common object infrastructure featuring generic services like collections, automatic memory management, command and chain of responsibility patterns, agent infrastructure and message network and the like ([Buschmann et al. 1996] and [Gamma et al. 1995]). This infrastructure also consists of platform abstractions and a generic scripting engine [Ierusalimschy et al. 2003].

During the project, several prototypes have been implemented in successive order. To reduce implementation effort, as much of the functionality as possible has been standardized over the years and integrated into a comprehensive NMR application framework (~150 kLOC of C++ code). Figure 70 gives an overview of the framework as it exists for the CARA version described in chapter 6.

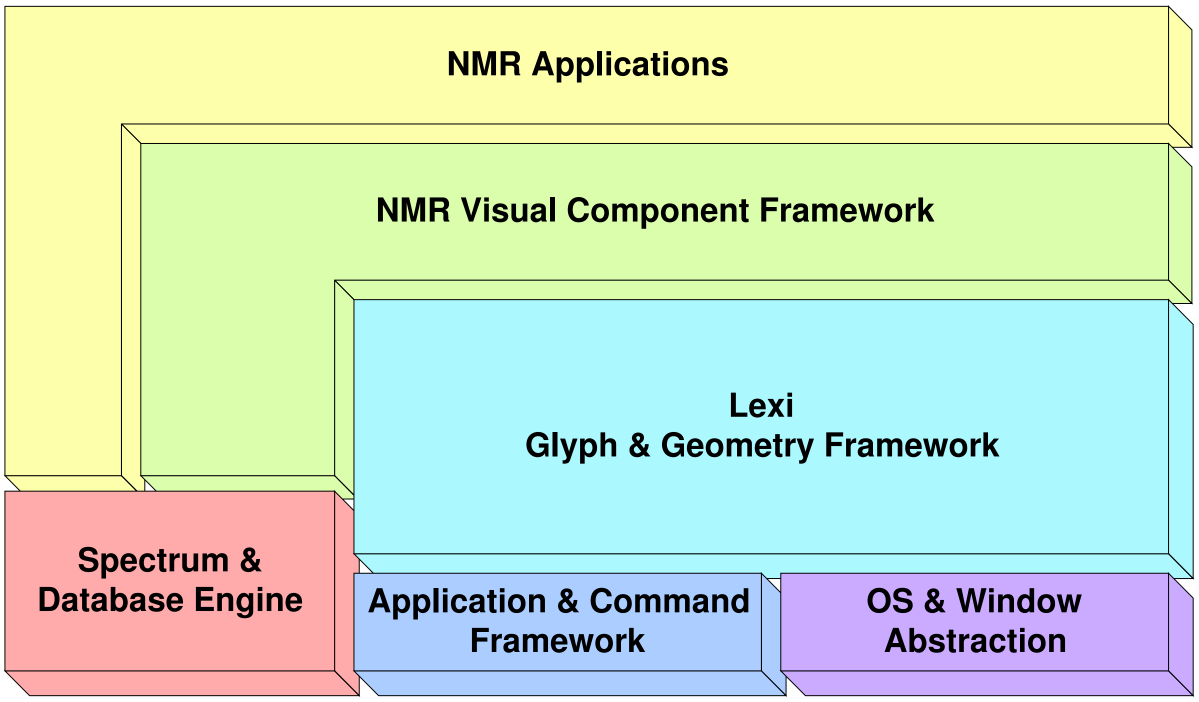

NMR Applications
NMR Visual Component Framework
Lexi
Glyph & Geometry Framework
Spectrum & Database Engine
Application & Command Framework
OS & Window Abstraction

Figure 70: NMR application framework

The framework consists of different reusable modules. The most generic modules are shown in the bottom right of Figure 70. Together they provide the common object infrastructure mentioned before. Lexi is a generic, platform independent user interface framework inspired by [Gamma et al. 1995].

The NMR application framework can be reused for different applications, one of which is CARA (in fact versions of it have been reused for all prototype applications). The layers of the CARA workbench described before are implemented on top of the visual component framework and the data management modules, which take care of the management of spectra and other persistent objects.

Appendix E: References

Abragam, A. (1983). Principles of Nuclear Magnetism. Oxford University Press.

Aue, W.P., Bartholdi, E., Ernst, R. (1976). J Chem Phys, 64, 2229-2246

Anil-Kumar, Ernst, R.R., Wuthrich, K. (1980). Biochem Biophys Res Commun, 95, 1-6

Bartels, C., Xia, T., Billeter, M., Güntert, P. & Wüthrich, K. (1995). The Program XEASY for Computer-Supported NMR Spectral Analysis of Biological Macromolecules, J. Biol. NMR 6, 1-10

Bax, A., Clore, G.M., Gronenborn, A. (1990). J. Magn Reson, 88, 425-431

Booch, G., Rumbaugh, J., Jacobson, I. (1999). The Unified Modeling Language Users Guide. Addison-Wesley

Braunschweiler, L., Ernst, R.R. (1983). J Magn Reson, 53, 521-528

Brüngera, A.T., Adamsb, P.D., Clorec, G.M., DeLanod, W.L., Grose, P., Grosse-Kunstlevea, R.W., Jiangf, J., Kuszewskic, J., Nilges, M., Pannuh, N.S., Readi, R.J., Riceb, L.M., Simonsonj, T. and Warrenb, G.L. (1998). Crystallography & NMR System: A New Software Suite for Macromolecular Structure Determination. Acta Cryst. D54, 905-921

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. (1996). Pattern-Oriented Software Architecture. A System of Patterns. John Wiley & Sons Ltd.

Cavanagh, J., Fairbrother, W.J., Palmer, A.g.I. and Skelton, N.J. (1996). Protein NMR Spectroscopy, Principles and Practice. San Diego, Academic Press

Coggins BE, Zhou P., (2003): PACES: Protein sequential assignment by computer-assisted exhaustive search. J Biomol NMR. 26(2):93-11129

Delaglio, F., Grzesiek, S., Vuister, G. W., Zhu, G., Pfeifer, J. and Bax, A. (1995): NMRPipe: a multidimensional spectral processing system based on UNIX pipes. J. Biomol. NMR. 6, 277-293.

Eccles, C., Güntert, P., Billeter, M. & Wüthrich, K. (1991). Efficient Analysis of Protein 2D NMR Spectra using the Software Package EASY. J. Biomol. NMR 1, 111-130

Ellgaard, L., Bettendorff, P., Braun, D., Herrmann, T., Fiorito, F., Jelesarov, I, Güntert, P., Helenius, A. and Wüthrich, K (2002): NMR structures of 36- and 73-residue fragments of the calreticulin P-domain. J. Mol. Biol. 322, 773-784.

Fesik, S.W. & Zuiderweg, E.R.P. (1988). Heteronuclear 3-Dimensional NMR-Spectroscopy - A Strategy For The Simplification Of Homonuclear Two-Dimensional NMR-Spectra. J. Mag. Res. 78, 588-593

Fiaux, J., Bertelsen, E., Horwich, A. and Wüthrich, K (2002): NMR analysis of a 900K GroEL-GroES complex, Nature 418, 207-211.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley.

Geyer, M., Neidig, K. P. and Kalbitzer, H. R. (1995): Automated Peak Integration in Multidimensional NMR Spectra by an Optimized Iterative Segmentation Procedure. J. Magn. Reson. 109, 31–38

Goddard, T. D. & Kneller, D. G. (2000): SPARKY 3, University of California, San Francisco

Gronwald, W. and Kalbitzer, H.R. (2004): Automated structure determination of proteins by NMR spectroscopy. Prog. NMR Spectrosc., 44, 33-96

Grzesiek, S., Bax, A. (1991). Empirical Correlation between Protein Backbone Conformation an Ca and Cb 13C Nuclear Magnetic Resonance Chemical Shifts. J.Am.Chem.Soc. 113, 5491-5492.

Grzesiek, S. & Bax, A. (1992). An Efficient Experiment For Sequential Backbone Assignment Of Medium-Sized Isotopically Enriched Proteins. J. Mag. Res. 99 (1), 201-207

Grzesiek, S., Bax, A. (1992). J Magn Reson, 96, 432-440

Grzesiek, S., Bax, A. (1993). Amino acid type determination in the sequential assignment procedure of uniformly 13C/15N-enriched proteins. J.Biomolecular NMR 3, 185-204

Güntert, P., Dotsch, V., Wider, G. and Wüthrich, K. (1992): Processing of multi-dimensional NMR data with the new software PROSA. J. Biomol. NMR 2, 619-629

Güntert, P., Mumenthaler, C. & Wüthrich, K. (1997). Torsion Angle Dynamics for NMR Structure Calculation with the new Program DYANA. J. Mol. Biol. 273, 283-298

Güntert, P., Salzmann, M., Braun, D. & Wüthrich, K. (2000). Sequence-Specific NMR Assignment of Proteins by Global Fragment Mapping with the Program MAPPER. J. Biomol. NMR 18, 129-137

Helgstrand, M., Kraulis, P., Allard, P. & Härd, T. (2000). Ansig for Windows: An interactive computer program for semiautomatic assignment of NMR spectra, J. Biomol. NMR 18, 329-336

Herrmann, T., Güntert, P. & Wüthrich, K. (2002a). Protein NMR Structure Determination with Automated NOE-Identification in the NOESY Spectra using the new Software ATNOS. J. Biomol. NMR 24, 171-189

Herrmann, T., Güntert, P. & Wüthrich, K. (2002b). Protein NMR Structure Determination with Automated NOE Assignment Using the new Software CANDID and the Torsion Angle Dynamics Algorithm DYANA. J. Mol. Biol. 319, 209-227

Holzner, S. (2001). Inside XML. New Riders Publishing, Indiana

Hyberts SG, Wagner G. (2003): IBIS - a tool for automated sequential assignment of protein spectra from triple resonance experiments. J Biomol NMR. 26(4):335-44

Ierusalimschy, R., Figueiredo, L.H., Celes, W. (2003). Lua 5.0 Reference Manual. PUC-RioInf.MCC14/03

Ikura, M., Kay, L.E., Tschudin, R., Bax, A. (1990). J Magn Reson 86, 204-209

Ikura, M., Kay, L.E. & Bax, A. (1990). A Novel Approach for Sequential Assignment of 1H, 13C and 15N Spectra of Larger Proteins: Heteronuclear Triple-Resonance Three-Dimensional NMR Spectroscopy. Application to Calmodulin. Biochemistry 29, 4659-4667.

Johnson, B.A. and Blevins, R.A. (1994). NMRView: A computer program for the visualization and analysis of NMR data. J. Biomol. NMR, 4, 603-614

Keller, R. (2003). The CARA/Lua Programmers Manual. NMR.014, DATONAL AG

Koradi, R., Billeter, M., Engeli, M., Güntert, P. and Wüthrich, K. (1998). Automated Peak Picking and Peak Integration in Macromolecular NMR Spectra Using AUTOPSY. J. Magn. Reson. 135, 288–297

Kraulis, P.J. (1989): ANSIG: A Program for the Assignment of Protein 1H 2D NMR spectra by Interactive Graphics, J. Magn. Reson. v 24, pp 627-633.

Krezel, A.M., Ulmer, J.S., Wagner, G., and Lazarus, R.A. (2000): Recombinant decorsin: Dynamics of the RGD recognition site. Protein Science, 9, 1428-1438.

Lee, D., Damberger, F., Guihong, P., Horst, R., Güntert, P., Nikonova, L., Leal, W. and Wüthrich, K (2002). NMR structure of the unliganded Bombyx mori pheromone-binding protein at physiological pH. FEBS 322, 314-318

Luginbühl, P. & Wüthrich, K. (2002). Semi-classical Nuclear Spin Relaxation Theory Revised for Use with Biological Macromolecules. Progr. Nucl. Magn. Reson. Spect. 40, 199-247

Lukin, J., Gove, A., Talukdar, S. and Ho, C. (1997): Automated probabilistic method for assigning backbone resonances of (^{13}C,^{15}N)-labeled proteins. J. Biomol. NMR 9, 151-166

Mori, S., Abeygunawardana, C., Johnson, M.O., Zijl, P.C.M. (1995). J Magn Reson B, 108, 94-98

Moseley, HN., Monleon, D., Montelione, GT. (1991): Automatic determination of protein backbone resonance assignments from triple resonance nuclear magnetic resonance data. Methods Enzymol. 2001. 339, 91-108

Moseley, HN. and Montelione, GT. (1999): Automated analysis of NMR assignments and structures for proteins. Cur. Op. Struct. Biol. 9, 635-642

Nilges, M., Macias, M., Donoghue, S. and Oschkinat, H. (1997): Automated NOESY Interpretation with Ambiguous Distance Restraints: The Refined NMR Solution Structure of the Pleckstrin Homology Domain from Beta-Spectrin. J. Mol. Biol. 269, 408-422

Otting, G., Liepinsh, E. & Wüthrich, K.. Protein Hydration in Aqueous Solution. Science 254, 974-980.

Pons, J. and Delsuc, M. (1999): RESCUE: An artifical neural network tool for the NMR spectral assignment of proteins. J. Biomol. NMR 15, 15-26

Rouvray, D.H. (1997). Fuzzy Logic in Chemistry. Academic Press, London

Talluri, S., Wagner, G. (1996). J Magn Reson B, 112, 200-205

Sebesta, R.W. (1999). Concepts of Programming Languages. Addison-Wesley

Stroustrup, B. (1997). The C++ Programming Language. Addison-Wesley

Vuister, G.W., Bax, A. (1992). J Magn Reson, 98, 428-435

Wilson, R.J. (1972). Introduction to Graph Theory. Academic Press, New York

Wittekind, M. & Müller, L. (1993). HNCACB, A High-Sensitivity 3D NMR Experiment To Correlate Amide-Proton And Nitrogen Resonances With The Alpha-Carbon And Beta-Carbon Resonances In Proteins. J. Mag. Res. 101, 201-205

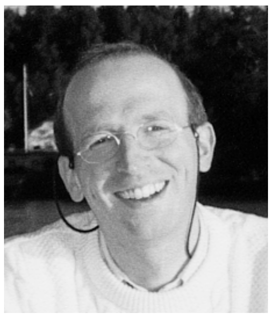
Wüthrich, K. (1986). NMR of Proteins and Nucleic Acids. Wiley, New York.

Wüthrich, K. (1995). NMR - This Other Method for Protein and Nucleic Acid Structure Determination. Acta Cryst. D51, 249-270.

Xu, Y., Wu, J., Gorenstein, D and Braun, W. (1999): Automated 2D NOESY Assignment and Structure Calculation of Crambin(S22/125) with the Self-Correcting Distance Geometry Based NOAH/DIAMOD Programs. J. Mag. Res. 126, 76-85

Zimmerman, D. E., Kulikowski, C. A., Huang, Y. P., Feng, W. Q., Tashiro, M., Shimotakahara, S., Chien, C. Y., Powers, R., Montelione, G. T. (1997): Automated Analysis of Protein NMR Assignments Using Methods from Artificial Intelligence. J. Mol. Biol., 269, 592–610

Appendix F: Curriculum Vitae



Rochus Leonhard Joseph Keller has studied electrical engineering and information technology at the Swiss Federal Institute of Technology (ETH Zürich). He received an academic degree as *Diplom-Ingenieur* in 1991. His master thesis about a "Real-time Expert System for Simulation of the Piano Accompaniment in a Jazz Band" with Prof. H. Baggenstoss and Denis L. Baggi was honored with a best mark. Between 1993 and 2000 he also attended accredited course work in business management at BWI, ETH Zurich, and law at the University of Zurich. In 2000 Rochus started this Ph.D. project at the Institute of Molecular Biology and Biophysics at ETH Zurich with Prof. Dr. Kurt Wüthrich.

Before starting his Ph.D. Rochus has been working as a senior system and technology consultant for international companies. He has many years of experience in software engineering, especially in interdisciplinary analysis, modeling, design and architecture, but also process reengineering, project and quality management, documentation and customer training. In 1995 he co-founded a software engineering company, where he was also responsible for the implementation of an object oriented development process according to SEI/CMM and ISO 9001. Under his direction the company built several large systems, e.g. a real-time simulator of a defense radar intelligence system based on distributed objects for Swiss Government (according to military quality and secrecy standards, 1995-1998).

Rochus also received a musical education in piano performance, improvisation and composition with Melchior Ulrich, Benno Ammann (who was his great-uncle and also a known Swiss composer), John Wolf Brennan and Christoph Stiefel, 1975-1988. He has been working as a performing musician, composer and producer of contemporary music for many years, e.g. in 2004 he composed and co-produced the music for the famous *Loisium* vine museum near Vienna (www.loisium.at).

Rochus can be reached at rkeller@nmr.ch and www.rochus-keller.info.