SignalDraw: GUI Tool For Generating Pulse Sequences

Konstantin Berlin
Department of Computer Science
University of Maryland
College Park, MD 20742
kberlin@cs.umd.edu

December 9, 2005

Abstract

Generating pulse sequences for the NMR Spectrometer is one of the basic skills that a student of NMR Spectrometry must learn. One of the most popular formats for expressing the pulse sequences is the Bruker's signal format. This syntax is fairly complex, error prone, hard to visualize and requires basic programming skills to understand. That means that the students must spend a large portion of their studies learning a particular format that can become obsolete before they graduate. We propose a design and present a prototype of a visual manipulation application that will read, generate, and modify the Bruker's signal format in a manner that is intuitive and easy for beginning students yet powerful enough for more experienced or advanced users.

1 Introduction

Nuclear Magnetic Resonance (NMR) Spectroscopy is a common method for determining the structure of molecules. The method involves placing the molecule in a static magnetic field, exposing it to a second oscillating magnetic field, and then collecting and analyzing resulting data. The device that houses and controls the magnets is called an NMR Spectrometer. In order to get meaningful data from the NMR spectrometer, the magnets need to controlled by a very specific series of instructions, referred to as a pulse sequence.

The pulse sequence is a signal that consists of multiple channels. Each channel consists of a number of pulses that specify the type of magnetic pulse, the power at which it must be sent, and rotation angle (90° or 180°). In the event that no pulse needs to be sent on a particular channel, a delay is specified instead. The pulses on different channels are periodically synchronized in order to indicate that they must start executing at the same time.

One of the most popular formats for expressing pulse sequences is the Bruker's signal format. Creating a Bruker's pulse sequence file is a very laborious task, requiring in depth knowledge of the syntax, a good understanding of computer programming, and a heavy short term memory load. However, because pulse sequences are much easier to comprehend visually, this method can be significantly simplified through the use of a visual manipulation tool.

In this paper we present a partial prototype of a visual manipulation tool for drawing pulse sequences, called "SignalDraw", Figure 1. This tool could be used in conjunction with the NMR simulation tools, like Virtual NMR Spectrometer [2], in helping students learn about NMR. Figure 2 gives an example of a pulse sequence, and the resulting Bruker file.

SignalDraw is designed to be deployable in a research environment where uncommon computer architectures are prevalent and software is not fully up-to-date. SignalDraw is designed with four goals in mind:

- Have cross platform compatibility.
- Be well documented, and easily maintainable.
- Be easily configurable, and upgradeable.

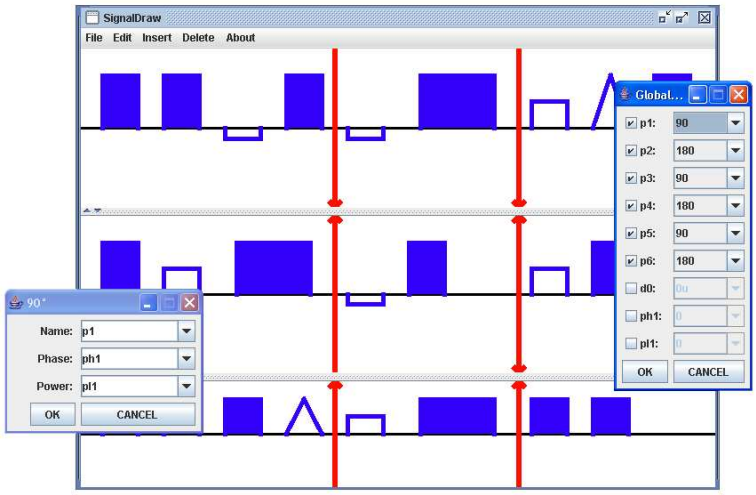
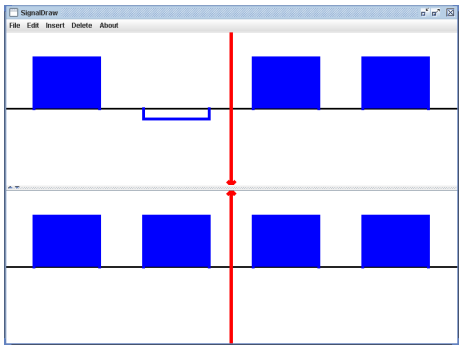


Figure 1: Screen capture of SignalDraw. The main screen, pulse property, and global properties windows are shown.

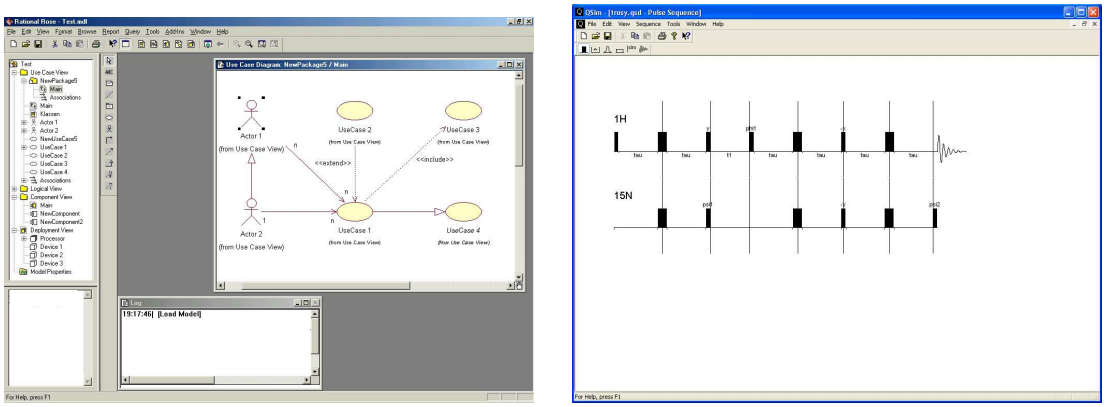


(a)

```
1: ;Autogenerated file created by SignalDraw
2:
3: "p1=90"
4: "p2=180"
5: "p3=90"
6: "p4=180"
7:
8: 1 ze
9: 2 (p1 ph1 d0):f1 (p3 ph1 p3 ph1):f2
10: 3 (p1 ph1 p1 ph1):f1 (p3 ph1 p3 ph1):f2
11: exit
```

(b)

Figure 2: A sample signal drawn in SignalDraw and the resulting output.



(a) Rational Rose (b) QSim

Figure 3: Screen captures of two visual design tools.

- Be fast and easy to use for novice, intermediate, and expert users.

Cross platform compatibility is critical for wide adaptation and usability of the program. A majority of labs are a mix of Linux, Unix, Windows, and MacOS systems. SignalDraw should deploy on all of the major platforms, and have a consistent look and feel to allow the user to seamlessly move between different computers without having to adjust to different layouts and changing interfaces.

2 Previous Work

Visual representations of information has been around for more then a decade in the software engineering community. Tools like Microsoft's Visual Basic [5] allows the user to draw and graphically specify GUI behavior without manually writing any code. The visual representation is then translated into a program that is further modified manually by the programmer.

Rational Rose [3], is a software tool that allows the user to graphically represent the architecture of an application. The representation is then transformed into code, written in a object oriented language like Java (See Figure 3a).

There are very few visual tools that are designed specifically for generating NMR pulse sequences. One example is Qsim [1], graphical user interface for the design of pulse sequences that is written in C++ (See Figure 3b). While it shares aspects with SignalDraw, several issues prevent its widespread use. It does not allow the user to save the pulse sequences in the Bruker format and instead saves the pulse in its own binary format. The user cannot use the saved file and proceed to use a different simulation tool to complete the analysis of the signal. QSim is not cross platform compatible, and would require significant coding effort and maintenance to become compatible with operating systems other than Windows. The user interface is very rudimentary, does not make use of the full screen area, and does not provide shortcuts for expert users.

3 Design

SignalDraw is written entirely in Java 1.4 and does not use any external libraries. Java 1.4 is the most widely deployed platform across different operating systems. Maturity and selection of software engineering tools for Java has no parallel compared to other languages. This is a good match to our requirements of maintaining cross platform compatibility, being easily maintainable, and being upgradeable.

To create a powerful and intuitive visual interface, we apply the Eight Golden Rules of Interface Design [4]:

- Strive for consistency
- Offer informative feedback

- Design dialogs to yield closure
- Support internal locus of control
- Reduce short-term memory load
- Enable frequent users to user shortcuts
- Permit easy reversal of actions
- Offer error prevention and simple error handling

3.1 Usage Cases

We present two prototypical students, Erin and Jack, to illustrate two typical usages of SignalDraw by students.

The first usage case is of Erin. Erin is a beginning student in biochemistry and needs to simulate a simple signal (two channels, one type of pulse, and one synchronization point). She has never used SignalDraw before. Erin opens SignalDraw and selects a "New" option from the File pulldown menu. She proceeds by adding two new channels using the "Add new channel" option in the "Add" pulldown menu. Erin then selects the type of pulse she wants to add from the "Add" menu, and proceeds to add the pulses to each channel by clicking on the spot she wants to add them.

To synchronize the pulses she right-clicks on the first pulse she wants to synchronize. A menu of possible options pops up, including an option to add the marker to the left or the right of the pulse. Erin chooses the left options. A red marker is drawn to the left of the signal. She then repeats the same steps for the second marker, on the second channel, with which she wants to synchronize.

Now that Erin has drawn the makers, she needs to synchronize them. She first right-clicks on the first marker and selects synchronization option. Immediately the cursor changes to a different pointer in order to indicate that she is in synchronization mode. Erin then clicks on the second marker that will be synchronized. The channels are redrawn such that the markers are drawn one under the other. Arrows are added to the markers to indicate that they are synchronized.

The second usage case if of Jack. Jack is an advanced student in biochemistry. Jack has a Bruker file that he needs to slightly adjust by erasing a single pulse. Jack opens SignalDraw and selects an "Open..." option from the pulldown menu. He is presented with a standard load file menu, from which he chooses his Bruker file. The file is read, and displayed in visual format of SignalDraw. He quickly finds the pulse that he wants to remove, and right clicks on it. He is presented with a pop options; to add a marker, check properties or delete. Jack selects to delete the pulse. He then goes to the pulldown and selects "Save As Bruker..." option. Jack is presented with a standard save file menu that he uses to save his file back in Bruker format.

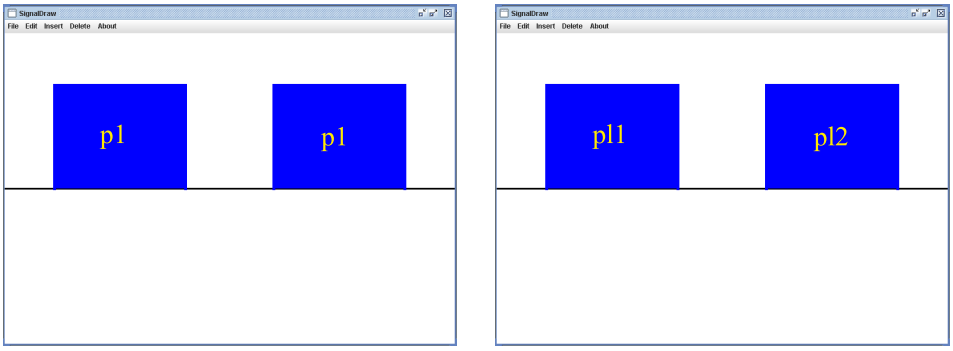
3.2 User Profiles

SignalDraw will target three different groups of users: novice, intermediate, and expert.

The novice user does not have a visual image of how the individual pulses are represented in the program, or the shortcuts available to him. They will use the drop down menu to select a pulse from the textual description. The drop down menu will also include an icon next to the textual description, in order for the user to start associating the icon with the pulse description.

To an intermediate user, the major slowdown in drawing signals is the menu selection of the pulse to draw. Selecting the pulse requires moving the mouse, opening the menu, reading the text description of the pulse, selecting the proper menu, and moving the mouse back to the location where the pulse should be inserted. The speed of the selection can be improved though the addition of a toolbar. The toolbar will be constantly displayed and the icon will visualize the pulse. The toolbar will offer users who have familiarized themselves with the icon representation faster drawing capabilities.

The expert user does not want to move the mouse, because it takes a large amount time to visually find the insertion spot again. Shortcuts will be introduced for all possible actions. For example, to change between different pulses during insertion the user will use CTRL-1, CTRL-2, etc. Other keyboard shortcuts will also be provided to quickly add channels, synchronizations, and save information.



(a) (b)

Figure 4: Displays pulse properties on the pulses. (a) Name property is displayed. (b) Power property is displayed.

Standardized shortcut keys such as CTRL-S for save, CTRL-Z for undo, and F1 will also be added to maintain consistency with other applications.

3.3 Reducing Short-term Memory Load

During the generation of a signal, a user needs to quickly visualize the power level, name, etc of pulses. The user would have to click on each individual pulse and memorize its power level. This requires a high short-term memory load, and is very slow.

In order to let the user quickly gauge the pulse sequences, we propose displaying the selected property on every pulse, as presented in Figure 4. The user can quickly detect any mistakes in the property values, and can immediately move the mouse over to the incorrect pulse.

4 Scaling of Pulses

The biggest usability issue is how to properly scale the pulses. Intuitively, it is easiest to visualize a pulse sequence when the pulses are drawn such that they are consistent with the actual timeline. Unfortunately, the pulses cannot be scaled relative to their interval, because length is not known ahead of time. Additionally, the delay pulses of small, or zero, length will no longer be visible or click-able.

We propose and evaluate two approaches. Both approaches are based on the concept we call “a fully synchronized window”. We will consider a fully synchronized window, a window of pulses and delays that lie between two markers that are synchronized between all channels. The first window is always fully synchronized on the left, and the last window always on the right, even if no markers are drawn. For example Figure 5d has one fully synchronized window, while Figure 5e has two. The size of each object (pulse or delay) is 1, or 2 if the pulse is 180°.

The first approach is to scale each synchronization interval, such that the pulses take up the full space between synchronization points, represented by Algorithm 1.

Algorithm 2 scales all pulses uniformly across all windows. This allows the user to visually detect longer pulses.

5 Further Work

SignalDraw is still in the early development phase. Shortcuts, and toolbars must be added to the interface. The Bruker translator has to handle changes in power levels. Gradient channel and different pulses need to be handled properly.

To improve the design and features of SignalDraw usage studies and surveys must be done to better gauge the needs of potential users. Extensive usability studies need to be done to figure out the optimal layout of the menus.

Algorithm 1 Scale independently

$I \leftarrow$ # fully synchronized windows;
$W \leftarrow$ width of the window;
for each channel **do**
  for each full window **do**
    $L \leftarrow$ aggregate of object sizes in window;
    for each pulse or delay in window **do**
      if pulse is 180° **then**
        object draw length = $2 * W/I/L$;
      else
        object draw length = $W/I/L$;
      end if
    end for
  end for
end for
return

Algorithm 2 Scale globally

$W \leftarrow$ width of the window;
$T \leftarrow 0$;
for each full window **do**
  $M \leftarrow 0$;
  for each channel **do**
    $L \leftarrow$ aggregate of object sizes in window;
    $M = max(M, L)$;
  end for
  $T \leftarrow T + M$
end for
for each channel **do**
  for each pulse or delay **do**
    if pulse is 180° **then**
      object draw length = $2 * W/T$;
    else
      object draw length = W/T;
    end if
  end for
end for
return

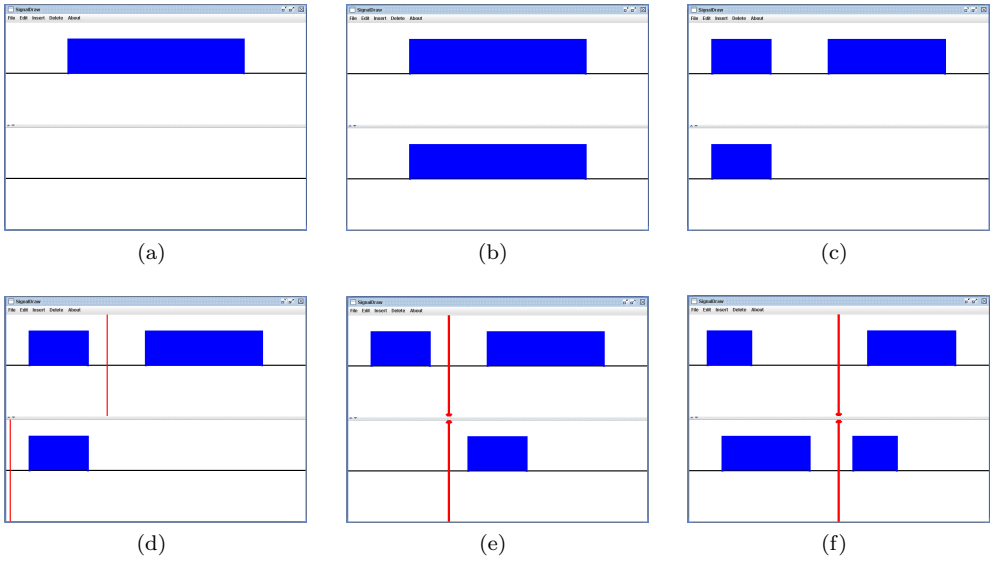


Figure 5: A sequence of pulse insertions and synchronizations: (a) 90° hard pulse is added to the first channel. (b) 90° hard pulse is added to the second channel. (c) 180° hard pulse is added to the first channel. (d) markers are added to both channels. (e) both markers are synchronized. (f) 180° hard pulse is added to the second channel.

SignalDraw is written in Java, so it is possible to set it up as an applet on the Internet, thereby Making it accessible to every student in the world with a web browser and an Internet connection. This would make it easy for the whole NMR community to use the tools without going through the hassle of trying to install it.

6 Conclusion

Currently generating pulse sequences for the NMR Spectrometer is a laborious task, that is better done through a visual interface. Just as the command prompt has been replaced with GUI interface in common computer use, so should the manual editing be replaced with a visual manipulation tool for manipulation of the Bruker pulse sequence file. Generating a pulse sequence is significantly easier and faster using a visual interface. SignalDraw provides such tool, in an easy and intuitive way.

References

[1] Helgstrand, M., and Allard, P. QSim, a program for NMR simulations. *J. Biomol. NMR. 30* (2005), 71–80.

[2] Nicholas, P., Fushman, D., Ruchinsky, V., and Cowburn, D. The Virtual NMR Spectrometer; a Computer Program for Efficient Simulation of NMR Experiments Involving Pulsed Field Gradients. *J. Mag. Res. 145* (2000), 262–275.

[3] *http://www-306.ibm.com/software/rational/.*

[4] Shneiderman, B. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[5] *http://msdn.microsoft.com/vbasic/.*