The Computer Aided Resonance Assignment Tutorial

Rochus L.J. Keller
Institute for Molecular Biology and Biophysics
The Swiss Federal Institute of Technology
Zürich, Switzerland

cantina verlag

The publisher offers discounts on this book when ordered in quantity for special sales. For more information, please contact:

CANTINA Verlag
Centralstrasse 1
CH - 6410 Goldau
Switzerland

Keller, Rochus. 1966 -
The Computer Aided Resonance Assignment Tutorial /
Rochus L.J. Keller
Includes bibliographical references and index.
ISBN 3-85600-112-3 (alk. paper)
1. Nuclear magnetic resonance assignment 2. NMR 3. CARA

ISBN 3-85600-112-3
Text printed on acid-free paper.

First edition, November 2004

Foreword

A major breakthrough in NMR occurred in 1966, when Richard R. Ernst discovered Fourier transformation NMR. This method is not only the fundamental basis of modern NMR spectroscopy, but also intertwined NMR indivisibly with computer technology. One decade later with the extension to two dimensions, computer technology started to constrain advances in biomolecular NMR. First, computer hardware was the major bottleneck requiring the development of NMR experiments, which for example use less disc space. Rapid progress in computer technology then eroded these hardware limitations. In contrast, computer software and its development started to play a central role in three-dimensional structure determination of proteins and nucleic acids by solution NMR. The first generation software established in the 80ies made NMR structure determination possible. With successive improvements thereof and the introduction of triple resonance experiments, structure determination became routine, but far from efficient and the software protocols not suitable to larger systems. Only recently, sophisticated software has been developed for structure calculation (i.e. CNS, CYANA), structure analysis (i.e. MOLMOL) and now with the software CARA also for spectra analysis. The ultimate fully-automated software package is not yet established, but the combination of the aforementioned software packages enables efficient and reliable three-dimensional structure determination of biomolecules. Furthermore, they are continuously updated, improved and extended and hence contain the promise for further automation.

The importance of these software packages is enormous and often underestimated by the scientific community. They not only relieve the structural biologist from much tedious work, but speed up the process of structure determination several times. This results in a gain of several months of man power per structure and concomitantly hundreds of years of man power per year throughout the NMR community. Furthermore, these software packages are opening solution structure determinations to molecular biologists and biochemists.

Roland Riek, Ph.D

Assistant Professor of Structural Biology Laboratory, The Salk Institute

Preface

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. To assign a protein with 100 or more residues, several thousand signals have to be analyzed and identified. 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.

This book is about CARA, which turned out to be an acronym for Computer Aided Resonance Assignment. I developed CARA as part of my Ph.D. under the supervision of Prof. Dr. Kurt Wüthrich at the Institute for Molecular Biology and Biophysics [Keller 2004]. 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 infer information and ensure consistency. It is the result of a four-year iterative improvement process, in the course of which requirements and different assignment strategies have been analyzed, and several prototypes have been constructed and evaluated. CARA is a comprehensive implementation of this model. It tries to be as self-explaining as possible. But NMR is a complex business, and even the new model and usability concepts cannot completely hide all complexity from the user. Former XEASY or Sparky users, who are used to manual work with peak lists, might experience a little cultural shock when they start using CARA, because it demands a new way of thinking about resonance assignment. But our experience shows, that if they have successfully managed to take this step, they immediately become productive and cannot imagine to ever return to the old way. This tutorial wants to help users to take this first and most important step.

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 and Pascal Bettendorff 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 for his great support as my official "Betreuer" and his help in defending my stakes. Finally I thank Drs. Kurt Wüthrich, Beat Meier, Konstantin Pervushin and Roland Riek for the opportunity to cooperate with their groups.

Rochus Keller

November 2004

Table of Contents

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 artificial 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 onto 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].

But this information is not for free. To assign a protein with 100 or more residues, several thousand signals have to be analyzed and identified. It is like 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. At the same time the tools were complicated to handle and their data models allowed management of only parts of the information, thus forcing the user to resort to provisional means sometimes as basic as paper and pencil.

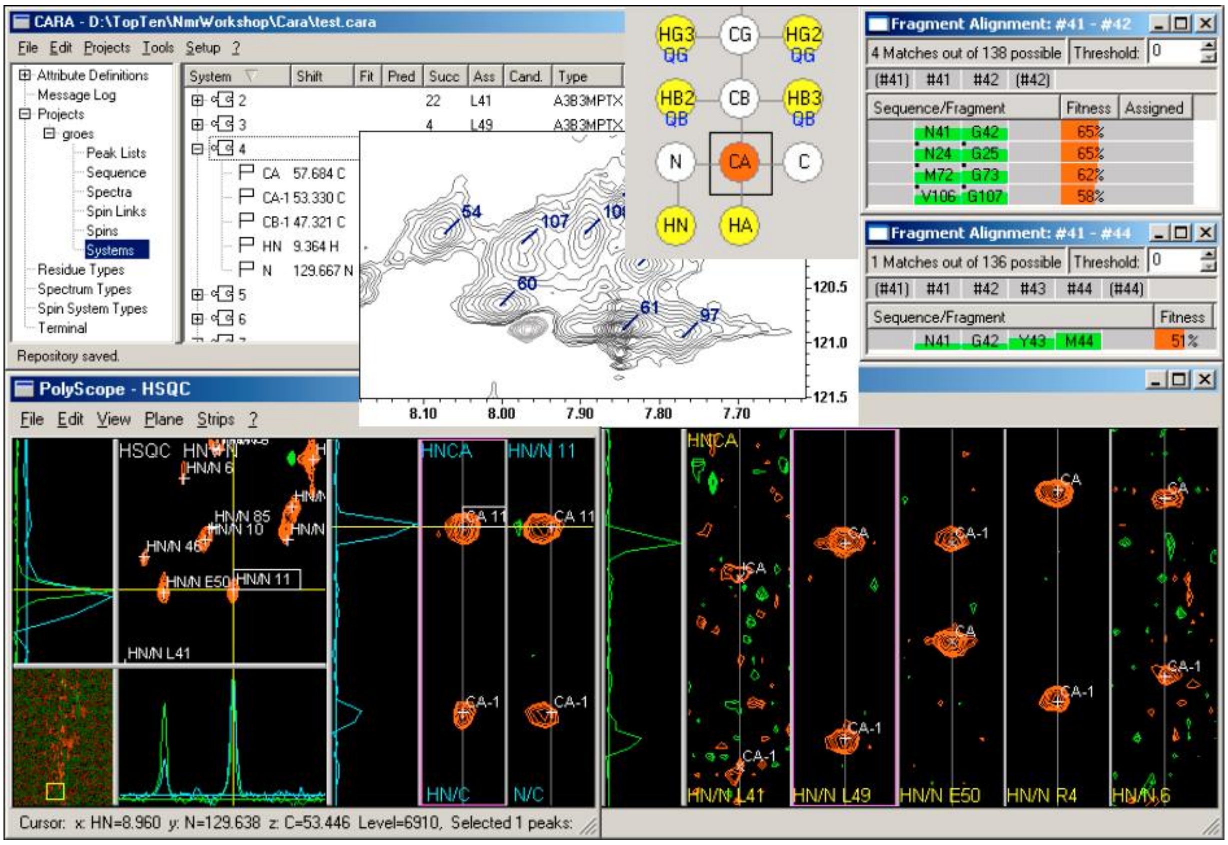
CARA (acronym for *Computer Aided Resonance Assignment*) is a novel application for the analysis of NMR spectra and computer aided resonance assignment. One of the major achievements realized in CARA is its formal and sufficiently complete conceptual model of the NMR resonance assignment application domain. This model is able to describe and capture all information coming up during the analysis and resonance assignment of NMR spectra and to ensure its consistency. The new software package CARA is a comprehensive implementation of this conceptual model. It follows a semi-automatic approach and causes a significant increase of process efficiency and a decrease of error probability. In contrast to other solutions, such as XEASY [Bartels et al., 1995] or NMRView [Johnson et al. 1995], whose information management is primarily based on individual files for peak lists, chemical shift lists etc., CARA makes use of a central repository to manage abstract and semantically interlinked information objects. The availability of this repository, which is stored in XML format on disk, 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 pathways of NMR experiments, to integrate cross-peaks and to back-calculate and efficiently store and access NMR spectra.

An important feature of CARA is its built-in, efficient scripting language [Ierusalimschy et al., 2003], which can be used to implement custom algorithms, data structures, input/output formats or user interfaces. The integrated programming environment offers a terminal window, editors and a persistent (i.e. being part of the repository) module management. Scripts have full access to the rich CARA object model. Each major CARA object can furthermore be extended dynamically by persistent attributes (i.e. named number or string variables), which can be edited through the user interface and accessed by scripts.

The input and output file formats of CARA are compatible with XEASY and the assignment and structure calculation algorithms DYANA [Güntert et al., 1997], CANDID [Herrmann et al., 2002] and CYANA. Additional formats can be implemented using the built-in scripting language. NMR spectra in BRUKER, XEASY, Sparky or NMR Pipe format can be used directly.

CARA features a state-of-the-art graphical user interface with many usability features (like undo/redo) and a look and feel commonly accepted by the Windows users community. CARA runs natively and offers identical features on all supported platforms, i.e. 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). The entire application is written in ANSI C++ adhering to modern software engineering standards and with particular emphasis on being efficient, maintainable, and extensible to future developments.

CARA can be downloaded for free from www.nmr.ch.



Screen views of CARA

2 NMR Structure Determination in a Nutshell

In this chapter we give a brief overview of the process of protein structure determination using NMR technology. Its purpose is to show where and how CARA fits in the process. In the next chapter we will then concentrate on the analysis and assignment of NMR spectra using CARA, which is the main focus of this tutorial.

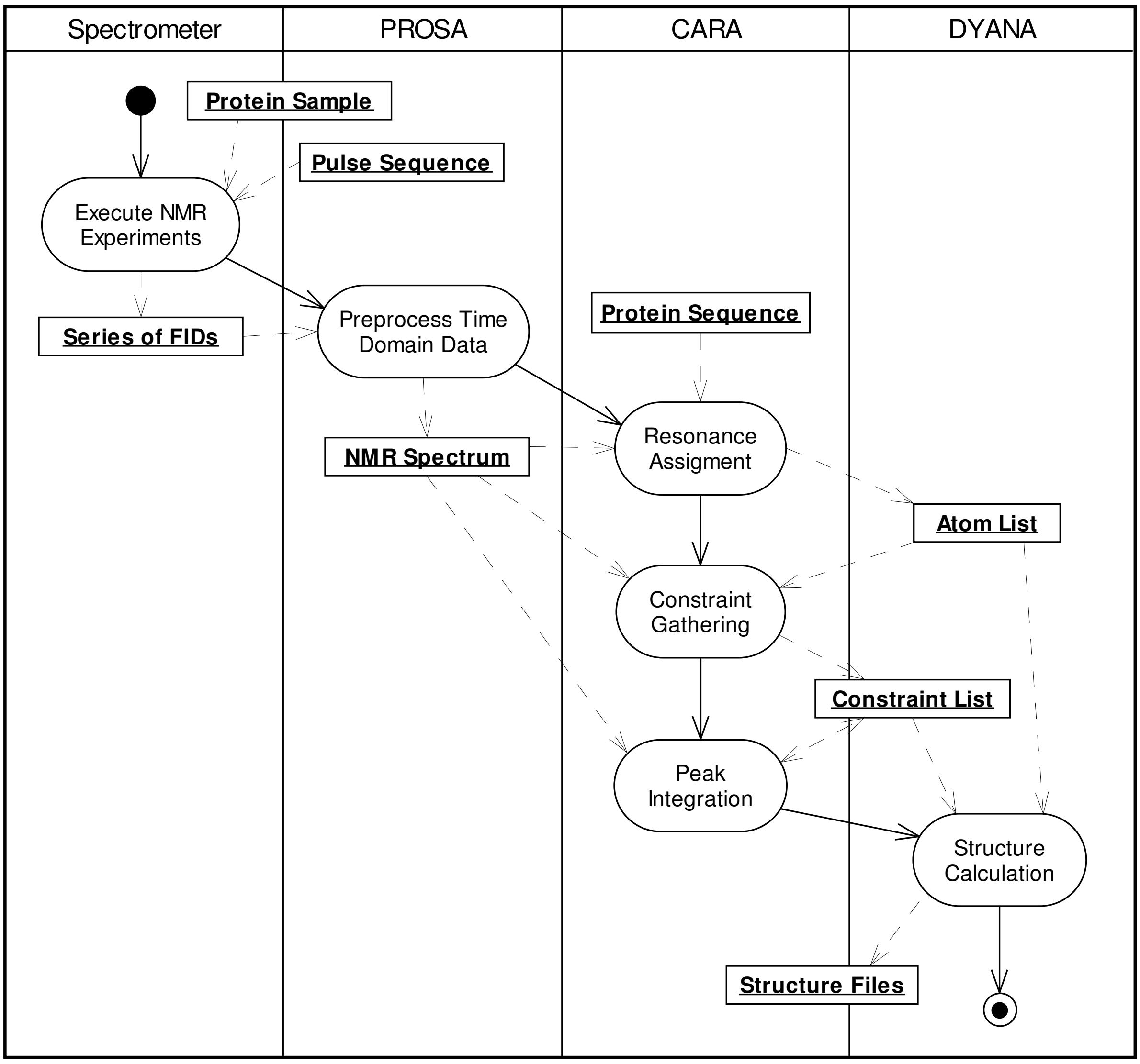


Figure 1: Overview of Structure Determination Process

Figure 1 shows the whole process as an UML 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 is kept in a text file (also called pulse program) and used to setup the spectroscope. 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 of the molecule is controlled. The corresponding physical effects are best described by quantum mechanics (e.g. Bloch equations [Luginbühl et al. 2002]), 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 Preprocessing Time Domain Data

During a multidimensional NMR experiment a signal (i.e. impulse response) depending on several time coordinates is recorded. 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 the decay of the signal is substituted by an algorithm using linear prediction, which extends the digitized time domain data with additional sample points. The additional data helps 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 different reasons. There are several ways 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).

2.3 Resonance Assignment

The goal of resonance assignment is to uniquely associate each NMR sensitive atom 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 gaussian line shape, called *peak*) represents one or more atoms. Given a small molecule and a spectrum with a good resolution (i.e. high number of samples per PPM), there is a good chance that separate frequency lines can be recognized and associated with exactly one atom each (or with a reactive group such as CH_3). With increasing molecule size and constant or decreasing resolution, more and more frequency lines will fall together (or overlap at least). This also happens if parts of the molecules are rotating (such as aromatic rings or methyl groups). We call this effect chemical shift degeneracy. It is also possible for experimental reasons that expected frequency lines are not visible, or that spurious signals appear. All these effects thus make it nearly impossible in practice to uniquely associate each frequency line with an atom.

Over the years several new experiment types have been developed to overcome some of these problems. 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. For example in a 2D $^1H,^1H$-COSY spectrum each peak represents a scalar spin-spin coupling between two hydrogen atoms being apart one to three covalent bonds. Because of this restriction the spectrum only shows correlations between atoms of the same residue. Nearly all intraresidual hydrogen atoms of an amino acid can be correlated as such that they form one connected graph (or up to three disconnected graphs in Histidine, Phenylalanine, Tryptophan and Tyrosine), where each vertex represents an atom (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 atom or reactive group to have a unique chemical shift position in ideal case.

A spin system has a characteristic topology in a 2D ^{1}H,^{1}H-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 of the residues are replaced by deuterium, so that the ^{1}H 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 experiment type is the 2D ^{1}H,^{1}H-NOESY (short form for *nuclear Overhauser spectroscopy*), which shows a signal for each pair of hydrogen atoms with a internuclear distance of less than 5 A. The related atoms can belong to arbitrary residues, according to the structural conformation of the protein. A NOESY experiment can thus be used to sequentially connect the spin systems found in COSY spectra (using the fact that in ideal case an atom is represented by the same chemical shift in both spectra). The same experiment will also be used for constraint gathering 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 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 atoms.
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) 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 the combination of 2D ^{15}N-HSQC and 3D HNCA is widely used. These experiments extend the concept of COSY and additionally show correlations between nuclei of different types (i.e. by scalar through-bond spin-spin couplings). The ^{15}N-HSQC shows one peak for each ^{15}N-^1H pair connected by a single covalent bond. Since the backbone of a protein only contains one ^{15}N, this experiment can be used to directly identify the HN and N chemical shifts of each residue (except for Proline). An HNCA spectrum shows two peaks for each HN-N pair (i.e. for each residue except the Prolines), the stronger one corresponding to the intraresidual CA and the weaker one to the sequentially neighboring CA (i.e. CA-1 in N-terminus direction).

Where in homonuclear assignment the characteristic topology of the spin systems is mainly used to match the fragments onto the sequence, the heteronuclear assignment can make use of the fact, that the CA and CB 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. Identyfy the HN-N pair of all possible residues using a 2D ^{15}N-HSQC spectrum of the protein.

2. For each HN-N pair identify the CA and CA-1 using a 3D HNCA spectrum. The HN-N pairs are used to cut out two-dimensional strips along the C dimension of the HNCA, each showing the CA and CA-1 peak.
3. Find pairs of strips with corresponding CA-1 and CA 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-NOESY, 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-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 synchronously analyzing COSY and TOCSY spectra (and also NOESY if necessary). A 2D ^{1}H-^{1}H TOCSY (short form for *total correlation spectroscopy*) spectrum shows a superset of the signals of a COSY spectrum, i.e. several successive ^{1}H-^{1}H relations over one 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 H atom of the system, and is symmetric with reference to the diagonal axis. This pattern can be used to find all towers belonging to a single spin system. By comparing the COSY and TOCSY spectrum at the position of a tower, 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 HCCH-TOCSY spectra, 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 *atomlist* (containing an atom with its associated chemical shift in each row). CARA implements a formal model of resonance assignment [Keller 2004].

2.4 Constraint Gathering and Peak Integration

As described before a 2D ^{1}H,^{1}H-NOESY shows a cross-peak for each pair of hydrogen atoms with a internuclear distance of less than 5 A. 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 aimed to associate each NMR sensitive atom 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 atoms represented by its chemical shift position. The volume (i.e. integral) of the peak is used to calculate the upper distance limit of the constraint (according to the given formula above). The quality of the structure grows with the number of discovered distance constraints and the accuracy of the peak integration.

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-NOESY spectra, 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 atomlist, the corresponding NOESY peaks can even be automatically found in most cases (as is done in the program ATNOS [Herrmann et al. 2002a]). CARA introduces a new algorithm for automatic peak integration.

The result of the process of constraint gathering and peak integration 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 atoms with an internuclear distance of less than 5 A. The volumes of the corresponding NOESY peaks were used to calculate the upper distance limits of the constraints (which are between 2 and 5 A). With this information a structure can be calculated by folding the primary structure of the protein as such, that its atoms optimally fulfill the distance constraints. It is not possible to directly formulate and solve an equation system due to the complexity and fuzziness of the problem. A solution has to be found by an optimization procedure instead. The program DYANA [Güntert et al. 1997] uses the same constraint set to simultaneously calculate a whole bunch of structures (called *conformers*). Each 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 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.

3.1 Overview

The computer program CARA is a comprehensive implementation of the conceptual model formally introduced in chapter 4 of [Keller 2004]. 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.

CARA is organized around a central *Explorer* window (see Figure 2). 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 using the built in scripting language.

Figure 2 shows the nine standard tool windows (i.e. environments). All of them (besides NEASY) 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.

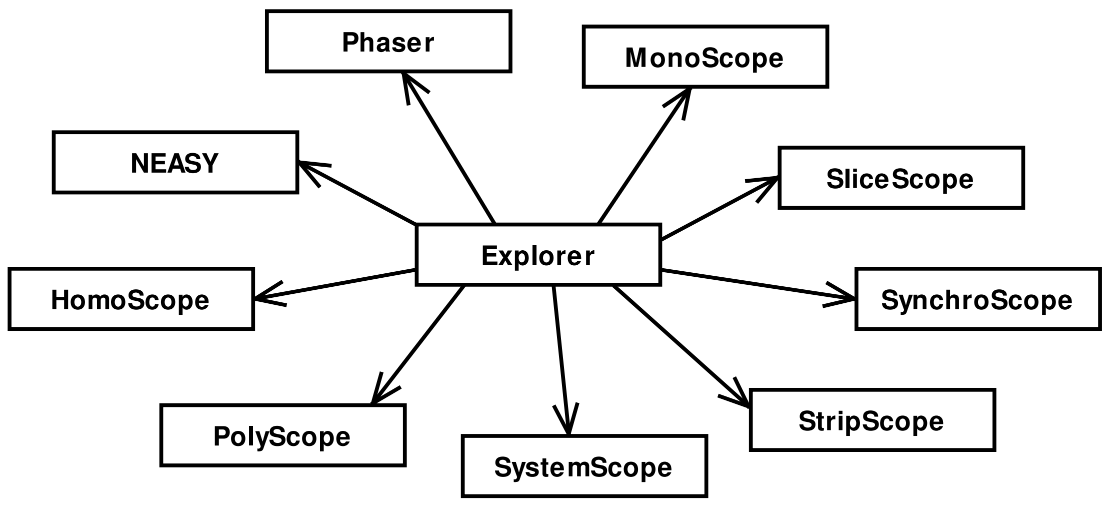


Figure 2: The CARA editors accessible with the Explorer

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 onto 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 13C-HSQC. 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 integration 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 3 gives an overview of the primary objects a *Repository* contains. Most of them were introduced and described in [Keller 2004].

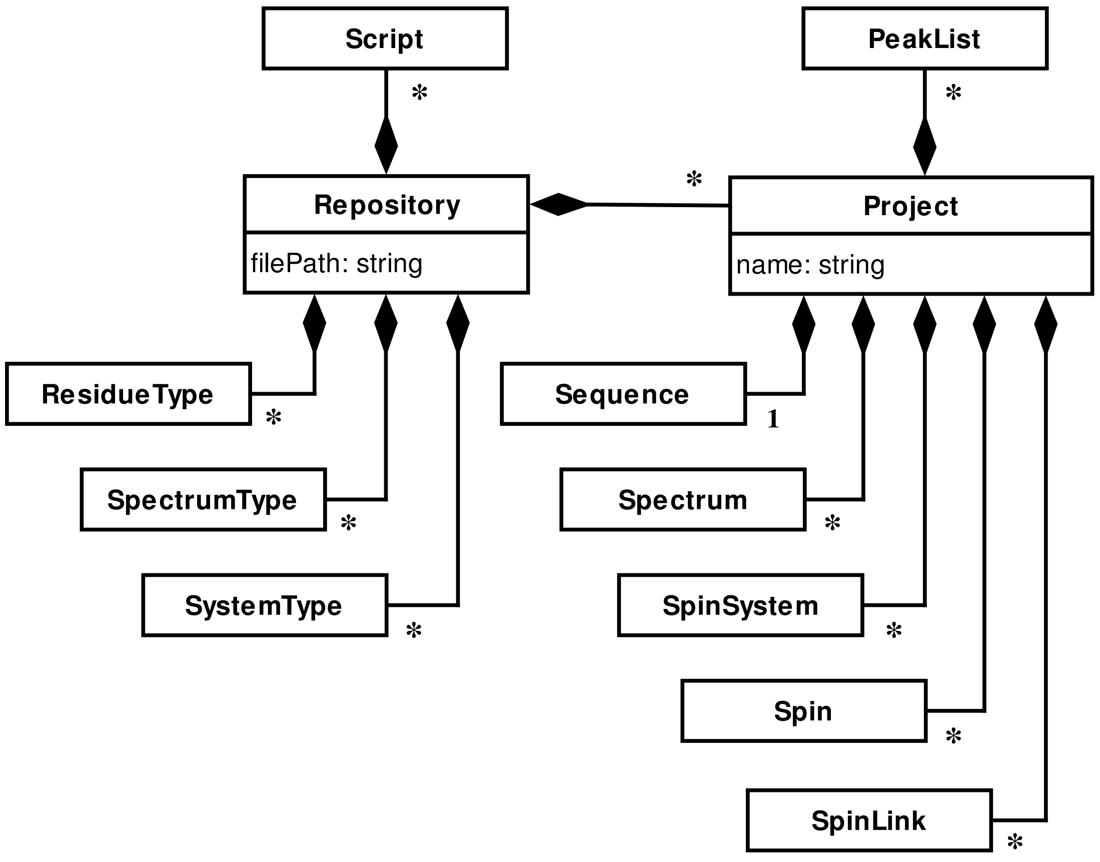
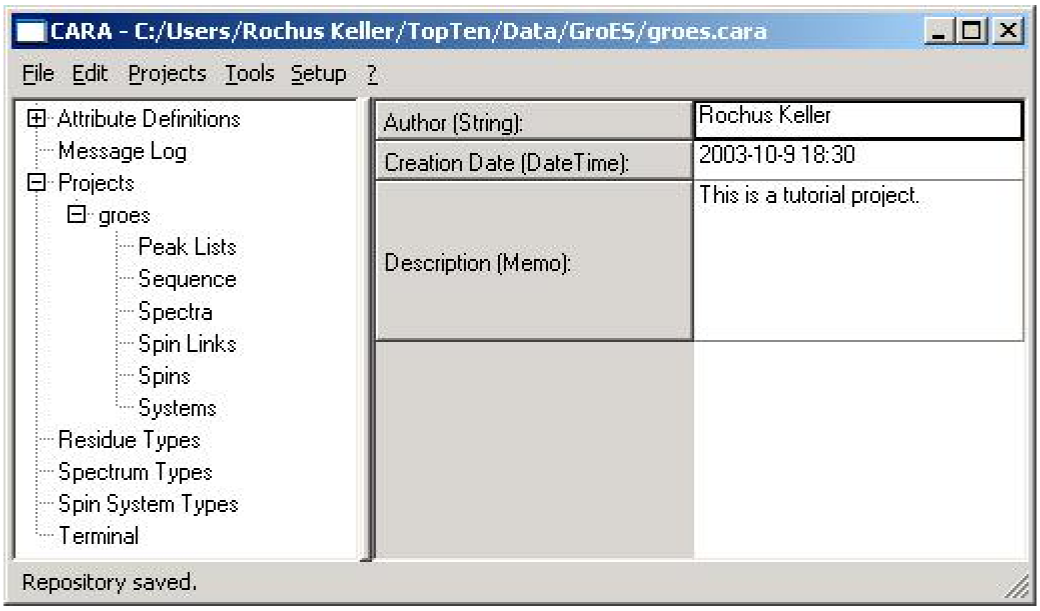


Figure 3: Structure of a Repository

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 4: CARA Explorer

The main purpose of the Explorer (Figure 4) 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 3.11).
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 4 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 *SpinLins* 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 appendix on page 66).

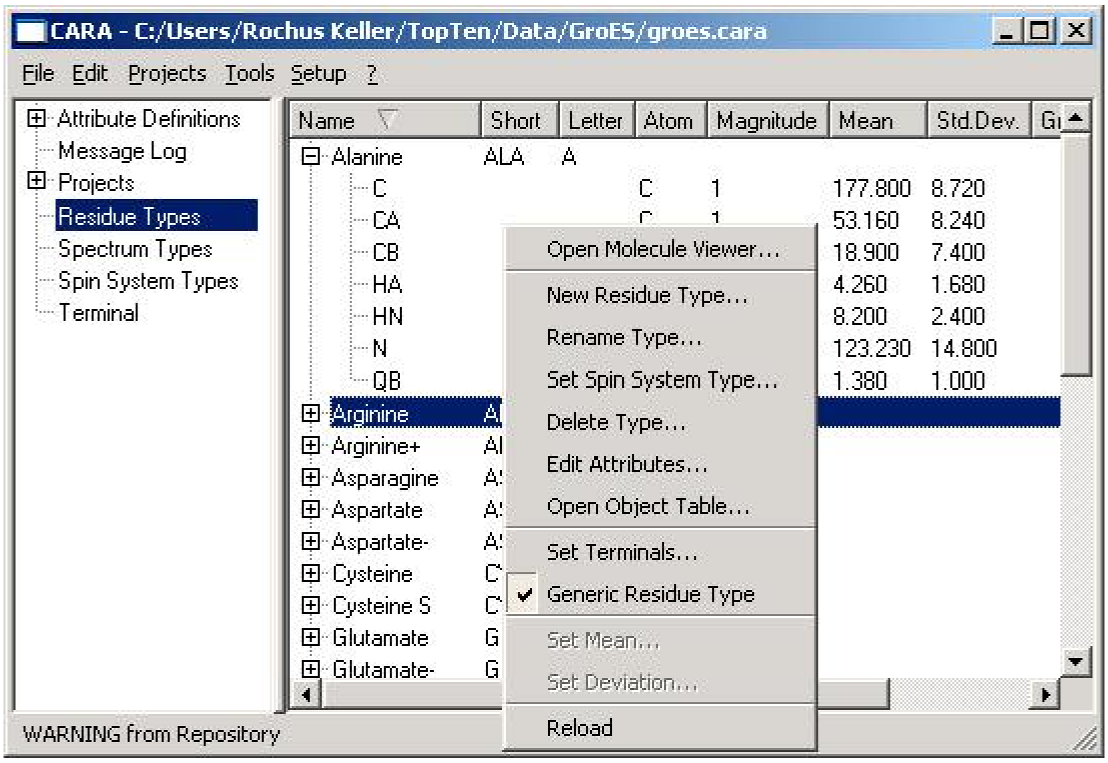
3.2 Setting up a Repository

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 3.3). For all other users we briefly explain how to survive without a template.

3.2.1 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 5 becomes visible (it might be empty or look different depending on how you started).



Name	Short	Letter	Atom	Magnitude	Mean	Std.Dev.	Gı
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+	Al						
Asparagine	A!						
Aspartate	A!						
Aspartate-	A!						
Cysteine	C						
Cysteine S	C						
Glutamate	G						
Glutamate-	G						

Figure 5: Explorer Residue Types pane

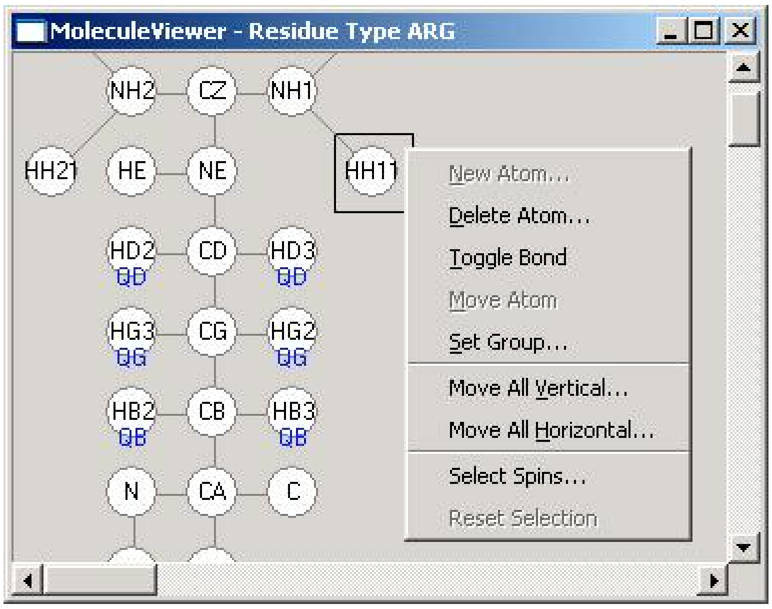
The figure also shows the context menu, which appears when the right mouse button is pressed (or the left button together with the command key on Macintosh OSX). You create a new residue type by executing the corresponding menu entry. A dialog box appears where you have to enter three alternative name 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 [Keller 2004] and chapter 3.6).

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). Both options affect the whole repository (i.e. all contained projects).

The options *Edit Attributes* and *Open Object Table* will be explained in chapter 3.11.

The molecule information of a new or existing type can be changed within the *Molecule Viewer* (Figure 6). The figure again shows the context menu.



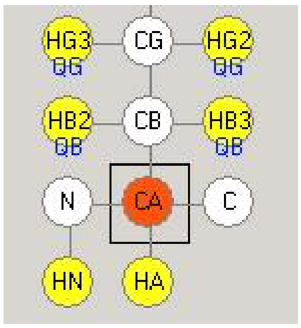


Figure 6: Molecule Viewer with Residue Type Arginine (left) and the effect of *Select Spins* on a CA with hopCount 3 on H (right)

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 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 onto 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 6, see also chapter 4.3 in [Keller 2004] for explanation).

3.2.2 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 7 appears.

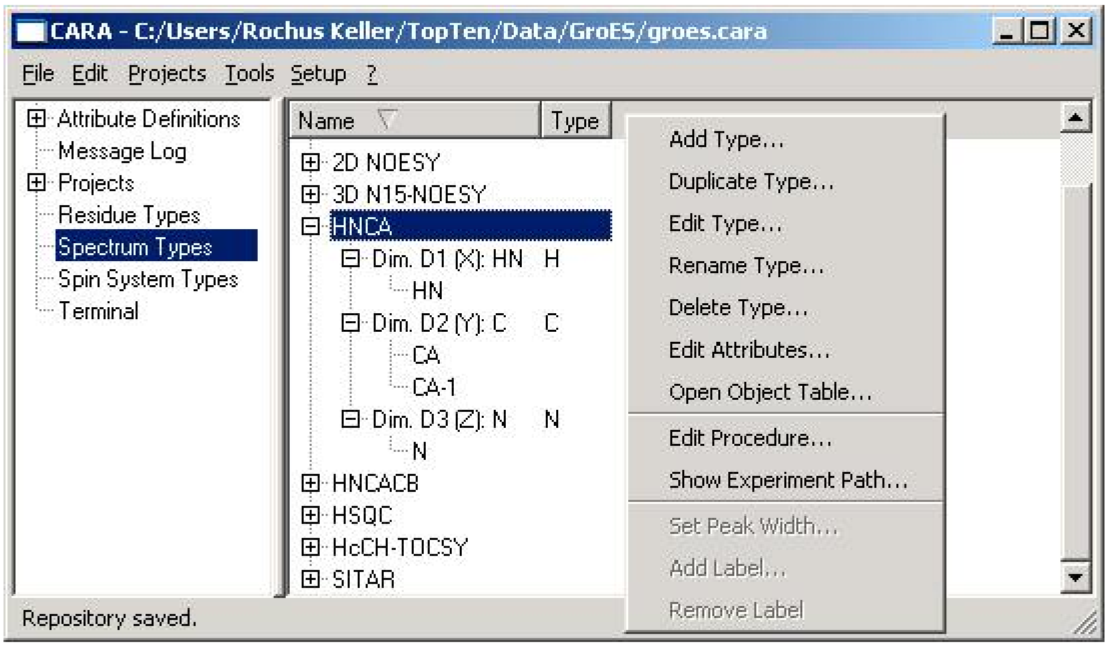
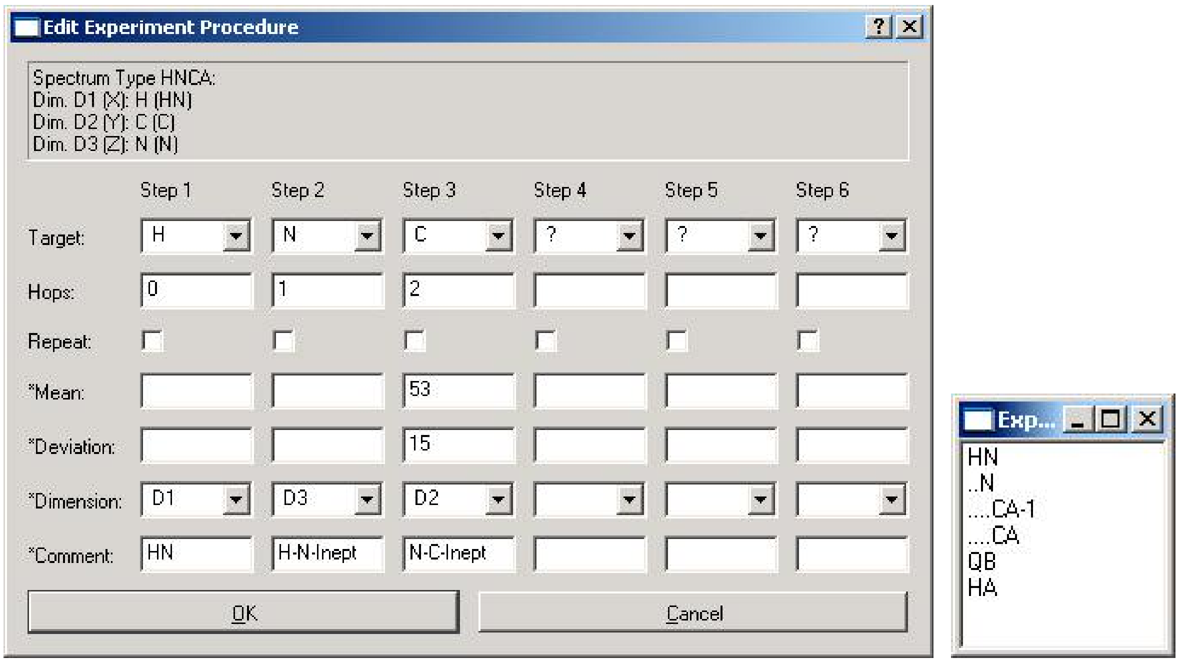


Figure 7: Explorer Spectrum Types Pane

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 7 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 3.4).

For each spectrum type an experiment procedure can optionally be defined. Figure 8 shows the dialog for procedure editing. The periods of the NMR experiment are declared in order of their execution (see chapter 4.3 in [Keller 2004]). The *Dimension* attribute is then used to map them onto 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.



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			

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

Figure 8: Experiment Procedure of HNCA (left) and simulated magnetization 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 8, 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 was explained in chapter 4.3.1 in [Keller 2004].

3.3 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 1: Sequence file variations, plain (left), with residue numbers (mid) and with assignments (right)

```
MET
ASN
ILE
ARG
PRO
LEU
```

```
MET    1
ASN    2
ILE    3
ARG    4
PRO    5
LEU    6
```

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

The plain variation 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.

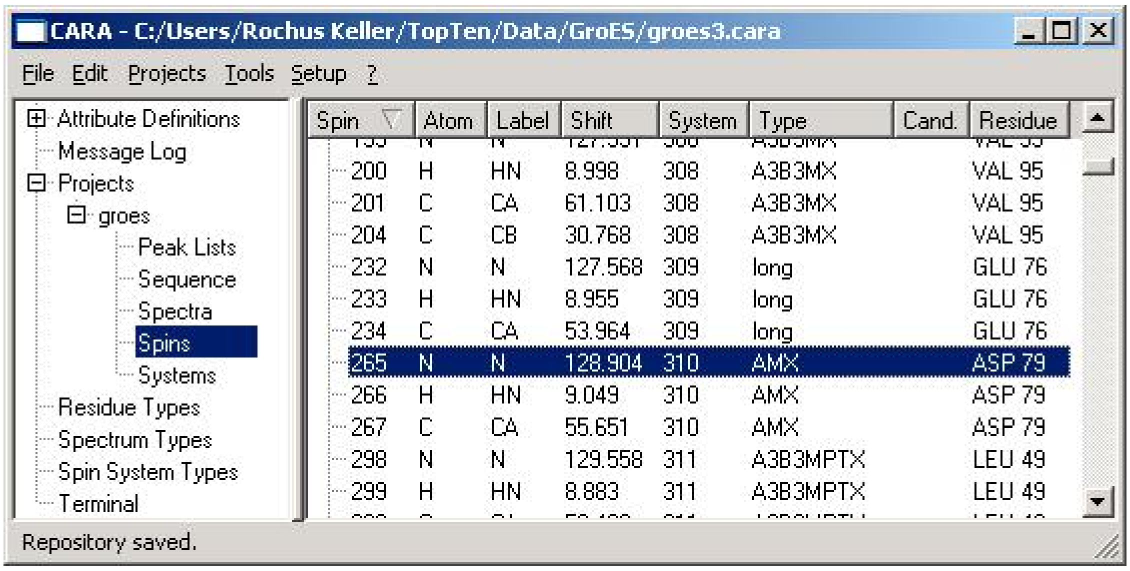
The user might optionally import an atomlist (using EASY proton list format) if one is abailable (i.e. corresponding to the sequence loaded). The user does so by selecting the project name in the category tree of the Explorer and activating *Project/Import/Atom List* from the menu. The user is then asked 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 atomlist 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 atomlist 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 atomlist 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 atomlist 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 2: Part of the imported atomlist

```
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 9). The panes are empty of course, if no atomlist was imported and no spins were created otherwise.



CARA - C:/Users/Rochus Keller/TopTen/Data/GroES/groes3.cara

File Edit Projects Tools Setup ?

- Attribute Definitions
- Message Log
- Projects
  - groes
    - Peak Lists
    - Sequence
    - Spectra
    - Spins
    - Systems
- Residue Types
- Spectrum Types
- Spin System Types
- Terminal

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

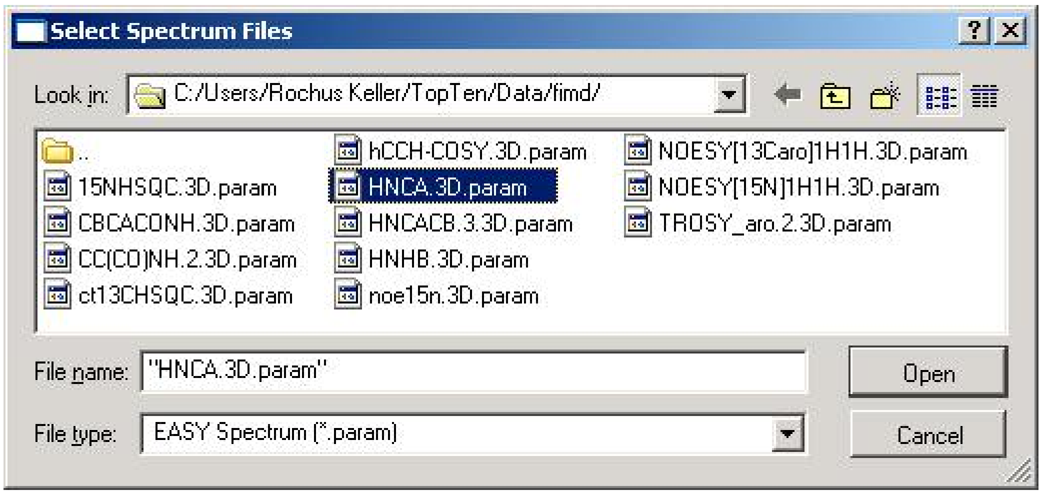
Repository saved.

Figure 9: 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 10 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).



Look in: C:/Users/Rochus Keller/TopTen/Data/fimd/

..
15NHSQC.3D.param
CBCACONH.3D.param
CC(CO)NH.2.3D.param
ct13CHSQC.3D.param
hCCH-COSY.3D.param
HNCA.3D.param
HNCACB.3.3D.param
HNHB.3D.param
noe15n.3D.param
NOESY[13Caro]1H1H.3D.param
NOESY[15N]1H1H.3D.param
TROSY_aro.2.3D.param

File name: "HNCA.3D.param"

File type: EASY Spectrum (*.param)

Figure 10: Spectrum file selector dialog

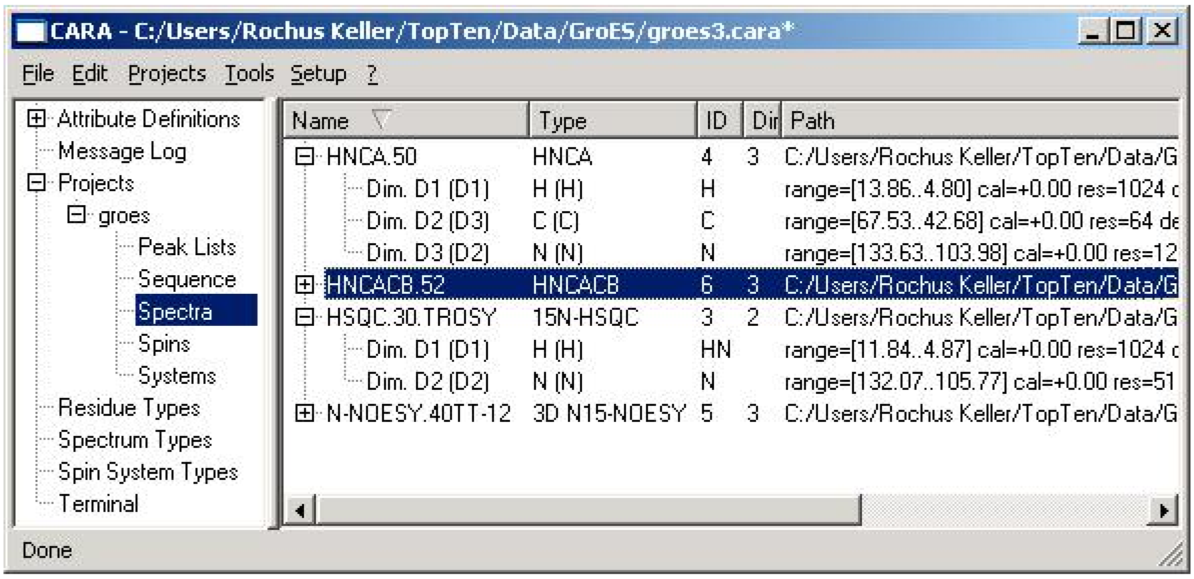
CARA tries to automatically rotate the dimensions of the loaded spectrum, so that they fit the given spectrum type. Depending on the spectrum file format, there are different hints about the atom types related with the dimensions. For example in a EASY param file (Table 3) the lines *Identifier for dimension* 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 3: 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.


CARA - C:/Users/Rochus Keller/TopTen/Data/GroES/groes3.cara*

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 11: Spectrum pane of the project after loading spectra

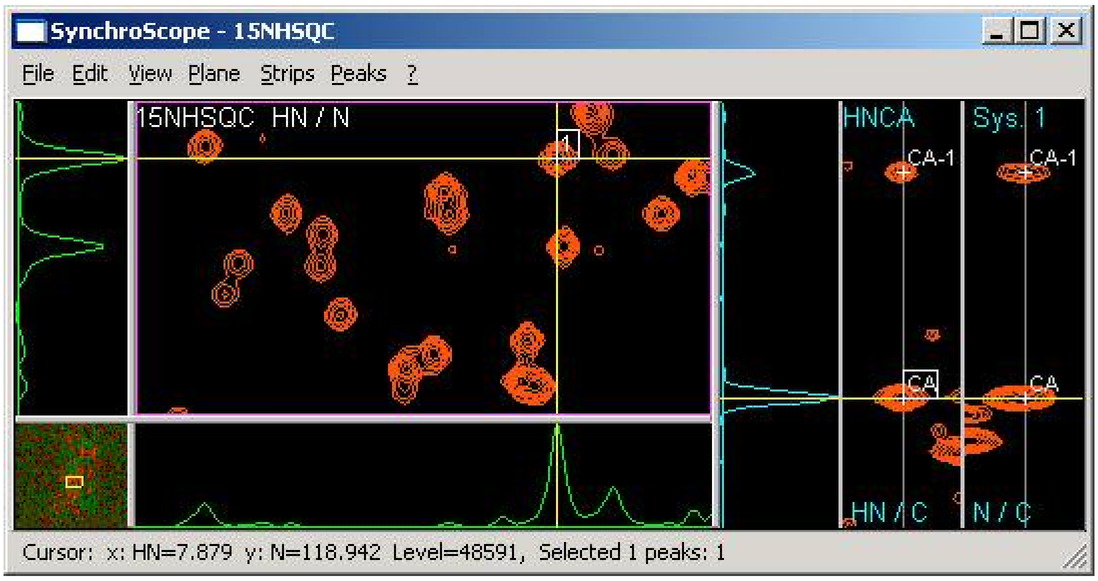
A spectrum can be removed any time, or the spectrum file can be replaced to redirect the the reference to another file (context menu items *Remove Spectrum* and *Replace Spectrum*). The latter can also be used if the spectrum file was renamed or moved to another directory (but don't do this while they're 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 11) 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.

3.4 Heteronuclear Backbone Assignment

Chapter 2.3 introduced general strategies for sequence specific assignment. In this chapter we will show how backbone assignment is accomplished using double and triple-resonance spectra like 15N-HSQC, HNCA, HNCACB and 15N-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 11). 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 will now start by selecting a 15N-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 7).



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

Figure 12: Identifying spin systems with SynchroScope

SynchroScope allows the simultaneous analysis of a 2D 15N-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 12 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 appendix on page 66). 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:

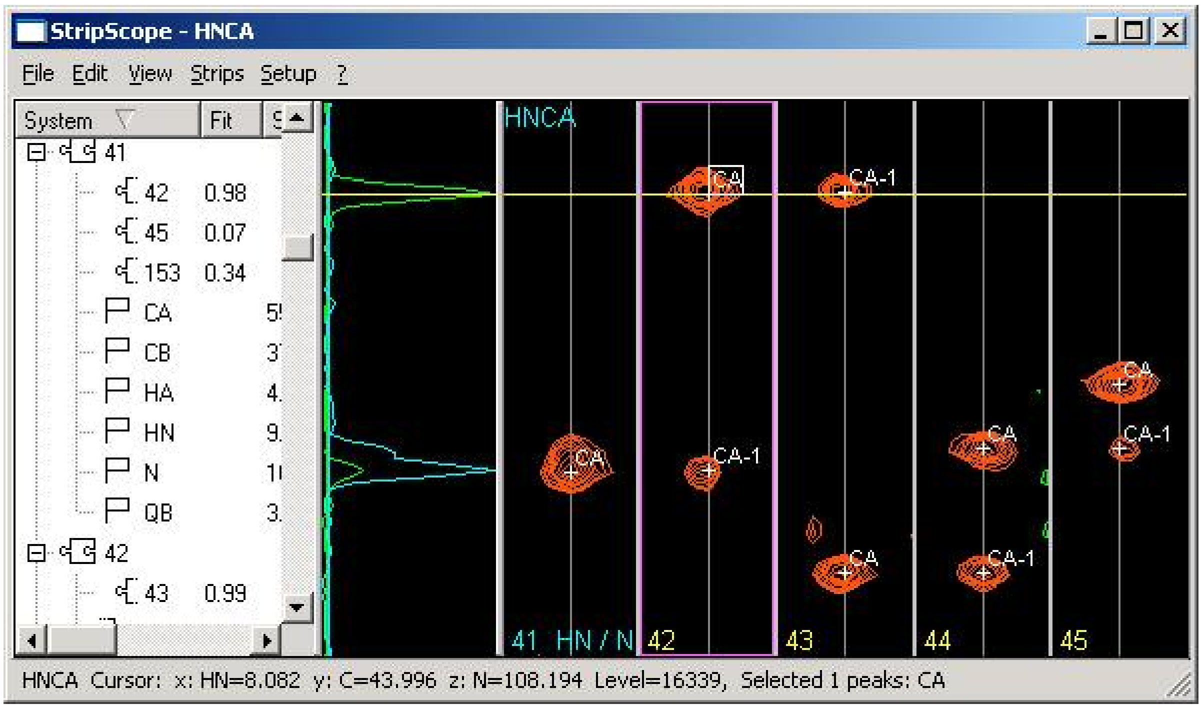
1. Select a single spin system peak within the HSQC plane and position the cursor on a peak within the strips.
2. Uncheck the menu *View/Hide 3D Slices* and check the menu *Plane/Show 3D Plane*. The selected HNCA plane is now displayed.

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 7).

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).



```
System      Fit
⊟ 41
    42    0.98
    45    0.07
    153   0.34
    CA          5
    CB          3
    HA          4.
    HN          9.
    N           1
    QB          3.
⊟ 42
    43    0.99
```

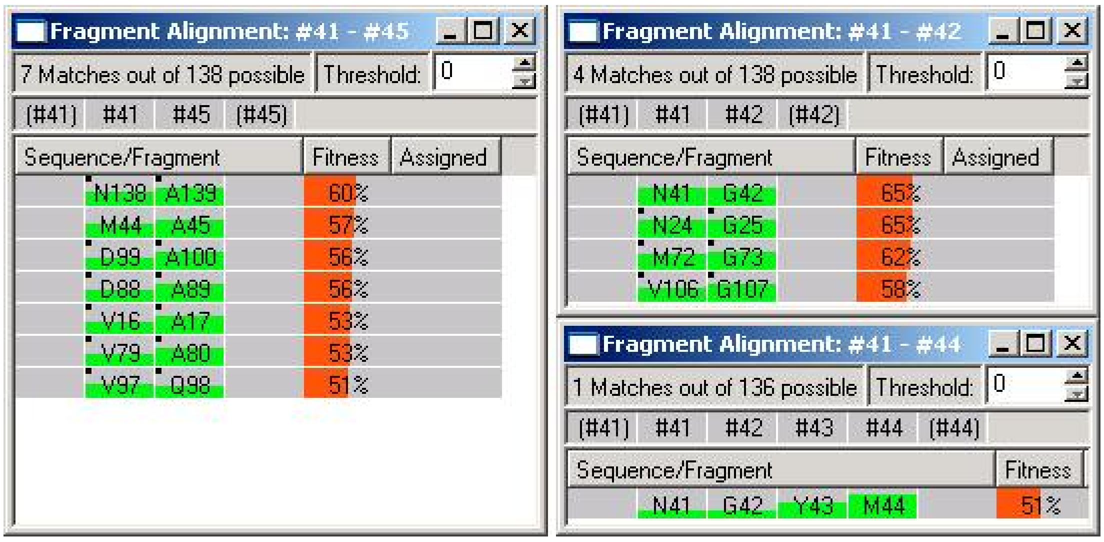
```
HNCA  Cursor:  x: HN=8.082  y: C=43.996  z: N=108.194  Level=16339,  Selected 1 peaks: CA
```

Figure 13: 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 in [Keller 2004]). The results of the algorithm are shown in the system tree (as split jigsaw pieces). In Figure 13 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 in [Keller 2004]. This supports for one part the decision about the placement of a fragment onto the sequence (i.e. the sequence-specific assignment), but can also be used as a decision aid which fragments should be combined (see Figure 14).



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%

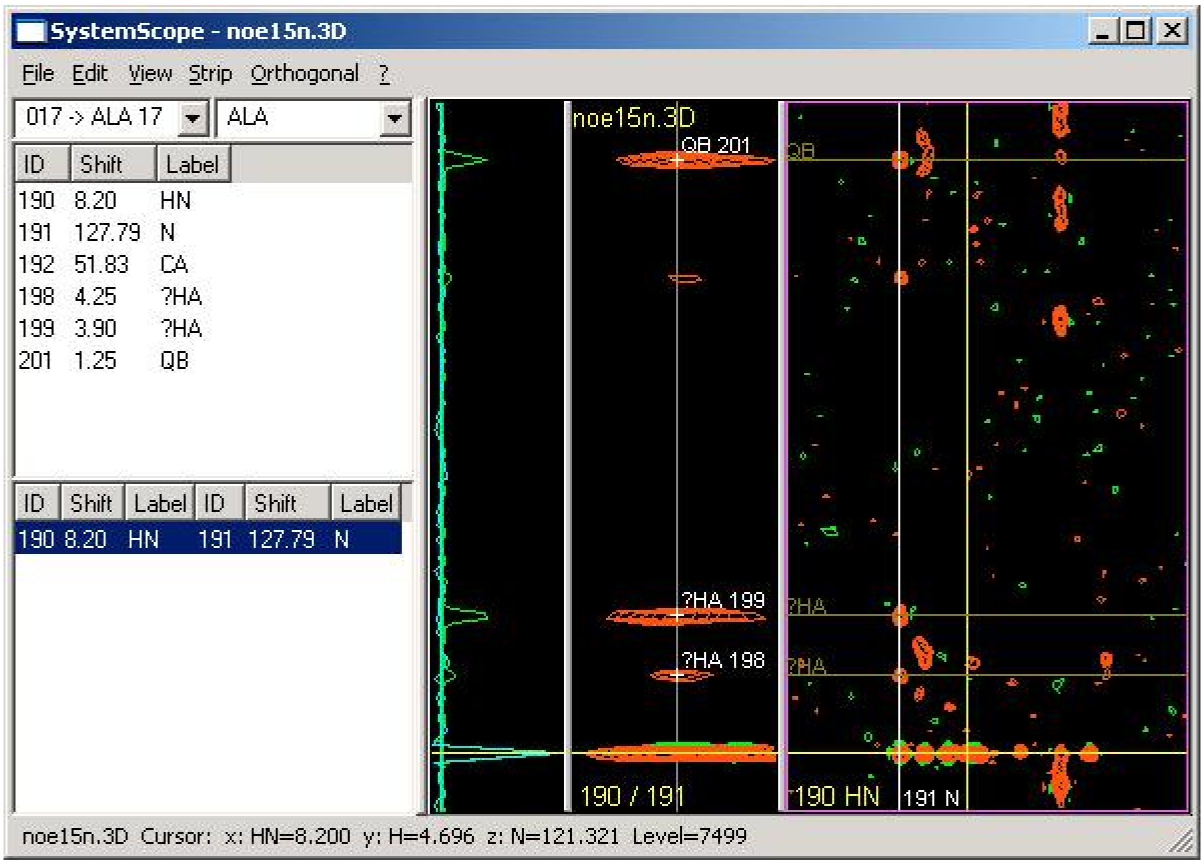
Figure 14: Sequence mapping of three fragment variations

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).

3.5 Heteronuclear Sidechain 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.



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

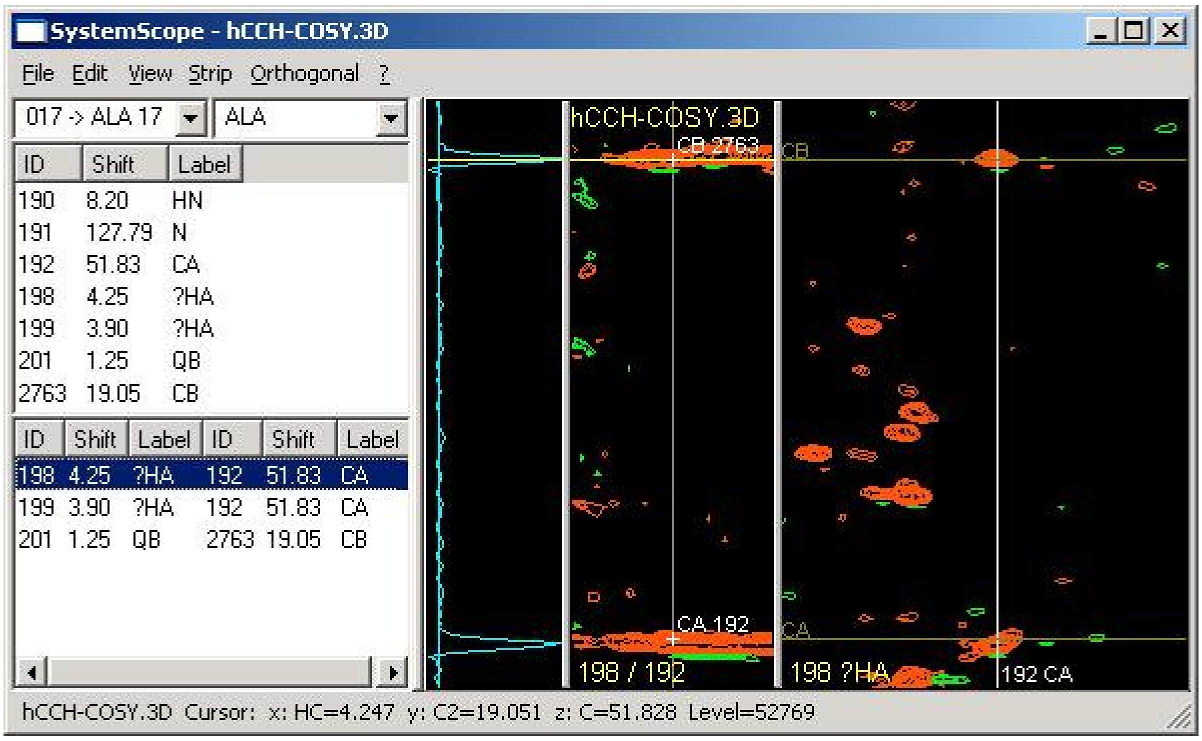
ID	Shift	Label	ID	Shift	Label
190	8.20	HN	191	127.79	N

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

Figure 15: N/HN strip of ALA 17 in 15N-NOESY

The main tool for sidechain assignment is SystemScope. We will now start by selecting a 15N-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 has four empty intensity peaks (see Figure 15). 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 15.



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

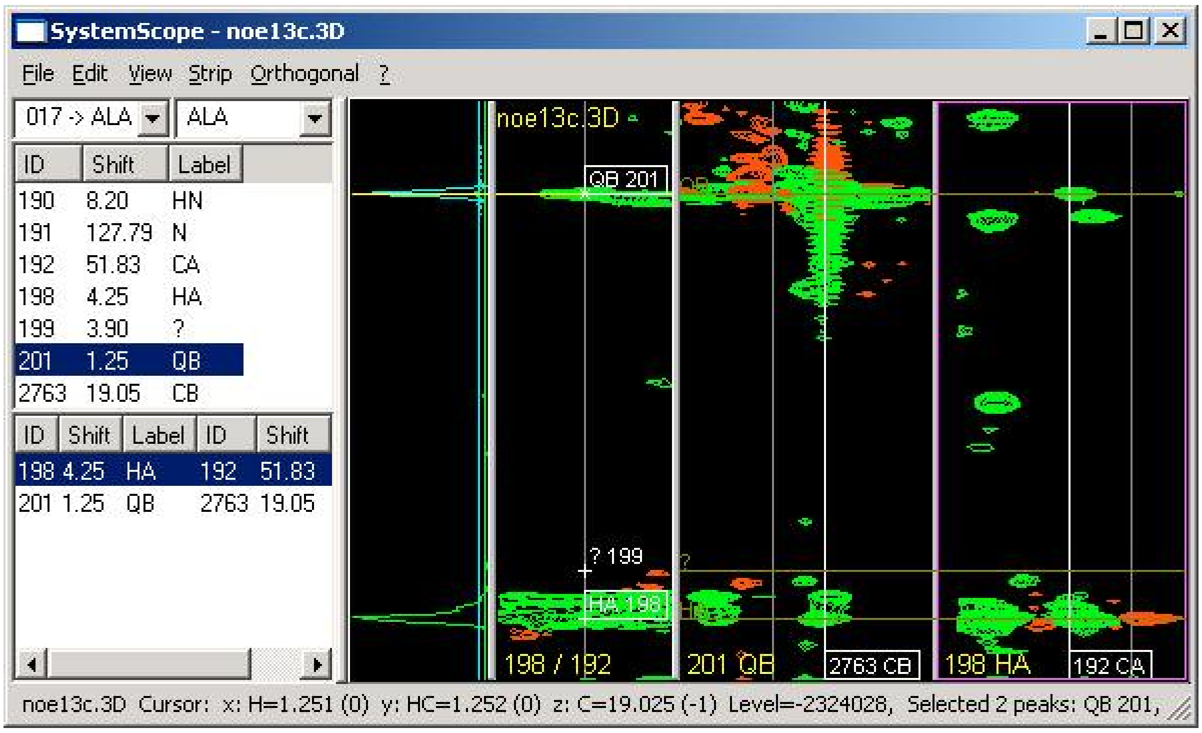
ID	Shift	Label	ID	Shift	Label
198	4.25	?HA	192	51.83	CA
199	3.90	?HA	192	51.83	CA
201	1.25	QB	2763	19.05	CB

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

Figure 16: 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 16, 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 $^1H/^{13}C$ 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 16 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 17: Detecting CB in the orthogonal plane with origin QB

In Figure 17 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.

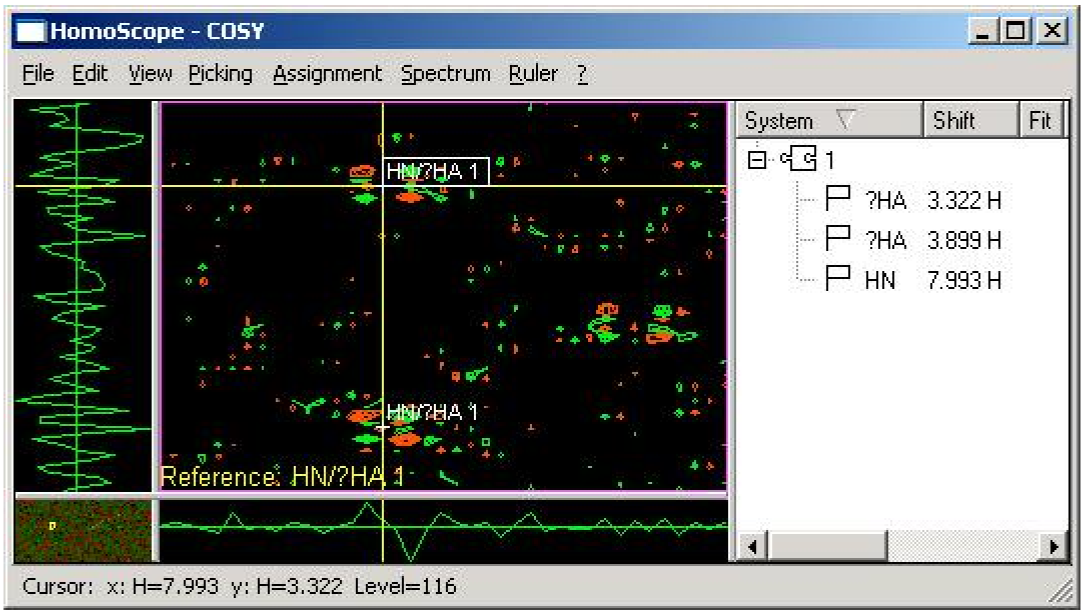
The completeness of a sidechain assignment can be checked for example by monitoring whether all peaks on a 13C-HSQC spectrum are identified. This is possible, if the 13C-HSQC is simultaneously displayed in a HomoScope window.

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.

3.6 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 3.3) 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).



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

Figure 18: Picking a Glycine in a 2D COSY spectrum

The pattern in Figure 18 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 18). 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 19. 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).

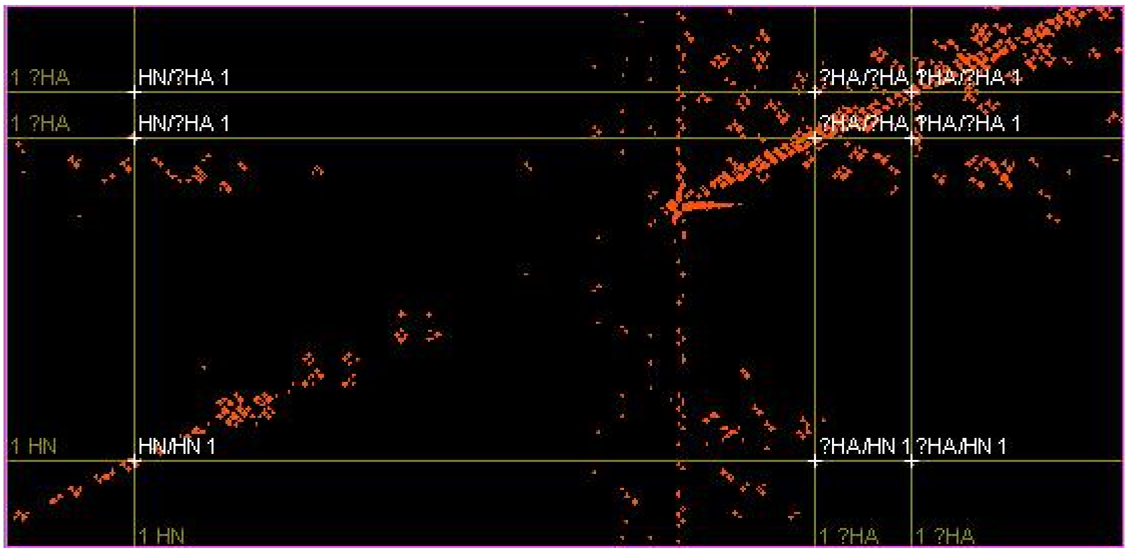
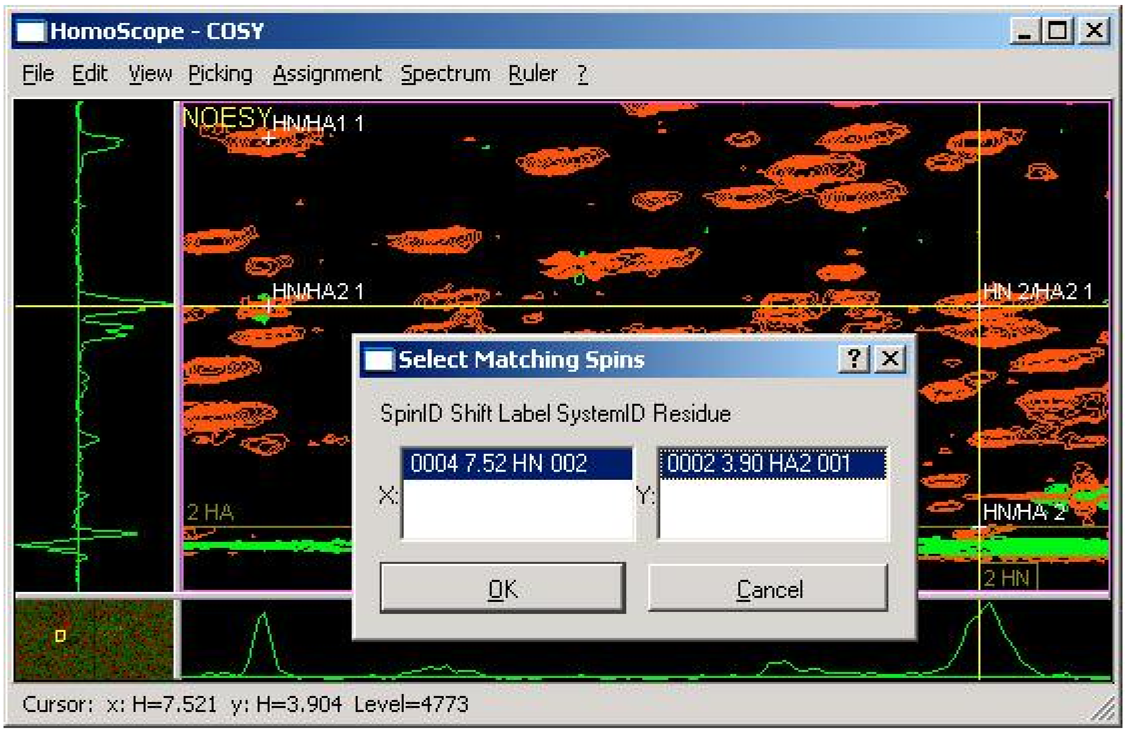


Figure 19: Detail of spin system pattern with rulers (positive contours only)

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 20). If too many spins are displayed, one should narrow the tolerance using menu *Picking/Set Verti. Tolerance*.



SpinID Shift Label SystemID Residue

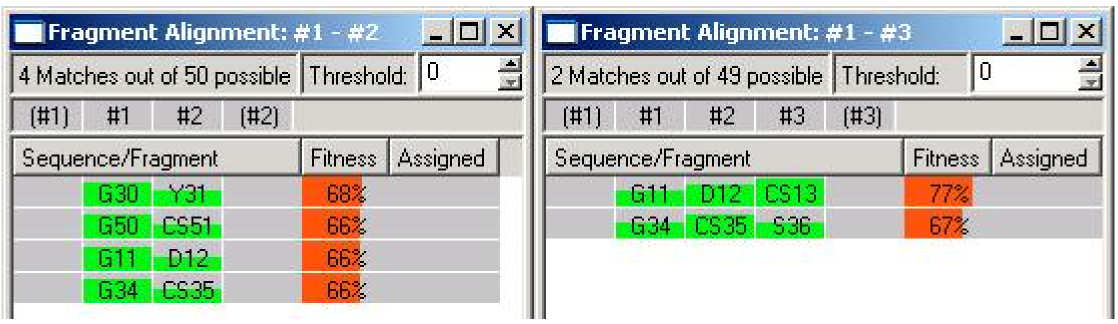
X: 0004 7.52 HN 002

Y: 0002 3.90 HA2 001

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

Figure 20: 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 in [Keller 2004]). The first line in the right part of Figure 21 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 21: Fragment alignment of length 2 (left) and length 3 (right)

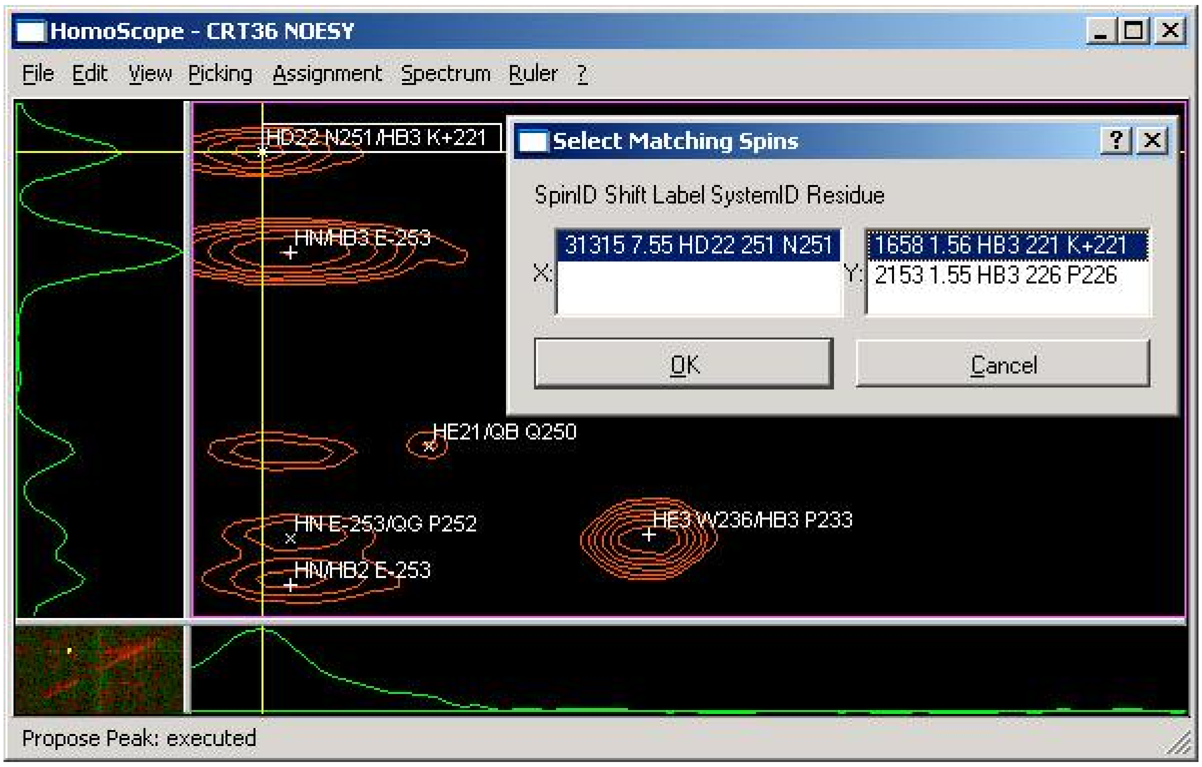
3.7 Constraint Gathering

With the advent of algorithms like ATNOS [Herrmann et al. 2002a] it should be no longer necessary to manually do constraint gathering and peak integration. 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 in this document and 4.4 of [Keller 2004] 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 in [Keller 2004]. 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 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 22). The spins are selected using Eq. 6 (see chapter 4.2.6 in [Keller 2004]) 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

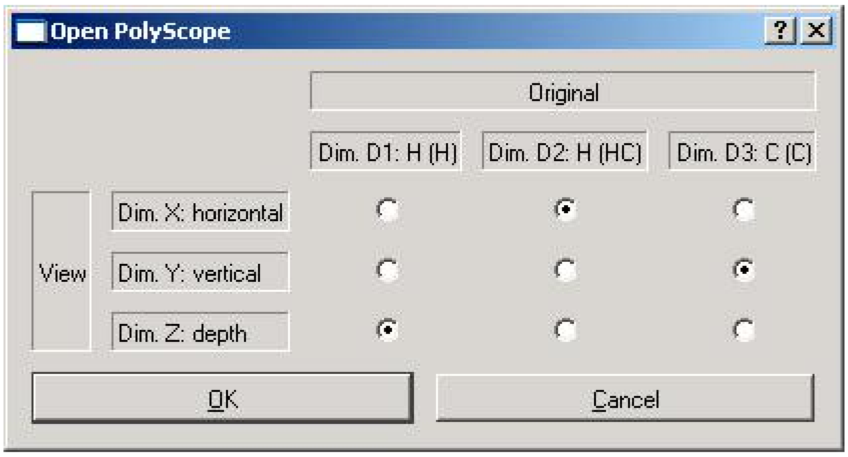
Propose Peak: executed

Figure 22: 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 integration (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 13C-NOESY has probably to be rotated to present the $^{1}H/^{13}C$ correlation in the plane and the NOESY dimension in the strip.

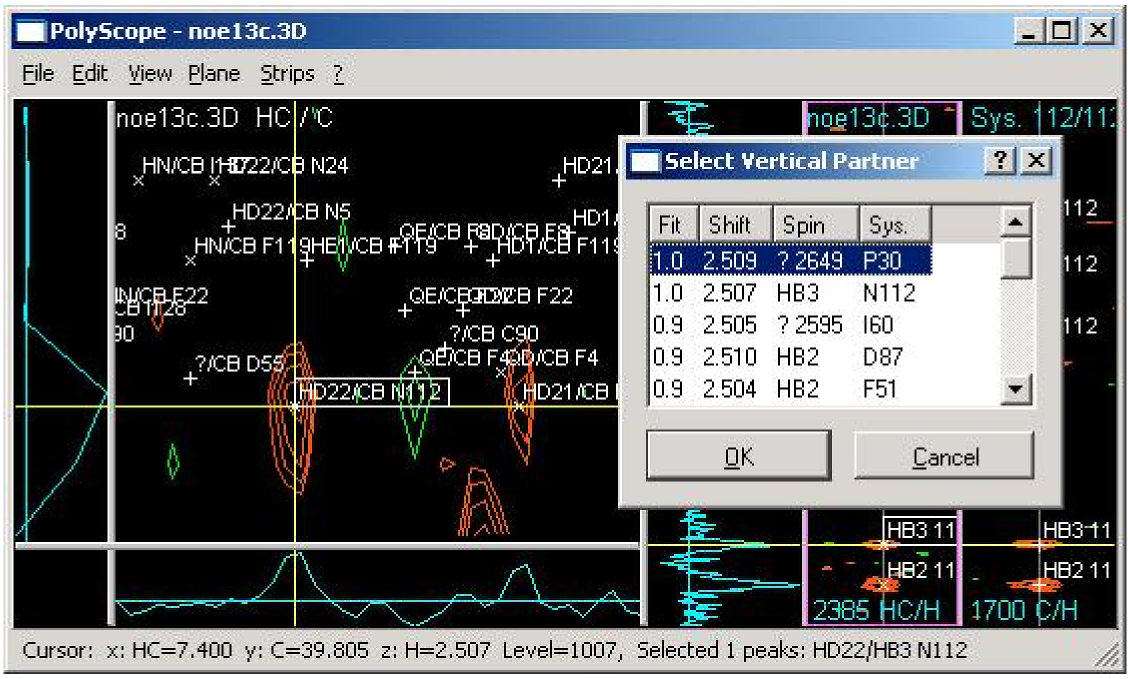


Open PolyScope

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

Figure 23: 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 23 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 24 shows a 13C-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 in [Keller 2004]).



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
```

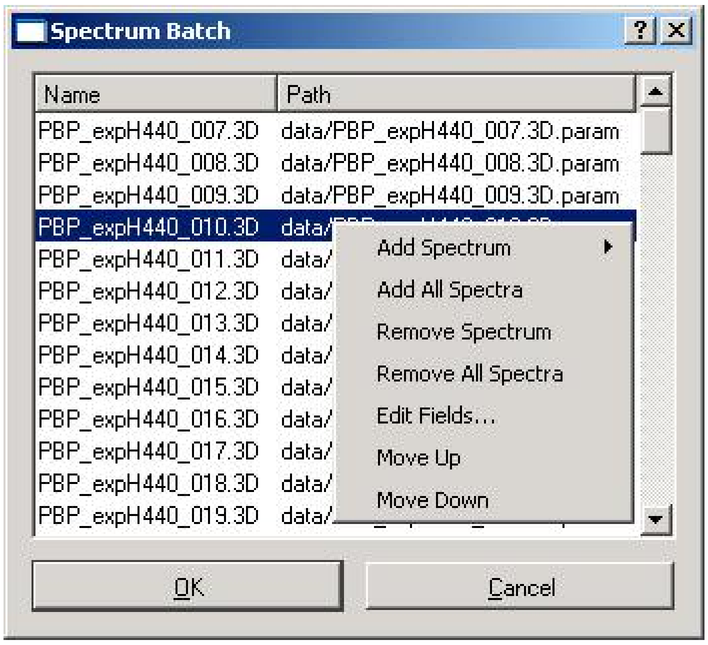
Figure 24: Identifying peaks in a ^{13}C-NOESY using PolyScope

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).

3.8 Peak Integration

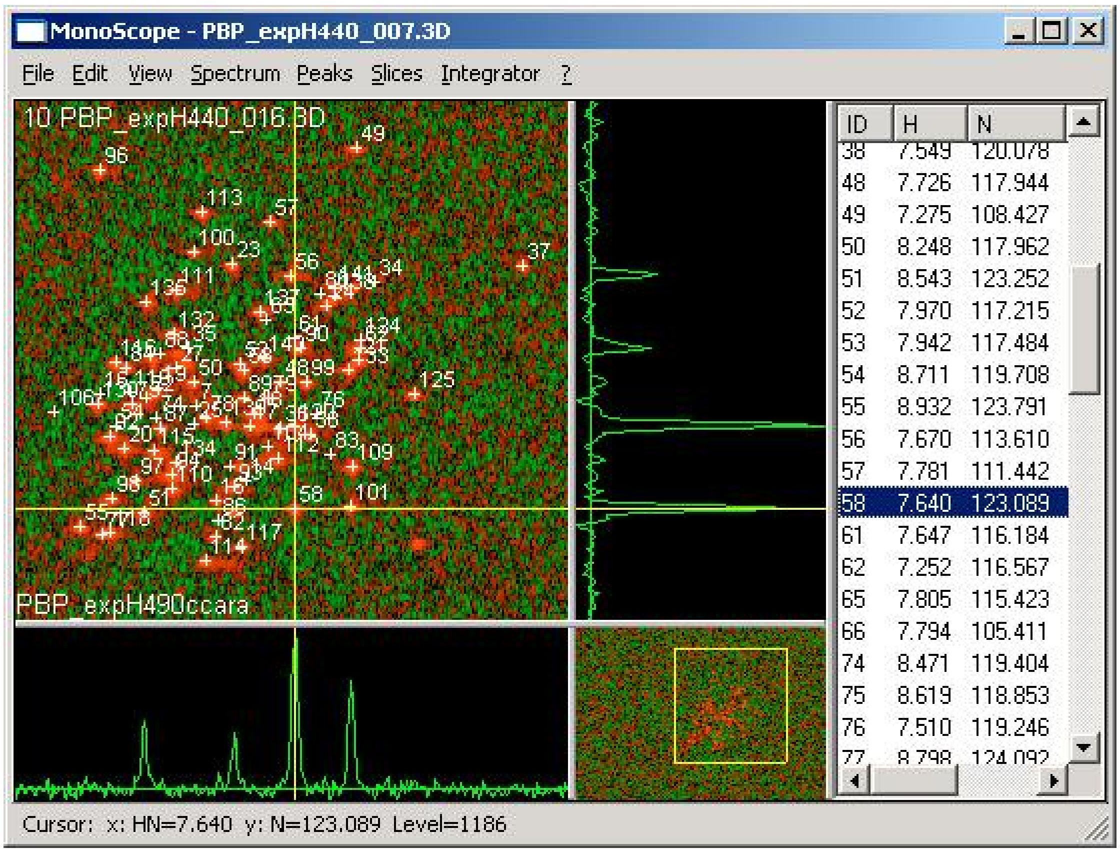
In this section we give a short overview on how to use CARA for peak integration. We take the D_2O exchange of Pheromone binding protein from Bombyx mori at pH 4.5 as an example [Lee et al. 2002]. The same procedure can be applied to NOESY peak integration with the difference of having only one spectrum 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 25 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 25: Spectrum Batch List

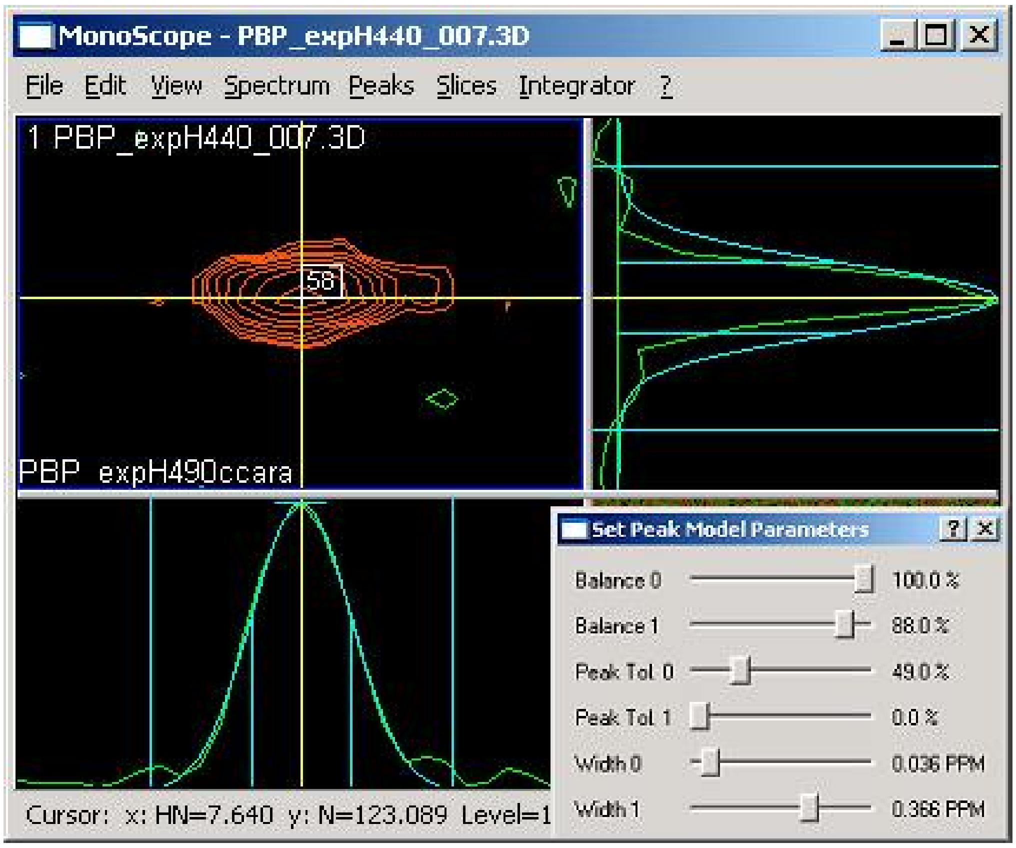

MonoScope - PBP_expH440_007.3D
File Edit View Spectrum Peaks Slices Integrator ?
10 PBP_expH440_016.3D
PBP_expH490ccara

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 26: 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 26 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.


MonoScope - PBP_expH440_007.3D
File Edit View Spectrum Peaks Slices Integrator ?
1 PBP_expH440_007.3D
58
PBP expH490ccara
Set Peak Model Parameters

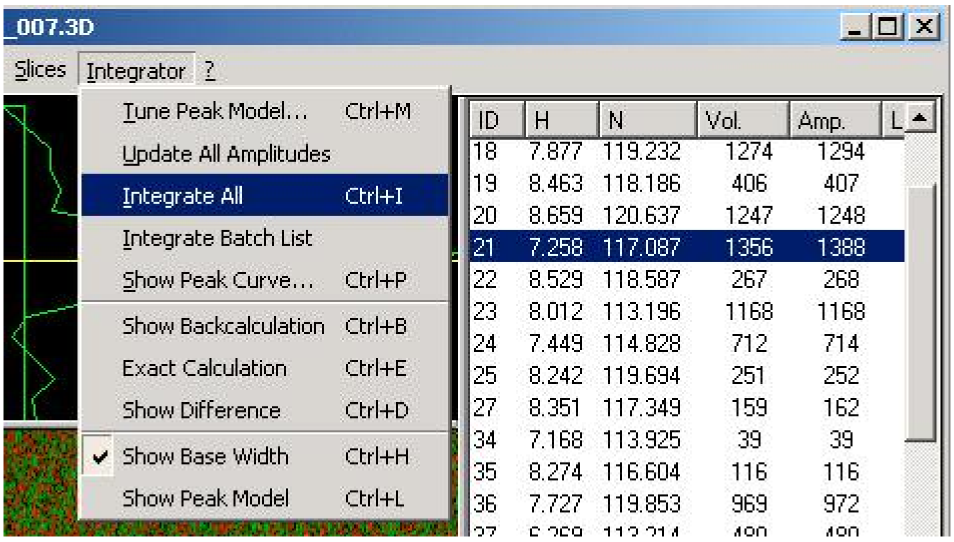
Balance 0	100.0 %
Balance 1	88.0 %
Peak Tol 0	49.0 %
Peak Tol 1	0.0 %
Width 0	0.036 PPM
Width 1	0.366 PPM

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

Figure 27: 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 27. 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 integration can be started using the corresponding menu (see Figure 28).

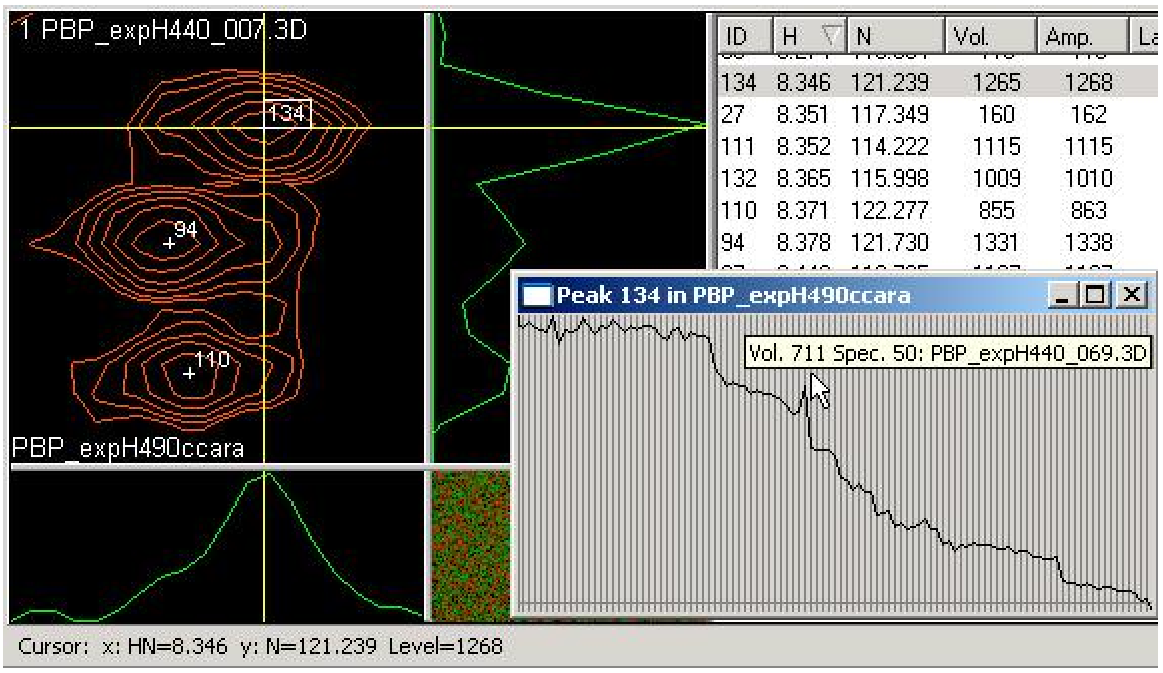
It is possible to integrate a single spectrum or alternatively the whole batch list in one pass. The integration of the batch list of Pheromone Binding Protein (107 spectra) took less than 10 seconds on a laptop (Windows XP, 2.4 GHz). The results of the integration are immediately visible in the peaklist (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 28: 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 29 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 integration. Because integration nearly happens in real-time, the user can apply an incremental and iterative optimization approach.


1 PBP_expH440_007.3D
134
94
110
PBP_expH490ccara
Peak 134 in PBP_expH490ccara
Vol. 711 Spec. 50: PBP_expH440_069.3D

ID	H	N	Vol.	Amp.
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

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

Figure 29: 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 integration by comparing the backcalculated and difference spectrum with the original one (Figure 30). Eq. 14 in [Keller 2004] 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.

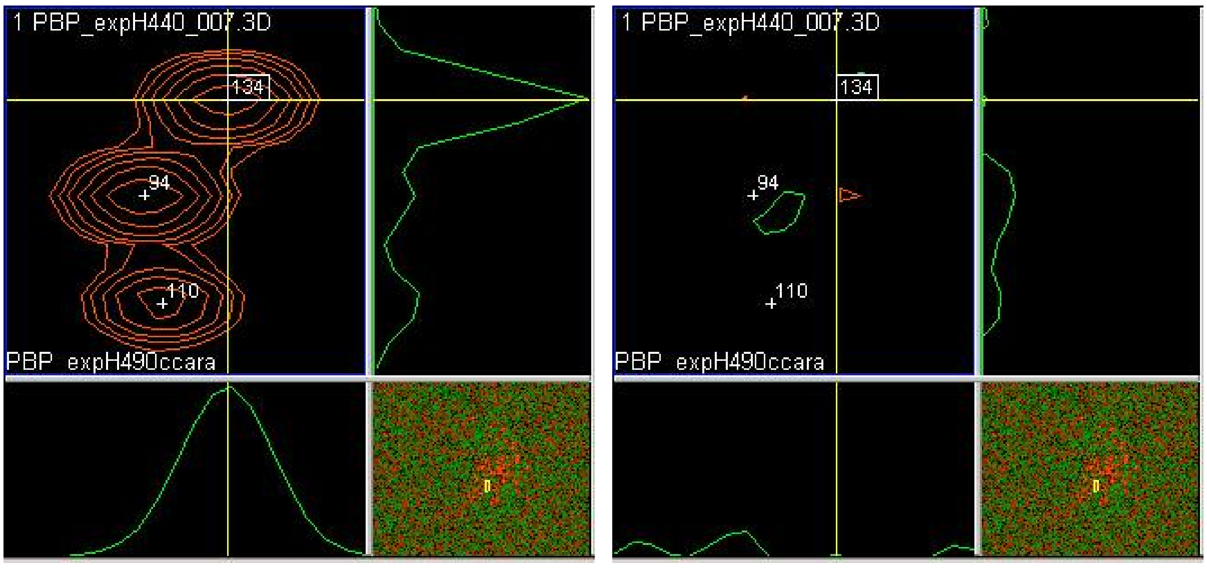


Figure 30: The backcalculated spectrum (left) corresponding to Figure 29 and the difference spectrum (right)

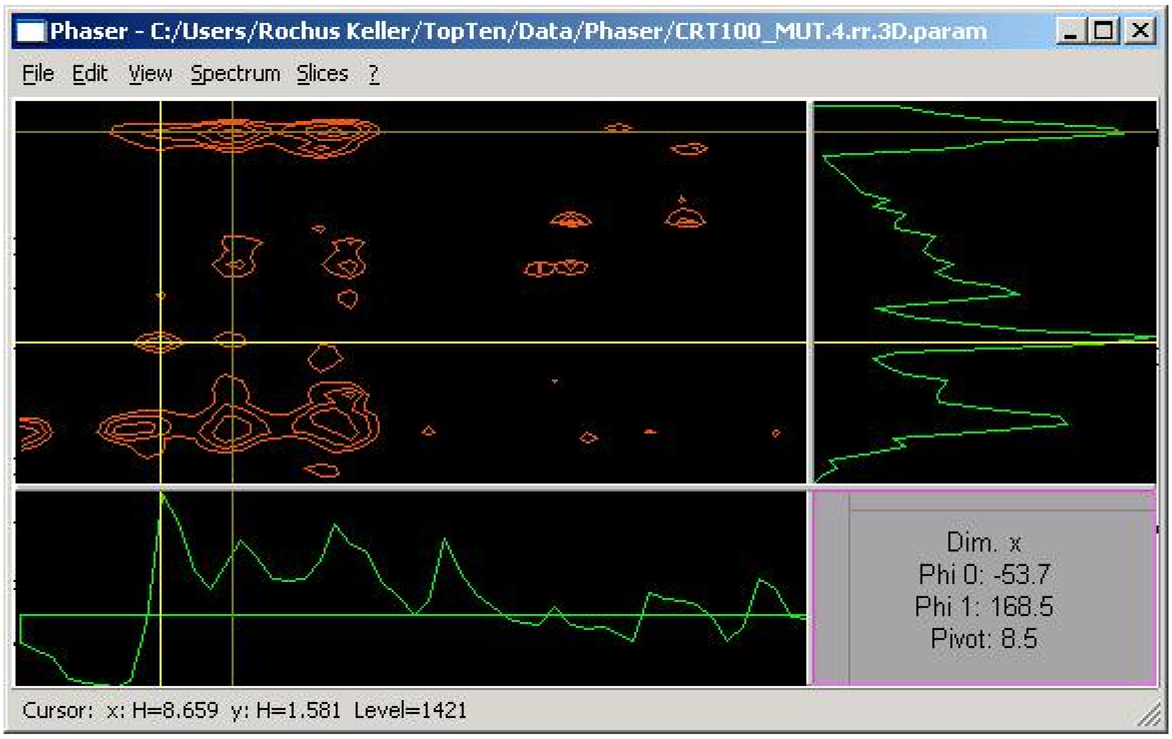
In the right part of Figure 30 the peaks have nearly disappeared, i.e. there is little intensity remaining. In case a peak was covered before or accidentally not picked, it would now appear in the difference spectrum. The user could pick the peak, reintegrate and immediately see the effect in the spectrum. This incremental and iterative process would continue until the user was satisfied by the result.

3.9 Phasing

As part of the preprocessing 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 31 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 31: 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 preprocessing software in use.

3.10 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 32 appears (the spectra come from the GroES project [Fiaux et al. 2002]).

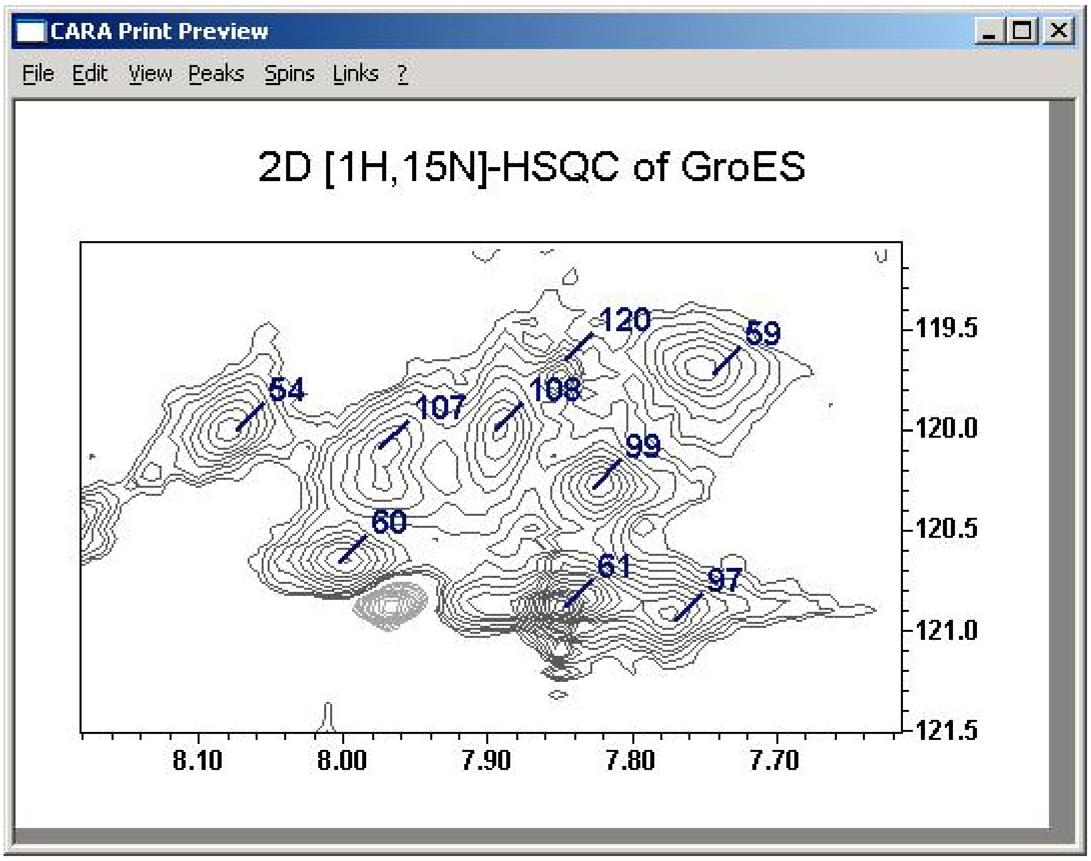


Figure 32: Print Preview window showing HN/N cross-peaks

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).

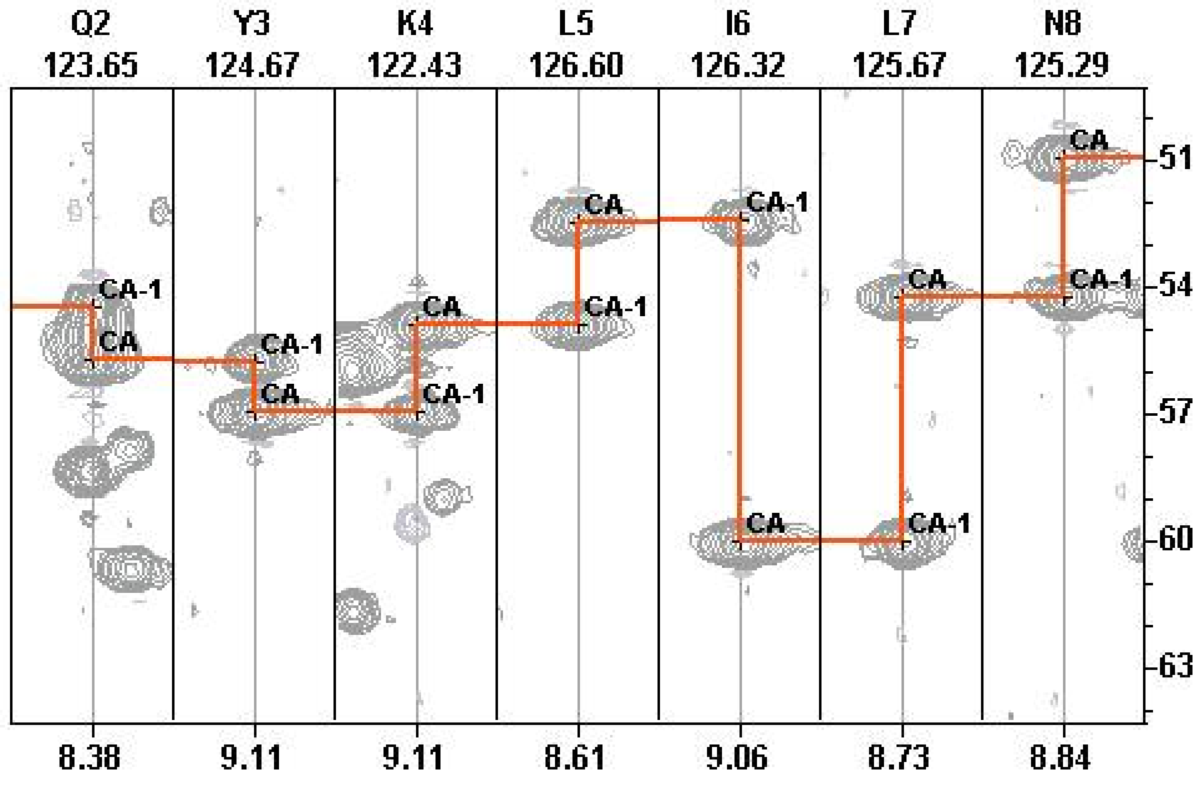
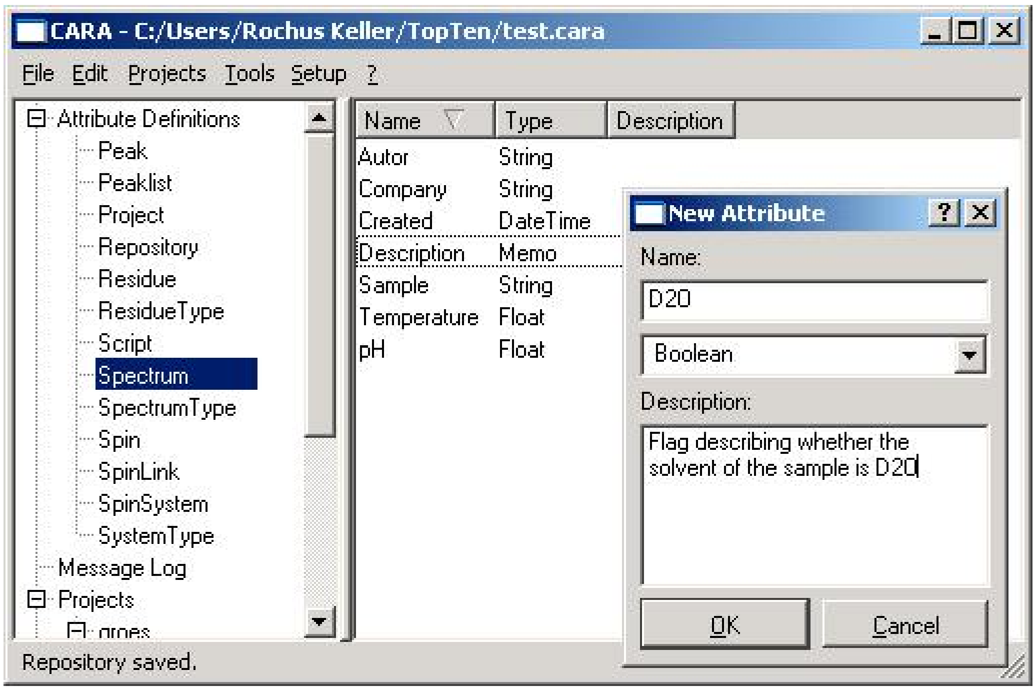


Figure 33: Print out of an assigned HNCA strip fragment

3.11 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 34). 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	

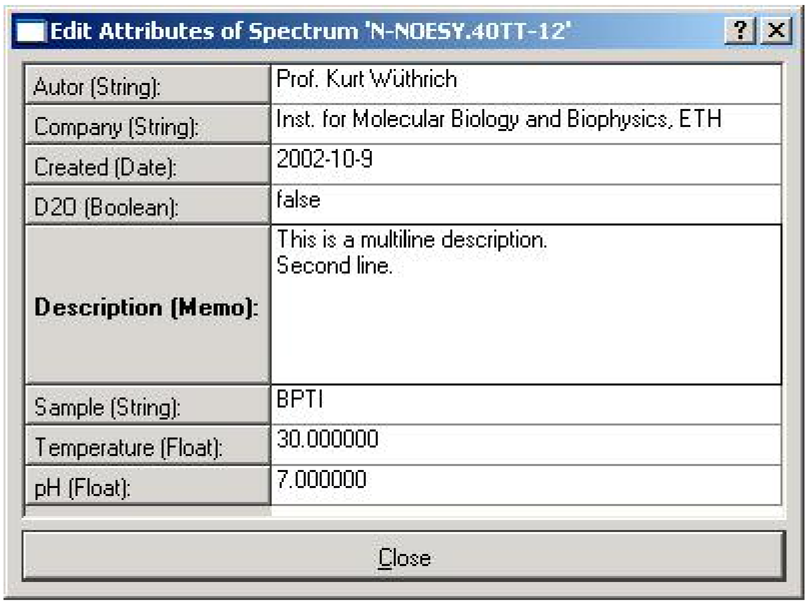
Name: D2O

Boolean

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

Figure 34: 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 11) and execute the menu command *Edit Attributes*. The attributes of most objects are edited using the dialog shown in Figure 35. The repository and project objects are privileged in that their attributes are directly editable from within a pane of the Explorer (see Figure 4). The height of the fields can be adjusted using the mouse.



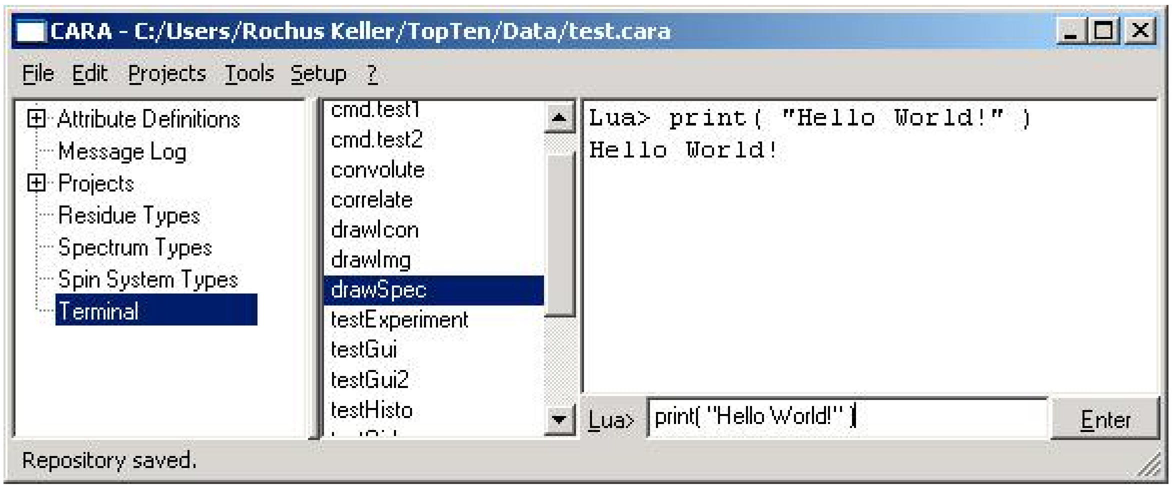
Autor (String):	Prof. Kurt Wüthrich
Company (String):	Inst. for Molecular Biology and Biophysics, ETH
Created (Date):	2002-10-9
D2O (Boolean):	false
Description (Memo):	This is a multiline description. Second line.
Sample (String):	BPTI
Temperature (Float):	30.000000
pH (Float):	7.000000

Figure 35: 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 an write the same attributes that the user can edit in the attributes dialog. An algorithm could calculate something and store the results in 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 part of the repository. The next chapter will show 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.

3.12 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 usual 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!
```

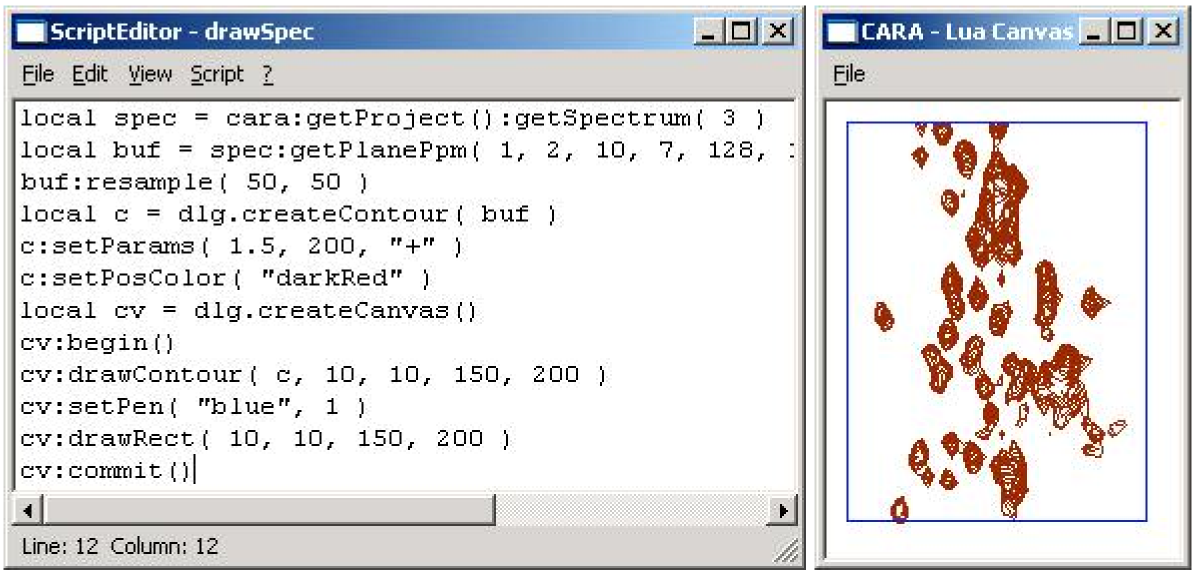
```
Lua> print( "Hello World!" )
```

Figure 36: The terminal pane of the CARA explorer

CARA/Lua consists of a full fledged programming environment with terminal, editors and persistent module management (Figure 36). Most objects managed by CARA can be programatically accessed 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 36), or she can create a new named script, which can then be edited using a dedicated scripting editor (Figure 37). The script becomes part of the repository for later reuse. Even the deployment of scripts by means the template concept (described in chapter 3.2) 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 she 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 37 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()
```

Line: 12 Column: 12

Figure 37: 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 1: Executing a pathway simulation

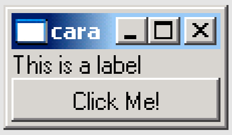
```
local nmr = spec.createExperiment(
  cara:getSpectrumType( "HNCA" ),
  cara:getResidueType( "ALA" ) )

local path = nmr:getPath( 1 )
```

```
for i,j in pairs( path ) do print( j ) end
-- Output:
HN
CA
N
```

Algorithm 1 shows again how to access the CARA API, this time to instantiate an NmrExperiment (see chapter 4.3 in [Keller 2004]) 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 2: 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 2 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 3: 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 3 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").

In the last chapter we saw how a user can present and edit dynamic attributes using the attributes dialog. The last example showed 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 34), the user would see it containing record reference an 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 34) and has full control about which attributes are visible in the dialog (see Figure 35) or only accessible by scripts.

Appendix A: A Brief Lua Overview

In this section I give a short overview over the most important concepts of the language and its libraries. For a full description please refer to the LRM.

Lua has straight forward syntax and semantics. It is a language with dynamic typing, which means you dont have to specify a data type for your variable as you might be used to in Pascal or Fortran. You create a variable by just assigning a value to a name.

```
valueA = 3
valueB = 3.5      -- valueB gets a floating point number
valueC = 314.16e-2
valueD = "this is a string"
valueE = true     -- a boolean value
valueF = nil
valueB = valueD   -- valueB now also has a string value

a, b, c = 10, 20, 30
-- multiple assignments can be done at once
-- a is now 10, b is 20 and c is 30
```

These variables, as soon they receive their value, are globally accessible to all scripts you might run. The value might be changed by later assignments and even receive a value of another data type. Variables carry their data type with them, transparent to the user. The example shows *basic types*, which are **number** (integer or floating point), **string**, **boolean** or **nil**.

Comments start with "--" and go to the end of the line. You can also write multi line comments using the following syntax:

```
--[[ this is a long comment
    going over more than one line ]]
```

Don't hesitate to write extensive comments, since they don't cost any performance and because experience shows that programs are written once but read many more times.

Besides handling scalar values and strings (i.e. basic types), Lua allows you to build data structures, as you might know them e.g. from Pascal as records and arrays. Lua offers so called *tables* as a central concept of data structuring. You can use table to represent both records and arrays. In contrast to basic types, tables have to be constructed before they can be used. The variable then contains a *reference* to the table (not the table itself).

```
tableA = {}          -- construct an empty table
tableA[ 1 ] = 3      -- index by number, like an array
tableA[ 2 ] = "a text"
tableA[ valueA ] = 3.4    -- index by value of a variable
(=3)
tableA[ "abc" ] = 3e-10   -- index by string, like a record
-- short form looks even more record like:
tableA.abc = "hello"

-- a constructor with initialization:
tableB = {[1] = 3,
    [2] = "a text",
    abc = "hello" }
tableA.sub = tableB       -- tables can be nested
tableA.sub.abc = "test"   -- accessing the nested table
```

In the above example tableB points to the same table as tableA.sub. So equality `tableB.abc == tableA.sub.abc` always holds.

Lua supports the usual arithmetic operators like "**+**" for addition, "**-**" for subtraction and negation, "*" for multiplication, "/" for division and also "**^**" for exponentiation. These operators work with numbers but also with strings, as far they can automatically be converted to numbers. Lua obeys the usual operator precedence (first "*" and "/", then "+" and "-", etc.), which can be changed by using brackets.

```
res = "3" + 4 * 2       -- res becomes 11
res = 2 ^ 3             -- res becomes 8
res = ( "3" + 4 ) * 2   -- res becomes 14
```

Lua offers the following relational operators: "**==**" for equality comparison and "**~=**" for unequality. There are also the usual "**<**", "**<=**", "**>**" and "**>=**" with the expected meaning. These operators all render a boolean value (**true** or **false**). Basic types are compared in the usual way. Tables are compared by their reference, not by the values they contain. If you want to compare tables by their contents, you have to write a comparison function iterating over the elements of the table yourself (see below).

Boolean values can be related using the logical operators "**and**", "**or**" and "**not**". Logical operators can also be applied to nil (with the meaning of false). The values of all other data types are considered true (the number 0 is also considered true!). The "or" operator returns the value of the first operand being true which is quite handy to select values.

```
res = 10 or "abc"     -- res becomes 10
res = nil or "abc"    -- res becomes "abc"
res = 10 and 20       -- res becomes 20
res = false and 20    -- res becomes false
```

In Lua you can define your own functions or use existing functions from a library (e.g. the math library specified in LRM). Functions are called as follows:

```
res = math.cos( math.pi ) -- res becomes -1
res = math.max( 1, 3, 4, 2 )  -- res becomes 4

-- functions can return multiple values:
l, o, s = spec.decomposeLabel( "CA-1" )
-- l becomes "CA"
-- o becomes -1
-- s becomes 1
```

Functions are objects themselves and can be assigned to variables or passed as arguments. That's why the following to variants of function declarations are equal:

```
add = function( a, b ) return a + b end   -- variant 1

function add( a, b ) return a + b end-- variant 2

res = add( 2, 3 )   -- res becomes 5
add2 = add
res = add2( 2, 3 )    -- res also becomes 5, it's the same
function

-- another function with a local variable:
function swap( a, b )
  local temp = a; -- local variables and parameter (a, b
  a = b           -- and temp) are only "alive" during
  b = temp        -- the execution of the function.
  return a, b     -- more than one return value is allowed
end
```

Since functions can be assigned to arbitrary variables, you can equip Lua tables with functions. With this possibility you can do "object oriented" programming. See the following examples:

```
bill = {}         -- create an empty table called "bill"

-- associate a function with field "sayHi":
bill.sayHi = function()
    print( "Hi!" )    -- of table "bill", see variant 1
end

-- this is syntactic sugar for writing
-- the same definition (see variant 2):
function bill.sayHi()
    print( "Hi!" )
end
```

```
bill.sayHi()        -- prints "Hi!" to the terminal

bill.age = 30       -- give bill an age

-- define another function:
function bill.printAge( me )
  print( "My age: "..me.age )
          -- ".." does string concatenation
end

bill.printAge( bill )
  -- prints "My age: 30" to the terminal

-- Lua offers syntactic sugar for the above case
function bill:printAge()
  print( "My age: "..self.age )
      -- self is an automaic variable provided by Lua
end

bill:printAge()
  -- prints "My age: 30" to the terminal
  -- also notice the use of the ":" instead of "."
```

The CARA/Lua API makes heavy use of the ":" syntax, since resulting code becomes much more readable.

Appendix B: 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.
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.

Function	*Gesture*
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.
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)

Function	*Gesture*
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 C: References

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

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

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.

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. (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. & 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

Keller, R. (2004). 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 (to be released).

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

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

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

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

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

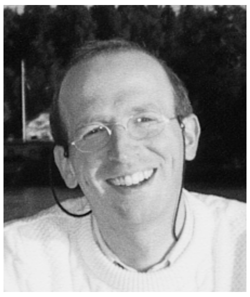
Wüthrich, K. (1986). NMR of Proteins and Nucleic Acids. Wiley, New York.

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 D: About the Author



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 the Ph.D. project "Optimizing the Process of Nuclear Magnetic Resonance Spectrum Analysis and Computer Aided Resonance Assignment" in the field of bioinformatics at the Institute of Molecular Biology and Biophysics at ETH Zurich with Prof. Dr. Kurt Wüthrich (Nobel Price 2002 for chemistry).

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.