3. Materials and Methods.

Code was modified from SPASS (version 3.0c) available from Max Planck Institut für Informatik in Saarbrücken at `http://spass.mpi-inf.mpg.de/download/sources/spass30.tgz`. The major modifications made were concentrated in source files *eml.c* and *eml.h*. Other modifications were purposely restricted to a minimum, but were unavoidable in source files *flags.c*, *flags.h*, *symbol.c*, *symbol.h*, *top.c*, *dfgscanner.l*, and *dfgparser.y*. In total more than 7850 lines of code and comments were added (extending the SPASS code base by 8.7%). Limited documentation for SPASS can be found in the archive `spass30.tgz`, in the files *handbook-spass.pdf*, *spass-input-syntax.pdf*, *tutorial.pdf*. The code obtained from `http://spass.mpi-inf.mpg.de/` is referred to in this study as SPASS. The code developed in this study is referred to as extended-SPASS. Occasional mention is made of MSPASS. This version included code for translation of modal and description logics into first-order logic by semantic translation. From version 3.0, SPASS and MSPASS have merged.

C code was developed in Eclipse Europa on Mac OSX 10.5 for Intel, Mac OSX 10.3 for PowerPC, on Windows 2000 Pro and on Windows XP Pro (sometimes hosted on VMWare Fusion under OSX 10.5, `http://www.vmware.com`). (Eclipse Europa version 3.3 from `http://www.eclise.org/` with C/C++ development tools from `http://www.eclise.org/cdt/`). SPASS code was imported into Europa as a pre-existing C project using the *Makefile* that was included with the SPASS sources for version 2.2 (`spass22.tgz` from `http://spass.mpi-inf.mpg.de/`). For release, modified code was added to SPASS 3.0c code, and compiled using the SPASS3.0c GNU *configure* script to create a system specific *Makefile*. An understanding of C was obtained from *'C Programming: A Modern Approach'* by King [2], and [4].

C code was compiled with gcc 4.0 on Mac OSX 10.5, with gcc 3.3 on Mac OSX 10.3, with gcc 3.4 on Windows 2000 Pro/XP Pro, with gcc 4.1 on Fedora Core Linux (i686 kernel 2.6.20-1.2316.fc5), and with gcc 2.95 on Solaris 5.9. Unix-like capabilities were provided by 'Cygwin' on Windows 2000 Pro/XP Pro (`http://www.cygwin.com/`, version 1.5.24 for i686-pc). Typical gcc compilation options used during development were *-ansi -pedantic -Wall -Wshadow -Wpointer-arith -Wwrite-strings -O2 –Winline*. GNU Make versions were, 3.81 on Mac OSX 10.5, version 3.79 on Mac OSX 10.3, version 3.81 for i686-pc-cygwin on Windows 2000 Pro/XP Pro, version 3.80 on Fedora Core Linux, and 3.79 on Solaris 5.9. Additional compilation was performed (as defined in the pre-existing *Makefile*) by bison (GNU Bison 2.3 parser generator) and flex (fast lexical analyzer generator 2.5) on Mac OSX 10.5; bison 1.28 and

flex 2.5 on Mac OSX 10.3; bison version 2.3 for i686-pc-cygwin and flex version 2.5 for i686-pc-cygwin on Windows 2000 Pro/XP Pro; bison version 2.1 and flex version 2.5 on Fedora Core Linux; and bison version 1.28 and flex version 2.5 on Solaris 5.9.

The web-based user interface was developed using Mozzila Firefox version 2.0 on Mac OSX and Windows 2000 Pro (`http://en-GB.www.mozilla.com/en-GB/firefox/`). The Firefox extension Firebug (version 1.05) was used during development to aid debugging of JavaScript, CSS and HTML (`https://addons.mozilla.org/en-US/firefox/`). HTML, CSS, and JavaScript code was edited using the oXygen XML editor (`http://www.oxygenxml.com`), and TextWrangler 2.3 (`http://www.barebones.com`). An understanding of JavaScript and CSS was obtained from *'JavaScript and Ajax'* by Negrino and Smith [3]. The small amount of Ajax used in the web-based interface was modified from code presented in [3]. Images organized by the CSS code were built and colored manually using PowerPoint for Office 97 on Windows 2000 (for example, the .png images used for rounded corners on tabs of the interface). The 'Help' tab in the user interface is intended to be used in conjunction with an independent MediaWiki web site. It was tested using MediaWiki 1.11 / PHP 5.2 / Postgresql 8.2 on OSX 10.5 (`http://www.mediawiki.org`), and using a demonstration page at `http://moodle.cs.man.ac.uk/mscwiki/index.php/Media_Handling`. An Apache HTTP web server was used during development (version 2.2 on both Mac OSX and on Windows 2000 Pro). A Perl cgi script was used to provide the interface between the SPASS executable and the web-based user interface (Perl version 5.8 on OSX 10.5, on Windows 2000 Pro and on Solaris 5.9). This Perl script was modified from the script driving the MSPASS interface at `http://www.cs.man.ac.uk/schmidt-bin/web-mspass.cgi`. The basic idea for the layout of the web-based interface (but *none* of the code or other materials) was developed from material in the course '*Interactive System Design COMP60172'* (Manchester 2007). Assessed material for the social networking interface designed during this course is documented at `http://moodle.cs.man.ac.uk/mscwiki/index.php/Introduction`.

The modified SPASS application is deployed on the Manchester student web server `n9plus.cs.man.ac.uk` (Solaris 5.9), and can be seen at `http://project.kjsmith.net/index.html`. The code can also be downloaded here, together with instructions for compilation and testing, the text of this dissertation, and other supplementary materials. (An alternative web address is `http://www2.cs.man.ac.uk/~smithk/`).

Test routines were developed in shell script (C shell and Bourne shell), and using the standard shell tools *egrep* / *grep*, *awk, tr, sed*, etc. Output from the extended-SPASS was checked both manually, and against output of a Prolog routine developed as described in [1, ml2dfg]. This `ml2dfg` code provided by Dr. R. Schmidt was executed on SWI-Prolog on OSX 10.5 and on Fedora Core Linux (version 5.6, `http://www.swi-prolog.org`).

A duplicate axiomatic translation system was developed in Java, and formulated as a test system. Java code was developed in NetBeans 6.1 (`http://www.netbeans.org/`) on OSX 10.5, using code compatible with Java 1.4 and above. A simple Java dfg-parser was developed using JavaCC 4.0 (`https://javacc.dev.java.net/`). UML design was performed in the NetBeans UML module. Diagrams were exported from this UML module as svg-format graphics, and converted to Adobe acrobat format using the Safari 3 browser on OSX 10.5.

Timing tests were executed on OSX 10.5 using a dual processor Mac, each processor a 1.83 GHz Intel Core 2 Duo, with 1GB SDRAM per processor.

Fonts used for documentation were *Zed* (from `http://www.cs.kent.ac.uk/people/staff/rej/Zedfont/latest/`) and *LicsX* (from `http://lucacardelli.name/Fonts.htm`). These are suitable for use with Microsoft Word for Mac X. PDF files were produced using the native routines for 'printing to PDF' in OSX 10.5.