# QSIM: Installation Instructions

Daniel J. Clancy

May 29, 1998

## 1 Introduction

This document provides instruction for the installation of QSIM on various systems. The main section describes the installation of QSIM on a Unix host. Here at UT, QSIM has been tested on Allegro Common Lisp and Lucid Common Lisp under Unix. Installation on either of these lisps should be straight forward. QSIM should work on most lisps that run under Unix although some minor modifications may be required.

In particular, the default QSIM installation uses a common lisp defsystem and logical pathname facility obtained from CMU. It is our experience when installing QSIM on a different lisp, these two packages cause most of the problems. For this reason, we have included instructions for installing QSIM without the defsystem or logical pathname facilities. The main drawback of this option is that the installation will not support maintaining separate binary directories for different architectures (e.g. SunOs4 vs Solaris).

In addition, QSIM has been ported to the MacIntosh, to Linux and to the PC. On the Mac, it has been tested on MacIntosh Common Lisp 3.9, on Linux Allegro CL 4.3 and on Windows Allegro CL. Displaying the results when running on a PC under Windows is still less than optimal. This is described later in this document. In case you are not aware, currently Allegro CL for Linux can be obtained from Allegro at no charge. Unfortunately, the Linux port for CLISP was performed outside of our research group and we have been unable to integrate and test these changes. Thus, for Linux installation please refer to our web site and obtain a copy of the file linux-installation.txt which contains instructions pertaining to a Linux installation for CLISP.

Finally, a section is provided describing the installation of the POS display facility. This facility is used to display the results of a QISM simulation.

Thus, there are five sections to this document:

1. Simple Installation (without using Common Lisp Defsystem).

2. Installation on a Unix Host using the Common Lisp Defsystem.

3. Installation on a Unix Host using lisps *other than* Allegro CL and Lucid

4. Using QSIM under Windows on a PC.
5. Installation of POS
6. Installation on a MacIntosh

## 2 Simple Install

The installation is QSIM is not much simpler than it has been in the past. We are still working out some of the kinks so if you have any problems please contact us. This option is designed for people that will be installing QSIM on a small number of architectures and Lisp versions and who will not be modifying the baseline and recompiling QSIM on a regular basis. I suspect that this is a vast majority of the users. This option does not take advantage of the Common Lisp Defsystem and thus when recompiling QSIM all of the source files must be recompiled and the binary files will be stored in the same directory as the source.

1. Unpack the tar file contained on the tape. This can be performed with the following command:

   ```
   zcat beta.tar.Z | tar xvf -
   ```

2. Modify the definition of ***qsim-root*** in the file `start-nq-no-defsystem.lisp` in the root QSIM directory.
3. Compile the c-code to run xbatch. This is described in more detail in the write-up on the POS system modifications for the Unix environment. For this installation, place the compiled program (it is called **xbatch**) in the **xpos** directory. (**This is only required when running under Unix. For Windows or the Mac, POS is not used.**)
4. Compile QSIM by loading the file **start-nq-no-defsystem.lisp** and executing **(compile-qvsim)** from the `user` package.
5. Test the installation by entering the `qsim` package (i.e. type in `(in-package :qsim)` and then execute the function `(q)` and select some models to run.
6. The installation is now complete. In the future, QSIM can be loaded by loading **start-nq-no-defsystem.lisp** and executing `(load-qsim)`.

## 3 Installing QSIM using the Common Lisp Defsystem

This section provides information about installing QSIM on a UNIX host using the common lisp defsystem.

1. Unpack the tar file contained on the tape. This can be performed with the following command:

   ```
   zcat beta.tar.Z | tar xvf -
   ```

2. A number of pathnames must be modified to reflect the location of various pieces of code at your installation site. Whenever possible we have lexically bound a special variable ***qsim-root*** to simplify the changes possible. The pathname for the directory in which the qsim hierarchy has been stored should be assigned to this variable wherever a binding is made. When this special variable is used, it must be set to a pathname without the trailing "/". The declarations for this special variable must be modified in the following files:

   start-nq.lisp
   clisp/defsystem-patches.lisp
   xpos/xpos-system.lisp

   This special variable should be set to the directory that contains the subdirectories `nq` and `xpos`.

3. QSIM allows binary files to be located in a separate directory from the source files. This feature is helpful if QSIM is run on multiple platforms or with different versions of lisp. For each platform/lisp combination, a separate binary directory is used.

   If you wish to create a separate directory for the binary files, then follow the first option below. Otherwise, follow the instructions in the second option and the binary files will be stored along with the source.

   **Creating Binary Directories** The QSIM binary directories are stored in nq/bins and xpos/bins. A binary directory must be created for each platform/software configuration in each of these directories.

   To determine the name of the subdirectory, load the file **start-nq.lisp**. Respond no to the questions asked about loading and compiling QSIM. Then call the function **(mk::get-bin-dir-name)** from the lisp prompt. This function will return the name of the subdirectory for the binaries for the machine that you are currently using. (e.g. If you are on a Sun 4 using Lucid 4.1 the directory is named sun4_4.1.) Create a directory by this name under nq/bins and xpos/bins.

   **No Binary Directories** If you do not want to use the binary directory feature, then define the special variable user::*no-bin-dirs* with a value of T. Currently, there is a declaration in start-nq.lisp that can be uncommented that will define this variable appropriately.

4. Compile the c-code to run xbatch. This is described in more detail in the write-up on the POS system modifications for the Unix environment.

5. Compile QSIM by loading the file **start-nq.lisp** and answering no when it asks if QSIM should be loaded and yes when it asks if QSIM should be compiled. Once start-nq.lisp has been loaded, qsim can be completely recompiled (if for some reason you need to do this) by executing (mk:compile-system 'nq :force :all).

6. The installation is now complete. In the future, QSIM can be loaded by loading **start-nq.lisp**.

### 3.1 Creating a Lucid Disk Image

Included in the distribution is a copy of our script file which we use to create a lucid lisp image containing QSIM. Using a lisp image greatly reduces the time required to load QSIM. If you wish to create a disk image of QSIM, modify the definition of the variable QR-DIR in the file **make-ne-image**. This script can be executed to create an nq image called **nq_x_lcl4.1** in the nq directory.

## 4 Installing QSIM on a Non-supported Common Lisp

QSIM has been developed using standard Common Lisp. As a result, it can usually be ported to most versions of Common Lisp with a minimal amount of effort. Currently, the distribution provided has only been debugged for Allegro CL and Lucid CL. It also may work fine for Harlequin Lisp. The changes for harlequin were made by somebody outside of our research group so I do not know what type of problems you might encounter. If you have questions or problems, please send mail to clancy@cs.utexas.edu.

The following suggestions should help you install QSIM using a different version of lisp.

- Most of the problems occur within the Common Lisp Defsystem and Logical Pathname code. This code was obtained from CMU. Documentation for the code is included in the c-lisp directory within this distribution. The code itself is well documented.

  I recommend that you try to install QSIM without these facilities. Look at the file `start-nq-no-defsystem.lisp` which should be in the `nq` directory and change the declaration of the special variables at the beginning of the file and then load the file.

  Then run the function:

  ```
  (compile-qsim)
  ```

In the future, QSIM can be loaded by loading the same file and running `(load-qsim)`. The file that actually has the load statements is `nq/source/main/nq-no-defsystem`.

If you run into problems doing this here are some suggestions to help you through them:

  - **An error occurs when compiling a file.** You might want to simply try and load the non-binary version of the file and see if you can get through the compilation process. A number of facilities are provided within the distribution that you will probably never use. Sometimes, these may cause problems in installation. By loading the lisp version, you avoid the compiler preventing you from getting everything else from running.
  - **Problems with undefined special variables.** Some lisps load after compiling and some do not. Also, some Lisps will complain if they encounter a special variable that is not initialized. This happens if it is declared in a preceding file and it does not have an `(eval-when (compile) ...` around it. If you modify the file that compiles the source files so that it loads some of the files immediately after compiling them, then this problem can be avoided. You can find out which file is causing the problem by doing a `grep` for the offending special variable.
  - If you keep running into problems, you might want to try and load the Lisp source files first and see if you can get things running.
  - If you get a continuable error, continue and see what happens.

- If you wish to use the Defsystem and logical pathname code, then attempt to get these components running independent of QSIM using an example defsystem that is included within the distribution. It is in the c-lisp directory. Once the Defsystem code is running you should not have much difficulty.

- You will not be able to use the "compile-qsim" and "load-qsim" functions if you are using a different lisp. These functions simply call common lisp defsystem functions. Look at the declarations of these functions in clisp/defsystem-patches.lisp.

- Ignore warnings when compiling QSIM. I would only be concerned with error messages.

- The function **qplot-end-display** in the file **xpos/qplot-primitives** will need to be modified. To display the results of a simulation, the POS package writes to a file and then forks a Unix process that executes the xbatch program. This program displays the results which are stored in this file. Qplot-end-display makes the lisp call that forks this process. Currently, the code will work for lucid, allegro and kcl. A new line with a read-time conditional will need to be added to this function for any new lisps that are used.

- If you have problems getting the function call to xbatch to work, a temporary work around is to route the output to a postscript file (select `CH` from the qsim-display menu) and then manually display this file using ghostview.

# 5 Running QSIM on a PC under Windows

Recently, we have ported QSIM to Windows running Allegro CL. Unfortunately, we have not had the resources required to modify the code that is used to display the results of a QSIM simulation for Windows. (Currently, we use an X-windows program to display the results under Unix.) The good news is that there is a work around. While the work around is less than ideal, it should be adequate. To display the results, you will need to obtain a copy of **`ghostview`** for the PC. Ghostview is a graphical front-end to the ghostscript program. You need to obtain and install both programs. These programs can be obtained from

**`ftp://ftp.cdrom.com/pub/simtelnet/win95/print/`**

I believe that you need to obtain all of the following files: gs403fn1.zip, gs403ini.zip, gs403w32.zip, gsview21.zip.

Once ghostview is installed, you can display the results by routing them to a postscript file. The version for the PC has already been set-up to force this to happen. When entering the display program (**`qsim-display`**). You will be prompted for a file name that will be used to store the displays. After you select the information that you wish to see (e.g. the behavior tree and a variety of behavior plots), you must exit **`qsim-display`** by selecting **`Q`** and then you can look at the results by starting ghostview and viewing the file in which the results were stored. Unfortunately, at this point you need to view the pages sequentially (i.e. starting at page 1 and then moving through the page numbers). You are currently unable to jump around as you normally are able to do with ghostview and a postscript file. (This is because the postscript file does not have certain markers required to navigate in this manner.) If you wish to view additional plots, then you can restart **`qsim-display`**, send them to the same file, and then refresh the display within the ghostview program.

**Potential improvements** If you are extensively using QSIM on the PC, it might be worthwhile to invest a little bit of time in one of the following improvements. Due to time restrictions we have been unable to pursue these improvements, however, we would gladly support your efforts.

**Porting POS to Windows 95** – This is probably about a weeks worth of work if you are experienced with LISP and also with the PC. Basically, there are a set of "atomic" functions that write create a display and generate certain graphical structures (like boxes, lines, text, etc) that need to be written for the PC. This is much more work than the other potential improvements discussed below.

**Modify the form of the postscript file so that you can freely navigate the file.** As mentioned above, you currently are unable to freely navigate the postscript file. This is because there are a few page markers that are missing. I spent a little bit of time trying to figure this out and was unable to. However, I suspect that it is fairly simple and with a good postscript book and our help this would probably only require about 1 days work or less.

**Automatically fork a ghostview process** – It would also be possible to have lisp automatically fork a ghostview process for each plot that it displays. To do this, it would open a file, write to it, close the file and then fork the ghostview process. It would do this for each plot thus providing the same level of interactivity provided with an X-windows environment.

**Allow viewing of the file while `qsim-display` is still writing to it.** – Currently, ghostview will not display any portion of the file while the Lisp process has the file open. I do not know if this is possible, but it would be nice if ghostview could display the portion of the file that has been written while the Lisp process still maintained the file as open. This would eliminate the need to leave `qsim-display` prior to viewing the results.

Please contact us if you make any improvements or encounter any problems.

# 6 Description of the POS Interface Modifications for X-Windows

POS is the graphical interface to QSIM. It provides a set of primitive plotting functions which are used to display the results of a simulation. Currently it runs under the Symbolics, Explorers and X-Windows. xPOS is the latest version of POS. This section is designed to provide a description of the modifications made to POS so that it is compatible with X-windows. For more information about POS, please refer to the POS User's Manual.

The interface between POS and X-windows occurs via a C program. The POS code places information about the plots to be drawn in a temporary file. This file is then read by the C program which interfaces with X-windows and creates the desired window. The interface between C and X-windows is a stable interface which is well documented. Conversely, the Lisp interface to X-windows

(CLX) is less-stable and poorly documented. Use of a C program for this interface should simplify maintenance and reduce the system dependencies.

## 6.1 Installation on a Unix Host

To get the POS interface working, you must first compile the C program Xbatch for the machine that is being used. If you are running QSIM on multiple Unix based machines (i.e. non-lisp machines), then you must compile a version of Xbatch for each machine.

**Modifications to the Makefile** In the directory xpos/xbatch-dir there is a file called Makefile. This file is used when compiling the C program. A number of changes must be made to this file to customize it for your particular site.

1. The following two lines in the Makefile tell the C program where various library and include files can be found. In particular, it must be able to locate the X11 interface library. These must be changed to indicate the location of these libraries and include files on your system.

   ```
   LIBS = -L /lusr/lib/X11 -lXext -lX11
   INCLUDES = -I/usr/include -I/lusr/include
   ```

2. The following line must be changed to point to the location of the directory containing xbatch.c and xbatch.h. Both of these files are in xpos/xbatch-dir. The root for this path on your system must be modified.

   ```
   CDIR = /v/ai/v5/qr/xpos/xbatch-dir
   ```

**Compiling Xbatch** Once the Makefile has been modified, you are now ready to compile xbatch for your machine.

1. From within the xbatch-dir directory, type "make xbatch". This will compile the xbatch file for the machine that you are currently on. The binary file which results from this compilation is called xbatch.

2. Copy the binary file xbatch into the binary directory for this machine. In an earlier section, you were instructed to create binary directories under xpos/bins and nq/bins. This file should be copied into the binary directory for this machine under xpos/bins. For example, if you were in a Sun 4 using Lucid 3.0 the directory would be sun4_3.0.

## 6.2 Testing Xbatch and Trouble Shooting

To test the xbatch program to ensure that this interface is working:

1. Type "make xbatch" from within the xbatch-dir.
2. type "xbatch -f xbatch-test.fle" from within the same directory. A window should be created with various objects and the text "QSIM lives" on the screen. If this window is not created, then there is a problem with the interface between the C program and X-windows.

If the POS interface is not working properly (i.e. a window is not created with the POS output), then the following steps will help isolate the problem. Informing us of the results of these steps will help us isolate the problem at your site.

1. Test xbatch as described above.
2. Check to make sure that there is a directory defined for the machine containing the binary file xbatch. If this file does not exist, an error may be returned.
3. Within the lisp environment, perform a trace on qplot-end-display and then attempt to display a plot. The C program is called from within this function. Qplot-end-display is in nq/xpos/qplot-primitives.lisp.
4. If you are using Lucid, perform a trace on the function user::shell. This is the call to the C program xbatch. Shell is not a Common Lisp function. If you are using a lisp other than Lucid, then this line in the code must be changed.

## 6.3 Overview of Changes

This section is only relevant to people who are familiar with the details of the POS package.

**Xbatch** Under X Windows, POS uses the C program Xbatch to draw all its graphics. The function qplot-behavior in qplot-primitives.lisp opens up a text file under usr/tmp. All succeeding drawing commands write out instructions to that file. (e.g. (qplot-line 10 10 100 100) would cause the following line to be written: drawline 10 10 100 100) When all drawing is done, the user calls the function qplot-end-behavior. This function closes the text file and then makes a system call to xbatch with the name of the text file as a parameter. Upon being called, Xbatch parses its input file and displays the specified plots in a newly created window. Thus, the graphics are displayed by a process which is forked by the lisp code in POS. As a result, as long as the machine running qsim supports X

Windows, the system call to Xbatch will be the only line of code that has to be changed when porting POS to another Lisp on Unix.

**Fonts** All functions in POS will accept font information in the same format as before. (e.g. If you are running on the symbolics, it will take symbolics. If you are on the explorers, it will take explorers fonts. For X Windows, the font keyword can be supplied with a string.) However, the symbolics format for fonts can be used to specify a font on any machine. This font format is specified in the POS manual and will be supported on all platforms in the future.

**Colors** You can either specify the alu argument in the old manner where each machine had its own color scheme or you can specify the color as a string. Currently all X Windows colors are supported for POS running under X Windows. However, the code for Symbolics and the Explorers is in the process of being updated. Specifying the color as a string will be supported on all platforms in the future. The variable ps::*colors* is a list of some of the more common colors available.

# 7 Installing QSIM on the MAC

The installation instructions focus on installing QSIM once the files have been transfered to your hard drive. All of the required files are supplied in our standard distribution of QSIM. If you do not have Tar or Compress for the MacIntosh, it is our experience that it is easiest to uncompress and untar the QSIM distribution on a Unix machine and then download the "qr" directory to your MacIntosh.

## 7.1 Customizing and Compiling QSIM

**Customizing QSIM** – Modifications to be made to the following files:

- start-nq-mac-mcl3.lisp

The definitions of the special variables *qsim-root* and *lisp-root* must be modified to reflect the name of your hard disk drive and the location of lisp and QSIM. If

These special variables are defined in the cl-user package and are referenced by the following files:

- sys-site/nq.system
- sys-site/xpos.system
- nq/source/main/cl-nq-system.lisp
- xpos/xpos-system.lisp

If more flexibility is required when loading QSIM, then changes may be required in these files as well.

**Compiling QSIM** – Load the file `start-nq-mac-mcl.lisp` into Lisp to load the functions to compile QSIM. and then execute the function

```
(compile-all-of-qsim)
```

**Loading QSIM** – Once QSIM has been compiled, QSIM can be loaded by loading the file `start-nq-mac-mcl.lisp` and then executing the function:

```
(load-all-of-qsim)
```

# 8 Using DOT for a graphical output

## 8.1 What is dot?

Dot is a program that performs graph layout and can generate an X-windows based display of a graph from a text input file. Various extensions have been implemented in QSIM that make use of this capability. These extensions have been developed for our private use at UT, but we are distributing them due to their usefulness. Currently, Dot is used for the following two purposes:

1. To generate a graphical description of a QDE along with qualitative value information for a specified state if desired.

2. To generate a graphical "explanation" of why there does not exists a successor that matches a specified template for a given state. This is called *successor explanation* and it is described in the qsim-extensions document.

## 8.2 How do I get a copy of dot?

Dot is distributed by AT&T Bell labs and is available for restricted use. It can be obtained via anonymous ftp or via WWW access. Dot is actually included in a package of four graph visualization tools. The other tools are called dotty, neato and tcldot. You can obtain a non-commercial license for dot/dotty/neato/tcldot and download the software from the Web page:

**http://www.research.att.com/sw/tools/graphviz**

Choose source or binary (executables are available for some common Unix machines and for Microsoft Windows). If you accept the non-commercial license, download the 'graphviz' package.

Our experience with dot is very limited. Thus, we will not be able to help you install it. Hopefully, installation should be fairly straight forward.

### 8.3 How do I get QSIM to work with dot?

To get QSIM to work with dot, you need to modify a couple of lines in the file

**qr/nq/source/interface/qsim-dot.lisp**

Dot generates a postscript file and then **ghostview** is called to display the postscript file. To use dot you need to change the definition of the following two variables within this file:

**+DOT-COMMAND+**
**+GS-COMMAND+**

Currently, both of these command specify a full path. If you have ghostview and dot in your path, then you can simply change them to the function name.

Also, if you are using a listp other than Allegro CL or Lucid CL, then you might need to change the line that makes the system call just as you had to do for POS. Look in the file with the `dot` code and search for `user::shell` to find where to do this. It might work right now for kcl and lispworks as well.

Once you have done this, the rest should work fine. For information about displaying the QDE using dot, please see the qsim-extensions.ps manual under the Additional Display Capabilities section.

### 8.4 If an explanation looks wrong

The successor explanation capability uses the QSIM constraint filtering algorithm when making inferences. To support this, a number of changes were required to the functions that handle individual constraints. While successor explanation has been tested, it has only been in use for a short period of time. Thus, it is possble that certain constraints have not been adequately tested. This is especially true for seldom used constraints. The only type of problems that might result are of the type where a constraint is identified as violated, but the values specified are such that the constraint should not have been violated. If you encountered a response that you are unsure of, please contact clancy@cs.utexas.edu with the model to see if there might be a problem.

I hope that these instructions are helpful. We will provide whatever help we can.

## 9 Conclusions

Thank you for your interest in QSIM. If you have any questions, please send mail to clancy@cs.utexas.edu or kuipers@cs.utexas.edu. Bugs should be reported to qsim-bugs@cs.utexas.edu.