Bill J. Ellis
February 24, 2005

Porting SPADE-PC to SICSTUS (Reissue)

Contents

[*]Hums are short notes intended for distribution between those involved with EPSRC grant GR/T12289/01. Hums describe ϵ-baked ideas, where $1 \geq \epsilon \geq 0$. (Hum refers to both the Praxis Humming bird and Winnie-the-Pooh's indirect approach to writing: "Poetry and Hums aren't things which you get, they're things which get you.").

1 Introduction (Prelude)

This is a reissue of CS-note 33, produced as part of the NuSPADE project. It describes and documents changes made to the SPADE-PROOF-CHECKER in porting from POPLOG-PROLOG to SICSTUS. As the SPADE-PROOF-CHECKER and the SIMPLIFIER share several common aspects, the details of this port may prove beneficial for porting both the SPADE-PROOF-CHECKER and the SIMPLIFIER.

Note that this port was never used as part of the NuSPADE system. It was found to be more straight forward to develop NuSPADE as a separate system that communicated with the SPADE-PROOF-CHECKER, rather than embed NuSPADE within the SPADE-PROOF-CHECKER. Consequently, the SPADE-PROOF-CHECKER was treated as a black box, and thus the POPLOG-PROLOG version of the SPADE-PROOF-CHECKER was sufficient.

The interpretation of a verification condition within the SPADE-PROOF-CHECKER is affected by operator precedence. This operator precedence varies between the default POPLOG-PROLOG behaviour, the default SICSTUS behaviour, and (in a few cases) the SPADE-PROOF-CHECKER behaviour. Rationalising these operator precedences across the different systems was necessary for the correct operation of NuSPADE. Thus, this aspect of the port was continued in NuSPADE and developed beyond the initial version seen in this early port. The most up to date version of this file is taken from the PropGen system, and is inserted in a new section 7.

2 Introduction

As discussed in CS-note 14, there are numerous issues with POPLOG-PROLOG which led to a desire to port SPADE-PC$_{\text{POPLOG}}$[1] to SICSTUS. A working version of this port has been completed. This note serves to describe the limitations of the port and document all of the changes made to SPADE-PC$_{\text{POPLOG}}$ in achieving the port.

3 Limitations

The process of performing the port was made fairly straight forward by their only being a few occurrences of POPLOG-PROLOG specific features inside SPADE-PC$_{\text{POPLOG}}$. There does however remain some differences between the two systems.

- **Cosmetic differences -** SICSTUS uses a different pretty-printing format than POPLOG-PROLOG. This is most apparent in the printing of hypothesis and conclusions, which are not formatted as neatly as in POPLOG-PROLOG. This might be fixed by writing a new pretty printing routine from scratch or appropriately integrating with the more sophisticated printing routines in SICSTUS.

  There is also a minor difference in the command prompt. In reaction to the behaviour of the SICSTUS read predicate[2], it is neater to take a newline after each question. By targeting the core interaction predicates, this is achieved with relatively few changes.

- **Licence support -** SPADE-PC$_{\text{POPLOG}}$ makes calls to an external library called FLEXLM to enforce licencing. This feature is unlikely to ever be important for NUSPADE, and shall not be ported in SPADE-PC$_{\text{SICSTUS}}$. It should be noted however that SICSTUS has a strong C interface, which should allow this to be achieved if ever desired.

- **Exceptions -** SPADE-PC$_{\text{POPLOG}}$ captures POPLOG-PROLOG exceptions, returns to a safe state, and continues. As SICSTUS has different exceptions it is very difficult to repeat this behaviour in SPADE-PC$_{\text{SICSTUS}}$. However, a good approximation is achieved using the SICSTUS exceptions in a similar manner.

  Of most notable difference is the behaviour of `CTRL-C`. This now displays SICSTUS debugging options rather than raising an exception and internally resetting to a safe state. The exception raising behaviour is useful, in particular for aborting the current line of input. Unfortunately, SICSTUS only seems to support such signal handling through some rather fiddly external system programming. As such this remains a missing feature[3].

- **Bugs -** Currently the port does not support the rewriting (replace) of subterms while these subterms are quantified. In SPADE-PC$_{\text{POPLOG}}$ this is implemented by un-wrapping, performing the rewrite and re-wrapping. The correctness of this is enforced by introducing dollar annotated variables for the quantified variables. Unfortunately, this code behaves differently in SPADE-PC$_{\text{SICSTUS}}$[4], and is not straight forward to fix. This is a fairly low priority issue, but does highlight a very genuine limitation of the port as it stands.

4 Execution

SPADE-PC$_{\text{SICSTUS}}$ can be run both as a development and release system.

4.1 Development

The development system includes the full Prolog environment, including support for debugging. The system is started by running SICSTUS, consulting the files of SPADE-PC$_{\text{SICSTUS}}$ and calling the predicate `startup_sequence`. This can be automated in a small shell script:

```
#!/bin/bash

#Create small file to start the program.
echo ":-consult(checker)." > startcode.pl
```

[1] The key components of NUSPADE are now referred to as SPADE-PROOF-CHECKER (SPADE-PC) and SPADE-PROOF-PLANNER (SPADE-PP). Thus: NUSPADE = SPADE-PP + SPADE-PC. In this particular note it is helpful to distinguish between the POPLOG-PROLOG version of SPADE-PC (SPADE-PC$_{\text{POPLOG}}$) and the SICSTUS port of SPADE-PC (SPADE-PC$_{\text{SICSTUS}}$).

[2] Which hides the `|:` prompt if used on a line where there are already some existing characters. (Note that this behaviour is as observed and not as documented. It may be a bug.)

[3] A possible solution is a top level C wrapper, responding to signals. This seems untidy. It might also be possible through TCL/TK.

[4] It introduces the dollar annotated variables into the VC.

```
echo ":-startup_sequence." >> startcode.pl

#Load sicstus, and execute startcode.pl
sicstus -l startcode.pl
```

4.2 Stand-alone

The stand-alone system operates using a small SICSTUS runtime library, with all development operations unavailable. The system is started by running SICSTUS, compiling all the files of SPADE-PC$_{\text{SICSTUS}}$ and saving this state. The SICSTUS utility program SPLD (application builder) is then supplied this saved state, along with requests for key resources to add to the runtime environment, and generates an executable. This can be automated in a small shell script:

```
#!/bin/bash

#Load sicstus and consult checker.
#Then save state as: sspade_pc.sav.
sicstus <<-END_OF_FILE
compile(checker).
save_program('sspade_pc.sav').
halt.
END_OF_FILE

#Create executable.
spld sspade_pc.sav --main=restore --resources=clpfd,system --output=sspade_pc.exe
```

5 Annotated Code

All of the changes made to SPADE-PC$_{\text{POPLOG}}$ in creating SPADE-PC$_{\text{SICSTUS}}$ have been recorded using a number of annotation blocks.

5.1 Styles Of Block Annotations

Each block starts on a new line with leading characters: `%PORT:BEGIN` and ends on a following line with leading characters: `%PORT:END`. A few differt block forms have emerged:

5.1.1 New Code

Brand new code which does not have an obvious connection to SPADE-PC$_{\text{POPLOG}}$.

```
%PORT:BEGIN:NEWCODE#######################################\/#
%Comments.
%COMMENT=====================================================
The new code.
%PORT:END:NEWCODE#########################################/\#
```

5.1.2 Remove Code

Code removed from SPADE-PC$_{\text{POPLOG}}$.

```
%PORT:BEGIN:REMOVECODE####################################\/#
%Comments.
%COMMENT=====================================================
%The code removed.
%PORT:END:REMOVECODE######################################/\#
```

5.1.3 Replace Code

SPADE-PC$_{\text{POPLOG}}$ code replaced by corresponding SPADE-PC$_{\text{SICSTUS}}$ code.

```
%PORT:BEGIN:REPLACECODE#####################################\/#
%Comments.
%COMMENT====================================================
%The code removed.
%REMOVETONEW================================================
The new code.
%PORT:END:REPLACECODE#######################################/\#
```

5.2 Automatically Extracting Changes

The block structures can be automatically extracted and used to generate LaTeX code. The following scripts were used. (**Note:** The capitalised 'Verbatim' is a result of using package: `fancyvrb`.)

5.2.1 gettex.awk

```
BEGIN{
    inBlock="outside"
    line=1
}

/%PORT:BEGIN/{
    if(inBlock=="outside")
    {
        inBlock="inside";

        printf "\n"
        printf "\\subsubsection{Starting Line: %i}\n",
               line
        printf "\n"
        printf "\\begin{Verbatim}\n"
    }
}

/%PORT:END/{
    if(inBlock=="inside")
    {
        inBlock="leaving";
    }
}

{
    line=line+1;
    if(inBlock=="inside")
    {
        print $0;
    }
    else
    {
        if(inBlock=="leaving")
        {
            print $0;
            printf "\\end{Verbatim}\n"
            inBlock="outside";
        }
    }
}

END{
}
```

5.2.2 generatechanges

```
#!/bin/bash

#Reset port info file.
:> portinfo.tex

#Start.

echo "%BEGIN auto inserted latex." >> portinfo.tex
echo "" >> portinfo.tex

#Get all prolog files as $1 .. ${$#}.
set -a `ls *.pl`

#For each prolog file add appropriate latex.
while [ $# -gt 0 ]
do

    fileNameForLatex=`echo "$1" | sed -e 's/_/\\\\_/g'`
    echo "$fileNameForLatex"
    echo "" >> portinfo.tex
    echo "\subsection{File: $fileNameForLatex}" >> \
        portinfo.tex
    echo "" >> portinfo.tex

    #Go through awk script, and strip any
    #dos newlines and CTRL-G (beeps) using sed.
    cat $1 | awk -f gettex.awk \
           | sed -e 's/^M//g' -e 's/^G//g' \
           >> portinfo.tex

    shift
done

echo "" >> portinfo.tex
echo "%END auto inserted latex." >> portinfo.tex
```

6 All Block Changes

6.1 File: aritheval.pl

6.2 File: cases2.pl

6.3 File: checker.pl

6.3.1 Starting Line: 35

```
%PORT:BEGIN:NEWCODE#######################################\/#
%Set SICSTUS to behave more like POPLOG-PROLOG.
%In particular, treat strings in "" as lists of
%character codes.
%COMMENT=====================================================
:-set_prolog_flag(language,iso).
:-set_prolog_flag(double_quotes,codes).
%PORT:END:NEWCODE#########################################/\#
```

6.3.2 Starting Line: 44

```
%PORT:BEGIN:NEWCODE#######################################\/#
%Introduce SICSTUS library(system).
%This makes available operating system access predicates.
%The library makes available the following predicates
%used by SPADE-PC:
%datime(...)
%environ(..., ...)
%COMMENT=====================================================
:-use_module(library(system)).
%PORT:END:NEWCODE#########################################/\#
```

6.3.3 Starting Line: 55

```
%PORT:BEGIN:NEWCODE#######################################\/#
%Introduce SICSTUS library(lists)
%Make available list utility predicates.
%The library makes available the following predicates
%used by SPADE-PC:
%suffix(..., ...)
%append(..., ..., ...)
%sublist(..., ...)
%reverse(..., ...)
%last(..., ...)
%COMMENT=====================================================
:-use_module(library(lists)).
%PORT:END:NEWCODE#########################################/\#
```

6.3.4 Starting Line: 69

```
%PORT:BEGIN:NEWCODE#######################################\/#
%In SICSTUS there is a distinction between a dynamic
%predicate that is currently not available and a
%static predicate that does not exist.
%To aid this distinction it in necessary to declare all
%of the dynamic predicates of SPADE-PC.
%Note that these occur in no particular order.
%COMMENT=====================================================
:- dynamic (cmd_line_filename/1).
:- dynamic (command_log_filename/1).
:- dynamic (in_declare_command/0).
:- dynamic (is_true_vc/2).
:- dynamic (could_infer/1).
:- dynamic (could_not_infer/1).
:- dynamic (cmd_line_proof_log/1).
:- dynamic (cmd_line_command_log/1).
:- dynamic (cmd_line_load/1).
:- dynamic (logfact/2).
:- dynamic (perform_script_file/1).
:- dynamic (command_arg/2).
:- dynamic (input_from_terminal/0).
:- dynamic (vc_standardisation/1).
:- dynamic (stan_in_deduce/1).
:- dynamic (standardise_in_infer/1).
:- dynamic (record_consults/1).
:- dynamic (command_logging/1).
:- dynamic (show_vc_changes/1).
:- dynamic (auto_newvc/1).
:- dynamic (recent_save_command_issued/0).
:- dynamic (totally_specified_replace/0).
:- dynamic (vcs_proved_this_session/1).
:- dynamic (auto_done/1).
:- dynamic (banned_rule/2).
:- dynamic (case/3).
:- dynamic (case_pointer/1).
:- dynamic (conc/2).
:- dynamic (csvfile_name/1).
:- dynamic (current_root/2).
:- dynamic (current_vc/2).
:- dynamic (current_vc_no/1).
:- dynamic (deleted/1).
:- dynamic (deleted_hyp/2).
:- dynamic (display_subgoals_max/1).
:- dynamic (display_var_free_only/1).
:- dynamic (echo/1).
:- dynamic (enumeration/2).
:- dynamic (fdl_file_title/1).
:- dynamic (fdlfile_name/1).
:- dynamic (forgotten/1).
:- dynamic (function/3).
:- dynamic (function_template/3).
:- dynamic (hyp/2).
:- dynamic (indentation/1).
:- dynamic (indentation_increment/1).
:- dynamic (inverse_video/1).
:- dynamic (is_vc/1).
:- dynamic (library_already_loaded/1).
:- dynamic (logfile_name/1).
:- dynamic (mk__function_name/3).
:- dynamic (normal_video/1).
:- dynamic (on_case/3).
:- dynamic (on_filename/1).
:- dynamic (print_command/1).
:- dynamic (prooflog_width/1).
:- dynamic (proved_for_case/2).
:- dynamic (qvar/1).
:- dynamic (record_function/6).
:- dynamic (ruleused/1).
:- dynamic (ruleused_this_session/1).
:- dynamic (saved_vc/2).
:- dynamic (simplify_in_infer/1).
:- dynamic (simplify_during_load/1).
:- dynamic (spark_enabled).
:- dynamic (step_number/1).
:- dynamic (subgoal_formula/4).
:- dynamic (type/2).
:- dynamic (type_alias/2).
:- dynamic (typechecking/1).
:- dynamic (typechecking_during_load/1).
:- dynamic (use_subst_rules_for_equality/1).
:- dynamic (used_ident/2).
:- dynamic (user_rulefile/2).
:- dynamic (user_classification/4).
:- dynamic (var_const/3).
:- dynamic (vc/2).
:- dynamic (vcgfile_name/1).
:- dynamic (vcs_to_prove/1).
:- dynamic (vc_name/1).
%PORT:END:NEWCODE#########################################/\#
```

6.3.5 Starting Line: 159

```
%PORT:BEGIN:NEWCODE#######################################\/#
%The unfortunate situation of the EXAMINER outputing
%for_all and for_some predicates with a following
%space requires some additional processing in SICSTUS.
%Temporrory operators for for_all and for_some are part
%of the solution.
%COMMENT=====================================================
:- op(800        ,fy   ,for_all                ).
:- op(800        ,fy   ,for_some               ).
%PORT:END:NEWCODE#########################################/\#
```

6.3.6 Starting Line: 175

```
%PORT:BEGIN:REMOVECODE####################################\/#
%Remove some POPLOG-PROLOG specific initialisations, as
%follows. Note that many of these are concerned with
%efficiency. The corresponding SICSTUS predicates might be
%employed eventually.
%
%prolog_memlim(...)
%This predicate is used to specify the amount
%of memory to make available during the execution
%of a POPLOG-PROLOG program.
%
%library(simplepop)
%This library allows small pieces of POPLOG code to
%be executed from within POPLOG-PROLOG. It adds
%the following predicates used by POPLOG-SPADE-PC:
%dopop(...)
%
%library(are)
%This library introduces the infix operator 'are'. This is
%used similar to the PROLOG is operator. However, are has
%access to arbitrary POPLOG procedures.
%
%library(languages)
%This library allows for extensive access to POPLOG
%from within POPLOG-PROLOG.
%
%library(current_disk)
%This library provides access to information about the
%external file structure. The library makes available
%the following predicates used by POPLOG-SPADE-PC:
%current_disk(...)
%current_disk(..., ...)
%
%prolog_unlock_code
%This predicate is used to free any previously locked
%code using prolog_lock_code.
%
%prolog_no_clauses
%This predicate instructs the compiler to stop recording
%the text associated with each read in predicate. This
%reduces the amount of information stored in the
%system and therefore should improve performance.
%
%prolog_syspredicates(.A.)
%This predicate instructs the compiler to treat the
%following predicates as either static or dynamic.
%.A. as on is for static.
%.A. as off is for dynamic.
%Static predicates can be held more efficiently in
%memory, as they do not need to support dynamic
%assertion and retraction. Informing the compiler
%of static predicates is therefore a means to
%improve performance.
%COMMENT=====================================================
%:- prolog_memlim(500000).
%:- library(simplepop).
%:- library(are).
%:- library(languages). /* C1.41 added */
%:- library(current_disk).
%:- prolog_unlock_code.
%:- prolog_no_clauses.
%:- prolog_syspredicates(on).
%PORT:END:REMOVECODE######################################/\#
```

6.3.7 Starting Line: 257

```
%PORT:BEGIN:NEWCODE######################################\/#
%SICSTUS does not support not as a synonym for \+.
%This is used throughout SPADE-PC, and thus is introduced
%as a new predicate, rather than changing all occurrences
%to \+.
%COMMENT====================================================
(not (X)):- \+ X.
%PORT:END:NEWCODE########################################/\#
```

6.3.8 Starting Line: 281

```
%PORT:BEGIN:REMOVECODE###################################\/#
%Remove more calls to Poplog-Prolog specific predicates.
%COMMENT====================================================
%:- prolog_clauses.
%:- prolog_lock_code.
%:- prolog_unlock_code.
%:- prolog_no_clauses.
%:- prolog_syspredicates(on).
%PORT:END:REMOVECODE#####################################/\#
```

6.3.9 Starting Line: 322

```
%PORT:BEGIN:REMOVECODE###################################\/#
%Remove more Poplog-Prolog specific predicates.
%
% prolog_clauses
% This predicates performs the same operation as
% as prolog_syspredicates(off). It is implemented
% in Poplog-Prolog thus:
% prolog_clauses :-
%         prolog_no_clauses(off).
%COMMENT====================================================
%:- prolog_syspredicates(off).
%:- prolog_clauses.
%:- prolog_lock_code.
%PORT:END:REMOVECODE#####################################/\#
```

6.3.10 Starting Line: 342

```
%PORT:BEGIN:REMOVECODE###################################\/#
%This dynamic modification of the code gives a tiny
%efficiency gain. It is not repeated in sicstus.
%COMMENT====================================================
%:- retractall(bonsult(_)).
%PORT:END:REMOVECODE#####################################/\#
```

6.3.11 Starting Line: 354

```
%PORT:BEGIN:REMOVECODE###################################\/#
%Remove macro actions. There is no corresponding
%macro facilitys in sicstus.
%COMMENT====================================================
%/*** WE DON'T WANT NO MACRO'S ***/
%:- library(macro).
%:- killmac("help").
%:- killmac("teach").
%:- killmac("ref").
%:- killmac("doc").
%:- killmac("showlib").
%:- killmac("src").
%:- killmac("ved").
%:- killmac("im").
%:- killmac("pwd").
%:- killmac("cd").
%:- retractall(killmac(_)), retractall(macro(_,_)), retractall(macval(_,_)).
%PORT:END:REMOVECODE#####################################/\#
```

6.3.12 Starting Line: 390

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Remove start up sequence. This contains a number of
%Poplog-Prolog specific predicates that need to be
%entirely reworked for sicstus.
%
%Note that currently all security predicates have been
%removed in SICSTUS-SPADE-PC. Only the operational aspects
%of SPADE-PC are required for NuSPADE.
%COMMENT====================================================
%startup_sequence :-
%    dopop(' procedure(); chain(prolog_invoke(% "bombout" %)) endprocedure -> interrupt; '),
%    version,
%    /* C1.41: newlines moved to after getlicence */
%    machine_startup,                                          /* CFR048 */
%    g1, /* C1.41: getlicence */
%    write_banner, /* C1.41 */
%    nl,
%    nl,
%    date_ok, /* C1.41: added */
%    process_command_line_data,
%    read_initialisations,
%    (load_vc),
%    nl,
%    write('Welcome to the SPADE Proof Checker -- for assistance type "help"'),
%    nl,
%    nl,
%    fail.
%startup_sequence :-
%    load_buffered_libs,
%    write_log,
%    fail.
%startup_sequence :-
%    do_do_newvc,
%    see_correct_input_stream,
%    execute_command(newvc),
%    write_log,
%    fail.
%startup_sequence :-
%    dopop(' procedure(); chain(prolog_invoke(% "start" %)) endprocedure -> interrupt; '),
%    !,
%    start.
%REMOVETONEW================================================
startup_sequence :-
    machine_startup,                                           /* CFR048 */
    nl,
    nl,
    process_command_line_data,
    read_initialisations,
    (load_vc),
    nl,
    write('Welcome to the SPADE Proof Checker -- for assistance type "help"'),
    nl,
    nl,
    fail.

startup_sequence :-
    load_buffered_libs,
    write_log,
    fail.

startup_sequence :-
    do_do_newvc,
    see_correct_input_stream,
    execute_command(newvc),
    write_log,
    fail.

startup_sequence :-
    !,
    %Call start.
    %If there is an exception within start (the main part
    %of the program) then call restartShowingError(...).
    catch(start,
    ISO_Error,
    restartShowingError(ISO_Error)).

%Display the exception and restart the main program.
%This is similar to the behaviour of SPADE-PC.
%Different exceptions in SICSTUS make it difficult to
%mirror exactly the same behaviour.
restartShowingError(ISO_Error):-
    nl,
    write('Syntax_error exception raised.'), nl,
    write('ISO description of exception:'), nl,
    write(ISO_Error), nl,
    write('Internal restart being performed.'), nl,!,
    catch(start,
    ISO_Error,
    restartShowingError(ISO_Error)).
%PORT:END:REPLACECODE####################################/\#
```

6.3.13 Starting Line: 481

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace Poplog-Prolog specific predicates for each
%machine. The main change here is for accessing
%environment variables.
%Note that SICSTUS-SPADE-PC does not operate under VAX.
%Thus if the VAX machine is detected it aborts.
%COMMENT====================================================
%machine_startup :-                                             /* CFR048 */
%        machine(vax),                                             /* CFR048 */
%        assert(print_command("PRINT/DELETE")),                     /* CFR048 */
%        assert(spade_checker_prefix("SPADE$CHECKER:")),             /* CFR048 */
%        assert(spade_chkhelp_prefix("SPADE$CHKHELP:")),             /* CFR048 */
%        assert(popsys_prefix("popsys:")),                       /* CFR048 */
%        assert(qualifier_prefix(47)),                             /* CFR048 */
%        !.                                                      /* CFR048 */
%machine_startup :-                                             /* CFR048 */
%        machine(sun),                                             /* CFR048 */
%        assert(print_command("lpr")),                             /* CFR048 */
%        assert(spade_checker_prefix("$SPADE_CHECKER/")),       /* CFR048 */
%        assert(spade_chkhelp_prefix("$SPADE_CHKHELP/")),       /* CFR048 */
%        assert(popsys_prefix("$popsys/")),                      /* CFR048 */
%        assert(qualifier_prefix(45)),                             /* CFR048 */
%        !.                                                      /* CFR048 */
%
%machine_startup :-                                             /* CFR048 */
%        machine(nt),                                             /* CFR048 */
%        assert(print_command("print")),                             /* CFR048 */
%        assert(spade_checker_prefix("%SPADE_CHECKER%\\")),      /* CFR048 */
%        assert(spade_chkhelp_prefix("%SPADE_CHKHELP%\\")),      /* CFR048 */
%        assert(popsys_prefix("%POPSYS%\\")),                     /* CFR048 */
%        assert(qualifier_prefix(47)),                             /* CFR048 */
%        !.                                                      /* CFR048 */
%REMOVETONEW================================================
%Make available in the PROLOG environment key environment
%variables.
introduceEnvironmentVariables:-
    environ('SPADE_CHECKER',RuleLocation1),
    environ('SPADE_CHKHELP',HelpLocation1),
    %Add a terminating '/' to copy original implementation.
    addToEndOfAtom(RuleLocation1,47,RuleLocation2),
    addToEndOfAtom(HelpLocation1,47,HelpLocation2),
    name(RuleLocation2,RuleLocation3),
    name(HelpLocation2,HelpLocation3),
    assert(spade_checker_prefix(RuleLocation3)),
    assert(spade_chkhelp_prefix(HelpLocation3)),
    assert(qualifier_prefix(47)),
    !.

addToEndOfAtom(InAtom,AtEnd,NewAtom):-
    name(InAtom,InAtomCharList),
    append(InAtomCharList,[AtEnd],NewAtomCharList),
    name(NewAtom,NewAtomCharList).

machine_startup :-
    machine(vax),
    nl, nl, put(7),
    write('!!! SICSTUS-SPADE IS NOT AVAILABLE FOR VAX !!!'),
    nl,
    !,
    halt.

machine_startup :-
    machine(sun),
    assert(print_command("lpr")),
```

```
    introduceEnvironmentVariables,
    !.

machine_startup :-
    machine(nt),
    assert(print_command("print")),
    introduceEnvironmentVariables,
    !.
%PORT:END:REPLACECODE###################################/\#
```

6.3.14 Starting Line: 572

```
%PORT:BEGIN:NEWCODE######################################\/#
%A hook is made from the key SICSTUS predicate
%runtime_entry(start) executed at startup to the main
%predicate of SPADE-PC, startup_sequence.
%COMMENT=================================================
runtime_entry(start):-
    startup_sequence.
%PORT:END:NEWCODE########################################/\#
```

6.4 File: cipher.pl

6.5 File: contra.pl

6.6 File: datecheck.pl

6.7 File: declar.pl

6.7.1 Starting Line: 26

```
%PORT:BEGIN:REMOVECODE###################################\/#
%The entire contents of the declare file is replaced to
%cope with the different SICSTUS operators.
%COMMENT=================================================
%:- op(20,xfx,**).   /* 'Raise to the power' operator */               /* CFR038 */
%:- op(20,xf,~).                          /* INITIAL VALUE POSTFIX */
%:- op(20,yfx,#).                /* New HYP/CONC syntax */
%
%:- op(21,fx,+).                 /* Unary + */
%:- op(21,fx,-).                 /* Unary - */
%
%:- op(25,yfx,*).
%:- op(25,yfx,/).
%:- op(25,yfx,div).
%:- op(25,yfx,mod).
%:- op(25,yfx,@).                 /* Sequences */
%:- op(25,yfx,/\).                 /* Sets */
%
%:- op(31,yfx,+).
%:- op(31,yfx,-).
%:- op(31,yfx,\/).                /* Sets */
%:- op(31,yfx,\).                 /* Sets */
%
%:- op(40,fy,set).                /* Sets */
%:- op(40,yfx,<>).                    /* Used in rules */
%:- op(40,xfx,subset_of).             /* Set types */
%:- op(40,yfx,strict_subset_of).      /* Set types */
%:- op(40,xfx,iss).              /* Signed 'is' */
%:- op(40,yfx,less_than).        /* Arithmetic */
%:- op(40,yfx,<=).                /* from FDL */
%
%:- op(100,yfx,in).                 /* Sets */
%:- op(100,yfx,not_in).           /* Sets */
%:- op(100,yfx,'..').                /* Used in aggregates */        /* CFR034 */
%
%:- op(200,fy,not).      /* Required by POPLOG */
%
%:- op(210,yfx,and).
%
%:- op(220,yfx,or).
%:- op(220,yfx,xor).                        /* Added for Z8002 */
%
%/* Note that in the _checker_ (but not the Simplifier) "->" binds more */
%/* tightly than "<->".                                                */
%
%:- op(230,xfy,->).
%:- op(230,yfx,requires).             /* Used in rules */
%
%:- op(235,yfx,<->).
%
%:- op(238,yfx,':=').                 /* Used in aggregates */        /* CFR034 */
%
%:- op(240,fx,consult).                /* For initialisation file */
%:- op(240,fx,load).                /* For initialisation file */
%:- op(240,yfx,&).                     /* Used in rules */
%:- op(240,fx,rule_family).            /* Used in rules */
%
%:- op(243,yfx,where).                 /* New command-line syntax */
%
%:- op(245,yfx,by).                /* New command-line syntax */
%:- op(245,yfx,with).                 /* New command-line syntax */
%:- op(245,yfx,to).                 /* New command-line syntax */
%:- op(245,yfx,may_be_deduced_from).  /* Used in rules */
%:- op(245,yfx,may_be_replaced_by).   /* Used in rules */
%:- op(245,xf,are_interchangeable).   /* Used in rules */
%:- op(245,yfx,if).                   /* Used in rules */
%:- op(245,xf,may_be_deduced).              /* Used in rules */
%
%:- op(247,yfx,=>).           /* Substitution  */
%:- op(247,yfx,using).                 /* New command-line syntax */
%:- op(247,yfx,for).                /* New command-line syntax */
%:- op(247,yfx,on).                /* New command-line syntax */
%
%:- op(250,xfy,:).                     /* Used in rules */
%:- op(250,yfx,from).                 /* New command-line syntax */
%PORT:END:REMOVECODE#####################################/\#
```

6.7.2 Starting Line: 103

```
%PORT:BEGIN:NEWCODE######################################\/#
%Previously this file introduced new operators that do not
%already exist in POPLOG-PROLOG. It also redeclared
%some of the operators that already exist in
%POPLOG-PROLOG. Where no changes needed to be made
%to POPLOG-PROLOG, SPADE-PC relies on the default
%precedence of POPLOG-PROLOG operators.
%
%SICSTUS employs a completely different operator
%precedence than that of POPLOG-PROLOG. Some SICSTUS
%operators can not (and many should not) be redeclared.
%
%It is therefore necessary to determine all of the
%operators of SPADE-PC. Then port the ordering of
%the resulting precedence to SICSTUS. It is also
%considered important to explicitly record the
%operators used even though the SICSTUS defaults
%might be relied upon.
%
%Note that NuSPADE will share these operators.
%NuSPADE operators not used in SPADE-PC will be
%declared as part of NuSPADE.
%
%Notes on operators
%=================
%Declare with: :- op(precedence, type, name).
%
%The possible types for an infix operator (X operator Y) are:
%xfx     xfy     yfx
%
%The possible types for a prefix operator (operator X) are:
%fx      fy
%
%The possible types for a postfix operator (X operator) are:
%xf      yf
%
%It is possible to have more than one operator of the same name,
%so long as they are of different kinds, i.e. infix, prefix or
%postfix. Note that the ISO PROLOG standard contains a
%restriction that there should be no infix and postfix operators
%with the same name, however, SICSTUS PROLOG lifts this
%restriction.
%
%About the table
%==============
%This following table only lists the operators required by
%SPADE-PC. There are many POPLOG-PROLOG and SICSTUS operators
%not listed here.
%
%1-
%In SICSTUS X#Y is the bitwise exclusive or of the
%integers X and Y. In SPADE-PC this is used for selecting
%hypothesis and conclusions (h#3, c#1). The SICSTUS
%operator is considered not very valuable, and is
%not even ISO standard. Thus the SPADE-PC
%definition for X#Y is taken instead, changing
%the SICSTUS precedence accordingly.
%
%2-
%In normal SICSTUS mode, the following extra operators
%are added:
%:- op(  500,  fx, [ +, - ]).
%:- op(  300, xfx, [ mod ]).
%Although SICSTUS allows fx and fy operators to coexist,
%it is worth noting this unusual behaviour here.
%
%4-
%SPADE-PC uses /\ with precedence 25 and \/ with 31.
%The Poplog /\ operator, has precedence 31. The SPADE-PC
%precedence is used.
%
%5-
%In Sicstus X:Y is used to work with modules.
%In SPADE-PC this is used in rule files, with a different
%precedence ordering. Both of these are key operations.
%For the moment the SPADE-PC precedence is used.
%It may be necessary to dynamically change the precedence
%of this operator.
%
%6-
%The operator -> is base in POPLOG and SICSTUS as the
%conditional operator.
%(Where: (P -> Q) ; R is analogous to (if P then Q else R))
%However, SPADE-PC uses -> for implication, with an incompatible
%precedence.
%As changing the SPADE-PC precedence would mean substantial
%rule changes, its precedence must remain. Thus where used
%as a conditional operator brackets may be required.
%In many cases the SPADE-PC precedence is unlikely to require
%any changes. However, if warranted a special -> operator might
%be declared in SICSTUS.
%
%7-
%The operator ; is key in PROLOG. The SICSTUS precedence
%definition is taken.
%
%%-
%This flag indicates that the SICSTUS defaults will be used.
%This means that the accuracy of SPADE-PC is relying on the
%SICSTUS operator defaults (As was previously done in POPLOG-PROLOG).
%For this reason this is only done where necessary.
%As it is necessary, the SICSTUS defaults are expected to be
%well established. Thus this is not an issue, so long as
%monitored across Sicstus versions.
%
/*
|-------------------------|-----------------------|-----------------------|
|Operator                 |POPLOG-PROLOG          |SICSTUS                |
|-------------------------|-----------------------|-----------------------|
|Name               |Type |Precedence |Base |Extra |Precedence |Base |Extra |
|-------------------|-----|-----------|-----|------|-----------|-----|------|
|^                  |xfy  |10         |XXXXX|      |200        |XXXXX|      |
|~                  |xf   |20         |     |XXXXXX|200        |     |XXXXXX|
|**                 |xfx  |20         |     |XXXXXX|200        |XXXXX|      |
|#                  |yfx  |20         |     |XXXXXX|500 => 200 |XXXXX|      |1
|/\                 |yfx  |21         |     |XXXXXX|500        |     |XXXXXX|
|-                  |fy   |21         |     |XXXXXX|200        |XXXXX|      |2
|+                  |fy   |21         |     |XXXXXX|200        |XXXXX|      |2
|*                  |yfx  |25         |XXXXX|      |400        |XXXXX|      |
```

```
|/                  |yfx  |25         |XXXXX|      |400        |XXXXX|      |
|div                |yfx  |25         |XXXXX|      |400        |     |XXXXXX|
|mod                |yfx  |25         |XXXXX|      |400        |XXXXX|      |2
|@                  |yfx  |25         |     |XXXXXX|400        |     |XXXXXX|
|/\                 |yfx  |31 => 25   |XXXXX|      |400        |XXXXX|      |4
|+                  |yfx  |31         |XXXXX|      |500        |XXXXX|      |
|-                  |yfx  |31         |XXXXX|      |500        |XXXXX|      |
|\/                 |yfx  |31         |XXXXX|      |500        |XXXXX|      |
|\                  |yfx  |31         |     |XXXXXX|500        |     |XXXXXX|
|set                |fy   |40         |     |XXXXXX|700        |     |XXXXXX|
|<>                 |yfx  |40         |     |XXXXXX|700        |     |XXXXXX|
|subset_of          |xfx  |40         |     |XXXXXX|700        |     |XXXXXX|
|strict_subset_of   |yfx  |40         |     |XXXXXX|700        |     |XXXXXX|
|iss                |xfx  |40         |     |XXXXXX|700        |     |XXXXXX|
|less_than          |yfx  |40         |     |XXXXXX|700        |     |XXXXXX|
|<=                 |yfx  |40         |     |XXXXXX|700        |     |XXXXXX|
|is                 |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|<                  |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|=<                 |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|==                 |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|=                  |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|=:=                |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|=..                |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|=\=                |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|>=                 |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|>                  |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|@<                 |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|@=<                |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|@>=                |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|@>                 |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|\==                |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|\=                 |xfx  |40         |XXXXX|      |700        |XXXXX|      |
|in                 |yfx  |100        |     |XXXXXX|750        |     |XXXXXX|
|not_in             |yfx  |100        |     |XXXXXX|750        |     |XXXXXX|
|..                 |yfx  |100        |     |XXXXXX|750        |     |XXXXXX|
|not                |fy   |200        |     |XXXXXX|800        |     |XXXXXX|
|and                |yfx  |210        |     |XXXXXX|810        |     |XXXXXX|
|or                 |yfx  |220        |     |XXXXXX|820        |     |XXXXXX|
|xor                |yfx  |220        |     |XXXXXX|820        |     |XXXXXX|
|->                 |xfy  |253 => 230 |XXXXX|      |1100 => 830|XXXXX|      |6
|requires           |yfx  |230        |     |XXXXXX|830        |     |XXXXXX|
|<->                |yfx  |235        |     |XXXXXX|835        |     |XXXXXX|
|:=                 |yfx  |238        |     |XXXXXX|838        |     |XXXXXX|
|consult            |fx   |240        |     |XXXXXX|840        |     |XXXXXX|
|load               |fx   |240        |     |XXXXXX|840        |     |XXXXXX|
|&                  |yfx  |240        |     |XXXXXX|840        |     |XXXXXX|
|rule_family        |fx   |240        |     |XXXXXX|840        |     |XXXXXX|
|where              |yfx  |243        |     |XXXXXX|843        |     |XXXXXX|
|by                 |yfx  |245        |     |XXXXXX|845        |     |XXXXXX|
|with               |yfx  |245        |     |XXXXXX|845        |     |XXXXXX|
|to                 |yfx  |245        |     |XXXXXX|845        |     |XXXXXX|
|may_be_deduced_from |yfx  |245        |     |XXXXXX|845        |     |XXXXXX|
|may_be_replaced_by |yfx  |245        |     |XXXXXX|845        |     |XXXXXX|
|are_interchangeable |xf   |245        |     |XXXXXX|845        |     |XXXXXX|
|if                 |yfx  |245        |     |XXXXXX|845        |     |XXXXXX|
|may_be_deduced     |xf   |245        |     |XXXXXX|845        |     |XXXXXX|
|=>                 |yfx  |247        |     |XXXXXX|847        |     |XXXXXX|
|using              |yfx  |247        |     |XXXXXX|847        |     |XXXXXX|
|for                |yfx  |247        |     |XXXXXX|847        |     |XXXXXX|
|on                 |yfx  |247        |     |XXXXXX|847        |     |XXXXXX|
|:                  |xfy  |250        |     |XXXXXX|550 => 950 |XXXXX|      |5
|from               |yfx  |250        |     |XXXXXX|950        |     |XXXXXX|
|\+                 |fy   |249        |XXXXX|      |900        |XXXXX|      |
|','                |xfy  |252        |XXXXX|      |1000       |XXXXX|      |%
|;                  |xfy  |254        |XXXXX|      |1100       |XXXXX|      |7
|:-                 |fx   |255        |XXXXX|      |1200       |XXXXX|      |
|?-                 |fx   |255        |XXXXX|      |1200       |XXXXX|      |
|-->                |xfx  |255        |XXXXX|      |1200       |XXXXX|      |
|:-                 |xfx  |255        |XXXXX|      |1200       |XXXXX|      |
|-------------------|-----|-----------|-----|------|-----------|-----|------|*/
%Note:
%Select the table content rows in visual mode in vim and
%use the following substitution to auto generate the ops.
%s/^|\([^|]*\)|\([^|]*\)|[^|]*|[^|]*|[^|]*\(|\||[0-9]* => \)\([0-9]*\)\( *\)|[^%]*\(%\=\)$/\6:-op(\4\5,\2,\1)./g
%COMMENT======================================================
:-op(200        ,xfy  ,^                  ).
:-op(200        ,xf   ,~                  ).
:-op(200        ,xfx  ,**                 ).
:-op(200        ,yfx  ,#                  ).
:-op(500        ,yfx  ,/\                 ).
:-op(200        ,fy   ,-                  ).
:-op(200        ,fy   ,+                  ).
:-op(400        ,yfx  ,*                  ).
:-op(400        ,yfx  ,/                  ).
:-op(400        ,yfx  ,div                ).
:-op(400        ,yfx  ,mod                ).
:-op(400        ,yfx  ,@                  ).
:-op(400        ,yfx  ,/\                 ).
:-op(500        ,yfx  ,+                  ).
:-op(500        ,yfx  ,-                  ).
:-op(500        ,yfx  ,\/                 ).
:-op(500        ,yfx  ,\                  ).
:-op(700        ,fy   ,set                ).
:-op(700        ,yfx  ,<>                 ).
:-op(700        ,xfx  ,subset_of          ).
:-op(700        ,yfx  ,strict_subset_of   ).
:-op(700        ,xfx  ,iss                ).
:-op(700        ,yfx  ,less_than          ).
:-op(700        ,yfx  ,<=                 ).
:-op(700        ,xfx  ,is                 ).
:-op(700        ,xfx  ,<                  ).
:-op(700        ,xfx  ,=<                 ).
:-op(700        ,xfx  ,==                 ).
:-op(700        ,xfx  ,=                  ).
:-op(700        ,xfx  ,=:=                ).
:-op(700        ,xfx  ,=..                ).
:-op(700        ,xfx  ,=\=                ).
:-op(700        ,xfx  ,>=                 ).
:-op(700        ,xfx  ,>                  ).
:-op(700        ,xfx  ,@<                 ).
:-op(700        ,xfx  ,@=<                ).
:-op(700        ,xfx  ,@>=                ).
:-op(700        ,xfx  ,@>                 ).
:-op(700        ,xfx  ,\==                ).
:-op(700        ,xfx  ,\=                 ).
:-op(750        ,yfx  ,in                 ).
:-op(750        ,yfx  ,not_in             ).
:-op(750        ,yfx  ,..                 ).
:-op(800        ,fy   ,not                ).
:-op(810        ,yfx  ,and                ).
:-op(820        ,yfx  ,or                 ).
:-op(820        ,yfx  ,xor                ).
:-op(830        ,xfy  ,->                 ).
:-op(830        ,yfx  ,requires           ).
:-op(835        ,yfx  ,<->                ).
:-op(838        ,yfx  ,:=                 ).
:-op(840        ,fx   ,consult            ).
:-op(840        ,fx   ,load               ).
:-op(840        ,yfx  ,&                  ).
:-op(840        ,fx   ,rule_family        ).
:-op(843        ,yfx  ,where              ).
:-op(845        ,yfx  ,by                 ).
:-op(845        ,yfx  ,with               ).
:-op(845        ,yfx  ,to                 ).
:-op(845        ,yfx  ,may_be_deduced_from ).
:-op(845        ,yfx  ,may_be_replaced_by ).
:-op(845        ,xf   ,are_interchangeable ).
:-op(845        ,yfx  ,if                 ).
:-op(845        ,xf   ,may_be_deduced     ).
:-op(847        ,yfx  ,=>                 ).
:-op(847        ,yfx  ,using              ).
:-op(847        ,yfx  ,for                ).
:-op(847        ,yfx  ,on                 ).
:-op(950        ,xfy  ,:                  ).
:-op(950        ,yfx  ,from               ).
:-op(900        ,fy   ,\+                 ).
%:-op(1000        ,xfy  ,','                ).
:-op(1100       ,xfy  ,;                  ).
:-op(1200       ,fx   ,:-                 ).
:-op(1200       ,fx   ,?-                 ).
:-op(1200       ,xfx  ,-->                ).
:-op(1200       ,xfx  ,:-                 ).
%PORT:END:NEWCODE###########################################/\#
```

6.8 File: decrypt.pl

6.9 File: deduce.pl

6.10 File: deduction.pl

6.11 File: done3.pl

6.12 File: fwdch2.pl

6.13 File: getdcldat.pl

6.13.1 Starting Line: 74

```
%PORT:BEGIN:REPLACECODE####################################\/#
%Replace POPLOG-PROLOG: ... is valof(poparglist)
%with SICSTUS predicate to access the command arguments
%as a list.
%COMMENT======================================================
%        DATA is valof(poparglist),
%REMOVETONEW==================================================
        current_prolog_flag(argv,DATA),
%PORT:END:REPLACECODE######################################/\#
```

6.13.2 Starting Line: 118

```
%PORT:BEGIN:REPLACECODE####################################\/#
%Replace POPLOG-PROLOG: ... explode(...)
%with the apparently similar SICSTUS predicate for
%destructing atoms.
%COMMENT======================================================
%        ATOM_CHARS are explode(ATOM),
%REMOVETONEW==================================================
        name(ATOM, ATOM_CHARS),
%PORT:END:REPLACECODE######################################/\#
```

6.13.3 Starting Line: 229

```
%PORT:BEGIN:NEWCODE########################################\/#
%To perform a beep in SICSTUS, an escape code (\a) is
%written. Rather than have lots of (\a)s an explicit beep
%predicate is used.
%COMMENT======================================================
beep:-
    write('\a').
%PORT:END:NEWCODE##########################################/\#
```

6.13.4 Starting Line: 240

```
%PORT:BEGIN:REPLACECODE####################################\/#
%SICSTUS does not like the use of control characters,
%in this case CTRL-G (the beep code), inside quotes. They
%are removed and replaced with an explicit call to beep.
%COMMENT======================================================
%        write('FAILED: ERROR IN COMMAND LINE SYNTAX'),
%REMOVETONEW==================================================
        beep,
        write('FAILED: ERROR IN COMMAND LINE SYNTAX'),

%PORT:END:REPLACECODE######################################/\#
```

6.13.5 Starting Line: 322

```
%PORT:BEGIN:REPLACECODE####################################\/#
%SICSTUS does not like the use of control characters,
%in this case CTRL-G (the beep code), inside quotes. They
%are removed and replaced with an explicit call to beep.
%COMMENT======================================================
%           write('FAILED: SCRIPT FILE NOT FOUND'),                 /* CFR014 */
%REMOVETONEW==================================================
        beep,
        write('FAILED: SCRIPT FILE NOT FOUND'),
%PORT:END:REPLACECODE######################################/\#
```

6.13.6 Starting Line: 347

```
%PORT:BEGIN:REPLACECODE####################################\/#
%SICSTUS does not like the use of control characters,
%in this case CTRL-G (the beep code), inside quotes. They
%are removed and replaced with an explicit call to beep.
%COMMENT=====================================================
%           write('FAILED: SCRIPT FILE NOT FOUND'),              /* CFR014 */
%REMOVETONEW=================================================
        beep,
        write('FAILED: SCRIPT FILE NOT FOUND'),
%PORT:END:REPLACECODE######################################/\#
```

6.14 File: getlicence.pl

6.14.1 Starting Line: 63

```
%PORT:BEGIN:REPLACECODE####################################\/#
%The entire licence code is replaced by a series of stubs.
%It is not ported to SICSTUS, rather it is taken out off
%SPADE-PC.
%COMMENT=====================================================
%/* ---------------------------------------------------------------------------
% * callc()                                    Codename: g2
% *
% * Pop-11 procedure to call the C function which interrogates FLEXlm
% * Preceeded by the Pop-11 code to import the objects from the C object code
% * files.
% * ---------------------------------------------------------------------------
% */
%POP11
%/* define data type to communicate with C */
%/* the field names are deliberately obscure in case they appear in the
% * resultant image file
% */
%p_typespec flex_data_type {
%  t1: exptr.exacc_ntstring, /* version number */
%  t2: int,                  /* seed used in encryption of version number */
%  t3: exptr.exacc_ntstring, /* tool name */
%  t4: int,                  /* seed used in encryption of tool name */
%  t5: int,                  /* inputseed : used by C to generate encryption
%                             * seed for rest of data */
%
%  t6_1: exptr.exacc_ntstring, /* banner line 1 */
%  t6_2: exptr.exacc_ntstring, /* banner line 2 */
%  t6_3: exptr.exacc_ntstring, /* banner line 3 */
%  t7: exptr.exacc_ntstring, /* warning date */
%  t8: exptr.exacc_ntstring  /* expiry date */
%};
%/* load external objects */
%exload 'pxflexlib' [pxflexlib]
%  a7(1): exptr,      /* fn returning ptr to address of flex_data structure */
%  a3(1): int,        /* init_data() */
%  a1(1): int         /* get_spade_licence() */
%endexload;
%
%/* procedure to call C */
%define g2(version, versionseed, tool, toolseed, inputseed);
%  /* init data structure */
%  lvars x, y;
%  exacc a7(1) -> x;             /* get pointer to C structure */
%  exacc a3(versionseed) -> y;   /* note that the parameter and return of
%                                 * init_data are not used but are included
%                                 * to confuse a potential hacker
%                                 */
%  /* write information to it: the >< is the string concatenation operator:
%   * we are using a sideeffect which is that it converts word arguments
%   * into strings
%   */
%  version >< '' -> exacc: flex_data_type x.t1;
%  versionseed -> exacc: flex_data_type x.t2;
%  tool >< '' -> exacc: flex_data_type x.t3;
%  toolseed -> exacc: flex_data_type x.t4;
%  inputseed -> exacc: flex_data_type x.t5;
%
%  /* call C */
%  exacc a1(y);  /* the parameter y is not used, but is again included
%                 * to confuse a hacker
%                 * note that the return value of callc is left on the stack
%                 * and discarded by getlicence
%                 */
%  /* retrieve information (and leave on stack) */
%  exacc: flex_data_type x.t6_1;
%  exacc: flex_data_type x.t6_2;
%  exacc: flex_data_type x.t6_3;
%  exacc: flex_data_type x.t8;
%  exacc: flex_data_type x.t7;
%enddefine;
%PROLOG
%
%/* ---------------------------------------------------------------------------
% * getlicence                                    Codename: g1
% *
% * Generate the seed to use in the handshake with the C function, and the seed
% * which the C function will derive from it to encrypt the data which
% * it returns.
% * Format the information required for the Pop-11 procedure callc.
% * Call it, and output the results
% * ---------------------------------------------------------------------------
% */
%g1 :-
%  z1(Version, VersionSeed),      /* encrypted tool version */
%  z2(ToolName, ToolNameSeed),    /* encrypted tool name */
%  x2(TodaysSeed),                /* derive a seed from the current date
%                                  * and time
%                                  */
%  x3(TodaysSeed, OutputSeed),    /* derives seed to use when decrypting
%                                  * information returned by C
%                                  */
%  asserta(g6(OutputSeed)),
%  Out are g2(Version, VersionSeed, ToolName, ToolNameSeed, TodaysSeed),
%  Out is [Dummy, EncryptedCustomerBanner1, EncryptedCustomerBanner2,
%          EncryptedCustomerBanner3, EncryptedExpiryDate,
%          EncryptedWarningDate],
%  asserta(g3_1(EncryptedCustomerBanner1)),
%  asserta(g3_2(EncryptedCustomerBanner2)),
%  asserta(g3_3(EncryptedCustomerBanner3)),
%  asserta(g4(EncryptedWarningDate)),
%  asserta(g5(EncryptedExpiryDate)).
%
%
%/* ---------------------------------------------------------------------------
% * customer_banner(Line)            Codename: g7_?
% *
% * Decrypts a line of the banner. Actually three predicates g7_1 g7_2 g7_3.
% * May return an empty word.
% * ---------------------------------------------------------------------------
% */
%/* first line of banner */
%g7_1(Line) :-
%  g3_1(EncryptedLine),   /* read encrypted line */
%  g6(OutputSeed),                   /* read seed */
%  x1(EncryptedLine, Line, OutputSeed).  /* decrypt */
%
%/* second line of banner */
%g7_2(Line) :-
%  g3_2(EncryptedLine),   /* read encrypted line */
%  g6(OutputSeed),                   /* read seed */
%  x1(EncryptedLine, Line, OutputSeed). /* decrypt */
%
%/* third line of banner */
%g7_3(Line) :-
%  g3_3(EncryptedLine),   /* read encrypted line */
%  g6(OutputSeed),                   /* read seed */
%  x1(EncryptedLine, Line, OutputSeed). /* decrypt */
%
%/* ---------------------------------------------------------------------------
% * warning_date(Date)
% *
% * Decrypts the warning date returned by C.
% * ---------------------------------------------------------------------------
% */
%warning_date(Date) :-
%  g6(OutputSeed),           /* read seed */
%  g4(EncryptedDate),        /* read encrypted date */
%  x1(EncryptedDate, Date, OutputSeed). /* decrypt */
%
%/* ---------------------------------------------------------------------------
% * expiry_date(Date)
% *
% * Decrypts the expiry date returned by C.
% * ---------------------------------------------------------------------------
% */
%expiry_date(Date) :-
%  g6(OutputSeed),           /* read seed */
%  g5(EncryptedDate),        /* read encrypted date */
%  x1(EncryptedDate, Date, OutputSeed). /* decrypt */
%
%/* ---------------------------------------------------------------------------
% * tool_version(Version)
% *
% * Decrypts the tool version number
% * ---------------------------------------------------------------------------
% */
%tool_version(Version) :-
%  z1(EncryptedVersion, Seed),  /* read encrypted version and seed */
%  x1(EncryptedVersion, Version, Seed). /* decrypt */
%
%/* ---------------------------------------------------------------------------
% * write_non_blank_line(Line)                Codename: g10
% *
% * If Line is non-blank then write it to the current output file.
% * ---------------------------------------------------------------------------
% */
%g10(Line) :-
%  Line = '', !.
%g10(Line) :-
%  write(Line), nl.
%
%
%/* ---------------------------------------------------------------------------
% * write_banner
% *
% * For each line of the customer banner which is non-empty, write it to
% * the current output file
% * ---------------------------------------------------------------------------
% */
%write_banner :-
%  g7_1(Line1),   /* read first banner line */
%  g10(Line1),    /* test whether non-blank and write */
%  !,
%  g7_2(Line2),   /* read second banner line */
%  g10(Line2),    /* test whether non-blank and write */
%  !,
%  g7_3(Line3),   /* read third banner line */
%  g10(Line3).    /* test whether non-blank and write */
%
%/* ---------------------------------------------------------------------------
% * centre_non_blank_line(Line)                Codename: g11
% *
% * If Line is non-blank then write it to the current output file in the
% * middle of the row. (Assume a row is 80 columns wide.)
% * ---------------------------------------------------------------------------
% */
%g11(Line) :-
%  Line = '', !.
%g11(Line) :-
%  name(Line, LineStr), length(LineStr, LineLen),
%  Spaces is (80 - LineLen) / 2,
%  tab(Spaces),
%  write(Line), nl.
%
%
%/* ---------------------------------------------------------------------------
% * write_centralised_banner
% *
% * For each line of the customer banner which is non-empty, write it to
% * the current output file. The lines written are centralised on the
% * output. (Assuming a row is 80 columns wide.)
% * ---------------------------------------------------------------------------
% */
%write_centralised_banner :-
%  g7_1(Line1),   /* read first banner line */
```

```
%  g11(Line1),    /* test whether non-blank and write */
%  !,
%  g7_2(Line2),   /* read second banner line */
%  g11(Line2),    /* test whether non-blank and write */
%  !,
%  g7_3(Line3),   /* read third banner line */
%  g11(Line3).    /* test whether non-blank and write */
%REMOVETONEW================================================
/* getlicence                            g1 */
g1 :-
    write('FUDGE: Calling: g1. Result: true'),
    nl,
    true.

/* tool_version                          tool_version */
tool_version(X) :-
    write('FUDGE: Calling: tool_version(X). Result: X=1'),
    nl,
    X=1.

/* warning_date                          warning_date */
warning_date(X) :-
    write('FUDGE: Calling: warning_date(X). Result: X=fudgedate'),
    nl,
    X=fudgedate.

/* expiry_date                           expiry_date */
expiry_date(X) :-
    write('FUDGE: Calling: expiry_date(X). Result: X=fudgedate'),
    nl,
    X=fudgedate.

/* write_banner                          write_banner */
write_banner :-
    write('FUDGE: Calling: write_banner. Result: true'),
    nl,
    true.

/*write_centralised_banner */
write_centralised_banner :-
    write('FUDGE: Calling: write_banner. Result: true'),
    nl,
    true.
%PORT:END:REPLACECODE####################################/\#
```

6.15 File: getlicence_vax.pl

6.16 File: help.pl

6.16.1 Starting Line: 47

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Unfortunately, the predicate 'help' is special in SICSTUS.
%To overcome this the help predicate used internally to
%SPADE-PC is renamed to spadehelp.
%COMMENT===================================================
%'help' :-
%       command_arg(subject,X),
%       !,
%       'help'(X),
%       !.
%'help' :-
%       display_general_help,
%       !.
%REMOVETONEW================================================
spadehelp :-
    command_arg(subject,X),
    !,
    spadehelp(X),
    !.

spadehelp :-
    display_general_help,
    !.
%PORT:END:REPLACECODE####################################/\#
```

6.16.2 Starting Line: 75

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Unfortunately, the predicate 'help' is special in SICSTUS.
%To overcome this the help predicate used internally to
%SPADE-PC is renamed to spadehelp.
%COMMENT===================================================
%'help'(X = Y) :-
%       get_help_identifier(X, XID),
%       get_help_identifier(Y, YID),
%       show_help(XID = YID),
%       !.
%'help'(X) :-
%       atomic(X),
%       !,
%       get_help_identifier(X, XID),
%       (
%          show_help(XID)
%       ;
%          show_help(rules = XID)
%       ),
%       !.
%REMOVETONEW================================================
spadehelp(X = Y) :-
   get_help_identifier(X, XID),
   get_help_identifier(Y, YID),
   show_help(XID = YID),
   !.
spadehelp(X) :-
   atomic(X),
   !,
   get_help_identifier(X, XID),
   (
      show_help(XID)
   ;
      show_help(rules = XID)
   ),
   !.
%PORT:END:REPLACECODE####################################/\#
```

6.17 File: induction.pl

6.18 File: infer2.pl

6.18.1 Starting Line: 51

```
%PORT:BEGIN:REMOVECODE###################################\/#
%This predicate clashes with a built in SICSTUS predicate
%of the same name and functionality. It is defined in
%SICSTUS as:
%
%file_exists(+FileName)
%
%FileName is the name of an existing file or directory.
%COMMENT===================================================
%file_exists(FILE) :- current_disk(FILE, _, _), !.
%PORT:END:REMOVECODE#####################################/\#
```

6.19 File: inferenc2.pl

6.20 File: initialise.pl

6.20.1 Starting Line: 40

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace particular use of POPLOG-PROLOG specific predicate
%current_disk(..., ..., ...) with corresponding SICSTUS
%predicate.
%COMMENT===================================================
%        current_disk(FILE, _, _),
%REMOVETONEW================================================
        file_exists(FILE),
%PORT:END:REPLACECODE####################################/\#
```

6.21 File: initvals.pl

6.21.1 Starting Line: 34

```
%PORT:BEGIN:REPLACECODE##################################\/#
%The following dynamic predicates are asserted rather
%than being declared while in a dynamic setting.
%COMMENT===================================================
%/*** DISPLAY_SUBGOALS_MAX(N) - maximum number of subgoals for rule display ***/
%display_subgoals_max(10).
%
%
%/*** DISPLAY_VAR_FREE_ONLY(FLAG) - on <-> only display if subgoals var-free ***/
%display_var_free_only(off).
%
%
%/*** CASE_POINTER(N) - proof-by-cases depth counter: initially zero ***/
%case_pointer(0).        /*** ADDED 8/3/85 ***/
%
%
%/*** INPUT_FROM_TERMINAL - true if standard input is sys$input ***/
%input_from_terminal.  /* retracted by command in file if not */
%
%
%/*** VC_STANDARDISATION(FLAG) - automatic standardisation status (on/off) ***/
%vc_standardisation(off).
%
%
%/*** STAN_IN_DEDUCE(FLAG) - standardisation in inequals deduction (on/off) ***/
%stan_in_deduce(off).
%
%
%/*** ECHO(FLAG) - echoing of file input (on/off) ***/
%echo(on).
%
%
%/*** AUTO_DONE(FLAG) - automatic multi-level exit of "done" (on/off) ***/
%auto_done(on).
%
%
%/*** SIMPLIFY_IN_INFER(FLAG) - whether "infer" may use simplifier (on/off) ***/
%simplify_in_infer(on).                        /***MODIFIED***/
%
%
%/*** SIMPLIFY_DURING_LOAD(FLAG) - whether "loadvc" may simplify (on/off) ***/
%simplify_during_load(on).              /***MODIFIED***/
%
%
%/*** STANDARDISE_IN_INFER(FLAG) - can "infer" use standardiser? (on/off) ***/
%standardise_in_infer(off).
%
%
%/*** step_number(N) - step number ***/
%step_number(0).
%
%
%/*** Indentation for proof log - should initially be 0 ***/
%indentation(0).
%
%
%/*** Proof frames indentation increment for proof log ***/
%indentation_increment(2).
%
%
%/*** Typechecking(FLAG) -- should normally be ON ***/
%typechecking(on).
%
%
%/*** Prooflog_width(WIDTH) -- maximum columns for proof log ***/
%prooflog_width(0).
%
%
%/*** record_consults(FLAG) -- whether to record rule-file consultations in
%        proof log ***/
```

```
%record_consults(on).
%
%
%/*** inverse_video(X)/normal_video(X) -- for highlighting on screen ***/
%inverse_video([]).
%normal_video([]).
%
%
%/*** typechecking_during_load(FLAG) -- prevent typechecking if VCGenerated ***/
%typechecking_during_load(on).
%
%
%/*** use_subst_rules_for_equality(FLAG) -- allow infer to use rewrite rules ***/
%use_subst_rules_for_equality(on).
%
%
%/*** command_logging(FLAG) -- whether or not to log user's commands/replies ***/
%command_logging(on).                                              /* CFR015 */
%
%
%/*** show_vc_changes(FLAG) -- whether or not to show new/changed hyps/concs ***/
%show_vc_changes(on).
%
%
%/*** auto_newvc(FLAG) -- whether or not to do a newvc automatically ***/
%auto_newvc(on).                                                      /* CFR025 */
%REMOVETONEW===============================================
/*** DISPLAY_SUBGOALS_MAX(N) - maximum number of subgoals for rule display ***/
:-assert(display_subgoals_max(10)).

/*** DISPLAY_VAR_FREE_ONLY(FLAG) - on <-> only display if subgoals var-free ***/
:-assert(display_var_free_only(off)).

/*** CASE_POINTER(N) - proof-by-cases depth counter: initially zero ***/
:-assert(case_pointer(0)).

/*** INPUT_FROM_TERMINAL - true if standard input is sys$input ***/
:-assert(input_from_terminal).

/*** VC_STANDARDISATION(FLAG) - automatic standardisation status (on/off) ***/
:-assert(vc_standardisation(off)).

/*** STAN_IN_DEDUCE(FLAG) - standardisation in inequals deduction (on/off) ***/
:-assert(stan_in_deduce(off)).

/*** ECHO(FLAG) - echoing of file input (on/off) ***/
:-assert(echo(on)).

/*** AUTO_DONE(FLAG) - automatic multi-level exit of "done" (on/off) ***/
:-assert(auto_done(on)).

/*** SIMPLIFY_IN_INFER(FLAG) - whether "infer" may use simplifier (on/off) ***/
:-assert(simplify_in_infer(on)).

/*** SIMPLIFY_DURING_LOAD(FLAG) - whether "loadvc" may simplify (on/off) ***/
:-assert(simplify_during_load(on)).

/*** STANDARDISE_IN_INFER(FLAG) - can "infer" use standardiser? (on/off) ***/
:-assert(standardise_in_infer(off)).

/*** step_number(N) - step number ***/
:-assert(step_number(0)).

/*** Indentation for proof log - should initially be 0 ***/
:-assert(indentation(0)).

/*** Proof frames indentation increment for proof log ***/
:-assert(indentation_increment(2)).

/*** Typechecking(FLAG) -- should normally be ON ***/
:-assert(typechecking(on)).

/*** Prooflog_width(WIDTH) -- maximum columns for proof log ***/
:-assert(prooflog_width(0)).

/*** record_consults(FLAG) -- whether to record rule-file consultations in
        proof log ***/
:-assert(record_consults(on)).

/*** inverse_video(X)/normal_video(X) -- for highlighting on screen ***/
:-assert(inverse_video([])).
:-assert(normal_video([])).

/*** typechecking_during_load(FLAG) -- prevent typechecking if VCGenerated ***/
:-assert(typechecking_during_load(on)).

/*** use_subst_rules_for_equality(FLAG) -- allow infer to use rewrite rules ***/
:-assert(use_subst_rules_for_equality(on)).

/*** command_logging(FLAG) -- whether or not to log user's commands/replies ***/
:-assert(command_logging(on)).

/*** show_vc_changes(FLAG) -- whether or not to show new/changed hyps/concs ***/
:-assert(show_vc_changes(on)).

/*** auto_newvc(FLAG) -- whether or not to do a newvc automatically ***/
:-assert(auto_newvc(on)).
%PORT:END:REPLACECODE####################################/\#
```

6.22 File: interrupt.pl

6.23 File: listthm.pl

6.24 File: loadlib.pl

6.25 File: loadvc5.pl

6.25.1 Starting Line: 123

```
%PORT:BEGIN:NEWCODE#####################################\/#
%A new line is inserted to give a cleaner command prompt.
%In SICSTUS, reads at points other than the first column
%seem to cause the prompt to disappear.
%COMMENT=====================================================
        nl,
%PORT:END:NEWCODE#######################################/\#
```

6.25.2 Starting Line: 270

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace POPLOG-POPLOG specific code for file access
%with the SICSTUS equivalent.
%COMMENT=====================================================
%checkfilesexist(ordinary) :-
%       vcgfile_name(VCG),
%       (
%          current_disk(VCG, _, _)
%       ;
%          tell_off(vcg)                                      /* CFR048 */
%       ),
%       !,
%       fdlfile_name(FDL),
%       (
%          current_disk(FDL, _, _)
%       ;
%          tell_off(fdl)                                      /* CFR048 */
%       ),
%       !.
%checkfilesexist(resume) :-
%       csvfile_name(CSV),
%       (
%          current_disk(CSV, _, _)
%       ;
%          tell_off(csv)                                      /* CFR048 */
%       ),
%       !.
%REMOVETONEW=================================================
checkfilesexist(ordinary) :-
    vcgfile_name(VCG),
    (
          file_exists(VCG)
       ;
          tell_off(vcg)
    ),
    !,

    fdlfile_name(FDL),
    (
          file_exists(FDL)
       ;
          tell_off(fdl)
    ),
    !.

checkfilesexist(resume) :-
    csvfile_name(CSV),
    (
          file_exists(CSV)
       ;
          tell_off(csv)
    ),
    !.
%PORT:END:REPLACECODE####################################/\#
```

6.25.3 Starting Line: 325

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace POPLOG-PROLOG specific code for file access
%with the SICSTUS equivalent.
%COMMENT=====================================================
%tell_off(TYPE) :-
%       put(7),
%       nl,
%       write('No .'),
%       write(TYPE),
%       write(' file of this name exists.'),
%       nl,
%       nl,
%       write('List of .'),
%       write(TYPE),
%       write(' files in current region:'),
%       nl,
%       name(TYPE, TL),
%       append("*.", TL, FL),
%       name(FILE, FL),
%       (
%          current_disk(FILE, _, _)
%       ;
%          write('     <THERE ARE NONE>'),
%          nl,
%          !,
%          fail
%       ),
%       !,
%       current_disk(FILE, [_, _, _, _, NAME, _, _], _),
%       tab(5),
%       write(NAME),
%       nl,
%       fail.
%REMOVETONEW=================================================
tell_off(Extension):-
    beep,
    nl,
    write('No .'),
    write(Extension),
    write(' file of this name exists.'),
    nl,
    nl,
    write('List of .'),
    write(Extension),
    write(' files in current region:'),
    nl,

    name(Extension, ExtensionCharList),
    append(".", ExtensionCharList, DotExtensionCharList),

    %Determine the current working directory.
    working_directory(Dir,Dir),
    %Find list of all files.
```

```
    directory_files(Dir,FileAtomList),
    %Find list of files with appropriate extension.
    filesWithExtension(FileAtomList,
                       DotExtensionCharList,
                       FilesMatchingList),
    (
          %No matching files found.
          FilesMatchingList=[],
          write('     <THERE ARE NONE>')
        ;
          %Some matching files found.
          displayListOfFiles(FilesMatchingList)
    ),

    %Copy cut and fail behaviour of tell_off.
    !,
    fail.

displayListOfFiles([]).

displayListOfFiles([H_FilesMatching | T_FilesMatchingList]):-
    %tab is obsolescent in SICSTUS - but still works.
    tab(5),
    write(H_FilesMatching),
    nl,
    displayListOfFiles(T_FilesMatchingList).

filesWithExtension([], _, []).

filesWithExtension([H_InFileAtom | T_InFileAtomList],
                   DotExtensionCharList,
                   [H_OutFileAtom | T_OutFileAtomList]):-
    name(H_InFileAtom, H_InFileCharList),
    %Check this has suffix looking for.
    suffix(DotExtensionCharList,H_InFileCharList),

    %Does have suffix, record this file, minus the suffix.
    append(JustNameCharList, DotExtensionCharList, H_InFileCharList),
    name(JustNameAtom, JustNameCharList),
    H_OutFileAtom=JustNameAtom,

    filesWithExtension(T_InFileAtomList,
                       DotExtensionCharList,
                       T_OutFileAtomList).

filesWithExtension([_ | T_InFileAtomList],
                   DotExtensionCharList,
                   OutFileAtomList):-
    filesWithExtension(T_InFileAtomList,
                       DotExtensionCharList,
                       OutFileAtomList).
%PORT:END:REPLACECODE####################################/\#
```

6.25.4 Starting Line: 1250

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Remove this call for replacement code to use the
%post processor that deals with the spaces after for_all
%and for_some.
%COMMENT=================================================
%              eread(FORM),                                        /* CFR034 */
%REMOVETONEW=============================================
              readFormula(FORM),
%PORT:END:REPLACECODE####################################\/#
```

6.25.5 Starting Line: 1271

```
%PORT:BEGIN:NEWCODE######################################\/#
%Replacement code to post processes read in formula,
%dealing with spaces after for_all and for_some.
%COMMENT=================================================
readFormula(CookedFormula):-
    %Read formula, and consume layout.
    %The ideal:
    %read_term(RawFormula, [consume_layout(true)]),
    %Requires SICSTUS 3.9. The following is used instead:
    read(RawFormula),
    skip_line,

    fixTerm(RawFormula,CookedFormula),
    echo_term(CookedFormula),
    !.

%convert: for_all (A)  to for_all(A) and
%        for_some (A) to for_some(A).
%This involves some term manipulation where A is
%a tuple (X,...,Y).
fixTerm(RawTerm,CookedTerm):-
    RawTerm=..RawTermList,
    fixExternalFunctor(RawTermList,CookedTerm),!.

%Working with functors.
fixExternalFunctor([H_Functor | T_RawArgList], CookedTerm):-
    (
          H_Functor = for_all
        ;
          H_Functor = for_some
    ),

    %The for_all and for_some operators must have one argument.
    [OneRawArg] = T_RawArgList,

    %This argument may be a tuple (a,b,...) or
    %something else. If a tuple, it is made a
    %list of arguments in the functor.
    %Otherwise the structure is made the single
    %argument.
    (
          functor(OneRawArg, ',', _),
          OneRawArg =.. [_|RawArgList]
        ;
          RawArgList = [OneRawArg]
    ),

    %Process this list of arguments.
    fixExternalArguments(RawArgList, CookedArgList),
    ModifiedFunctorArgList = [H_Functor | CookedArgList],
    CookedTerm =.. ModifiedFunctorArgList.

fixExternalFunctor([H_Functor | T_RawArgList], CookedTerm):-
    fixExternalArguments(T_RawArgList, CookedArgList),
    ModifiedFunctorArgList=[H_Functor | CookedArgList],
    CookedTerm =.. ModifiedFunctorArgList.

%Working with arguments.
fixExternalArguments([], []).

fixExternalArguments([H_RawArg | T_RawArgList],
                     [H_CookedArg | T_CookedArgList]):-
    H_RawArg=..RawTermList,
    fixExternalFunctor(RawTermList, H_CookedArg),
    fixExternalArguments(T_RawArgList, T_CookedArgList).
%PORT:END:NEWCODE########################################/\#
```

6.25.6 Starting Line: 1884

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace use of get0(...) with the new
%poplogPrologGetCharacterCode(...).
%COMMENT=================================================
%    get0(C),
%REMOVETONEW=============================================
    poplogPrologGetCharacterCode(C),
%PORT:END:REPLACECODE####################################\/#
```

6.26 File: lvc.pl

6.27 File: machine.pl

6.28 File: make_vax.pl

6.29 File: myspy.pl

6.30 File: newrules.pl

6.31 File: newvc.pl

6.31.1 Starting Line: 142

```
%PORT:BEGIN:REMOVECODE###################################\/#
%Use SICSTUS built in last predicate instead.
%The SICSTUS manual describes this as:
%
%last(?List, ?Last)
%Last is the last element in List.
%COMMENT=================================================
%last([N],N) :- !.
%last([X|Y],Z) :- last(Y,Z), !.
%PORT:END:REMOVECODE#####################################/\#
```

6.32 File: prooflogs.pl

6.32.1 Starting Line: 589

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Swap POPLOG-PROLOG fast_setof for SICSTUS setof.
%COMMENT=================================================
%        fast_setof(X, ruleused(X), U),
%REMOVETONEW=============================================
        setof(X, ruleused(X), U),
%PORT:END:REPLACECODE####################################/\#
```

6.32.2 Starting Line: 604

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Swap POPLOG-PROLOG fast_setof for SICSTUS setof.
%COMMENT=================================================
%        fast_setof(X, ruleused_this_session(X), U),
%REMOVETONEW=============================================
        setof(X, ruleused_this_session(X), U),
%PORT:END:REPLACECODE####################################/\#
```

6.32.3 Starting Line: 872

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace POPLOG-PROLOG div with corresponding SICSTUS //.
%COMMENT=================================================
%        I1 is I div 10,
%REMOVETONEW=============================================
        I1 is I // 10,
%PORT:END:REPLACECODE####################################/\#
```

6.33 File: quantif.pl

6.33.1 Starting Line: 245

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Swap POPLOG-PROLOG fast_bagof for SICSTUS bagof.
%COMMENT=================================================
%    fast_bagof(Q, qvar(Q), LIST),
%Use SICSTUS bagof.
%REMOVETONEW=============================================
     bagof(Q, qvar(Q), LIST),
%PORT:END:REPLACECODE####################################\/#
```

6.34 File: records2.pl

6.35 File: repall.pl

6.36 File: replace2.pl

6.36.1 Starting Line: 1213

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Swap POPLOG-PROLOG fast_bagof for SICSTUS bagof.
%COMMENT===================================================
%        (
%           fast_setof(Subs1, (hyp(N, H), find_subs(H, Subs1)), SubsList1)
%        ;
%           SubsList1 = []
%        ),
%        !,
%        (
%           fast_setof(Subs2, (conc(N, C), find_subs(C, Subs2)), SubsList2)
%        ;
%           SubsList2 = []
%        ),
%REMOVETONEW===============================================
        (
           setof(Subs1, (hyp(N, H), find_subs(H, Subs1)), SubsList1)
        ;
           SubsList1 = []
        ),
        !,
        (
           setof(Subs2, (conc(N, C), find_subs(C, Subs2)), SubsList2)
        ;
           SubsList2 = []
        ),
%PORT:END:NEWCODE######################################/\#
```

6.36.2 Starting Line: 1273

```
%PORT:BEGIN:REMOVECODE###################################\/#
%DEBUG
%A bug emerges in this code when used for SICSTUS.
%In replacing subterms of quantified expressions, it
%introduces internal doller annotated values to the main
%problem.
%Always unwrap before replace, or fix this.
%COMMENT===================================================
%PORT:END:REMOVECODE####################################/\#
```

6.37 File: rulefiles.pl

6.38 File: save.pl

6.39 File: semistan.pl

6.40 File: setflags.pl

6.41 File: simplify.pl

6.42 File: simp.pl

6.42.1 Starting Line: 969

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace POPLOG-PROLOG div with corresponding SICSTUS //.
%COMMENT===================================================
%                    Z iss A div B
%REMOVETONEW===============================================
                    Z iss A // B
%PORT:END:REPLACECODE###################################/\#
```

6.42.2 Starting Line: 992

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace POPLOG-PROLOG div with corresponding SICSTUS //.
%COMMENT===================================================
%                    Z iss A div B,                              /*1.4*/
%REMOVETONEW===============================================
                    Z iss A // B,
%PORT:END:REPLACECODE###################################/\#
```

6.43 File: spadepp.pl

6.44 File: spycheck.pl

6.45 File: standard.pl

6.45.1 Starting Line: 523

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Replace POPLOG-PROLOG div with corresponding SICSTUS //.
%COMMENT===================================================
%   Z iss X div Y,
%REMOVETONEW===============================================
   Z iss X // Y,
%PORT:END:REPLACECODE###################################/\#
```

6.46 File: startcode.pl

6.47 File: subgoal.pl

6.48 File: toplevel.pl

6.48.1 Starting Line: 59

```
%PORT:BEGIN:REMOVECODE###################################\/#
%Remove POPLOG exit call.
%COMMENT===================================================
%        dopop('1 -> pop_exit_ok'),
%PORT:END:REMOVECODE####################################/\#
```

6.48.2 Starting Line: 81

```
%PORT:BEGIN:NEWCODE######################################\/#
%A new line is inserted to give a cleaner command prompt.
%In SICSTUS, reads at points other than the first column
%seem to cause the prompt to disappear.
%COMMENT===================================================
        nl,
%PORT:END:NEWCODE######################################/\#
```

6.48.3 Starting Line: 886

```
%PORT:BEGIN:REPLACECODE##################################\/#
%Unfortunately, the predicate 'help' is special in SICSTUS.
%To overcome this the help predicate used internally to
%SPADE-PC is renamed to spadehelp.
%COMMENT===================================================
%        call_once(C),
%        !,
%        (                                                      /* CFR005 */
%           trivial_command(C)                                      /* CFR005 */
%        ;                                                      /* CFR005 */
%           C = exit                                              /* CFR005 */
%        ;                                                      /* CFR005 */
%           C = forceexit
%        ;                                                      /* CFR005 */
%           retractall(recent_save_command_issued)              /* CFR005 */
%        ),                                                      /* CFR005 */
%        !,                                                      /* CFR005 */
%        tidy_up_logfacts,
%        !.
%REMOVETONEW===============================================
        convertHelp(C,CC),
        call_once(CC),
        !,
        (
           trivial_command(CC)
        ;
           CC = exit
        ;
           CC = forceexit
        ;
           retractall(recent_save_command_issued)
        ),
        !,
        tidy_up_logfacts,
        !.

convertHelp(help,spadehelp):-
    !.
convertHelp(C,C):-
    !.
%PORT:END:REPLACECODE###################################/\#
```

6.49 File: traverse.pl

6.50 File: typecheck5.pl

6.51 File: utilities.pl

6.51.1 Starting Line: 82

```
%PORT:BEGIN:NEWCODE######################################\/#
%A new line is inserted to give a cleaner command prompt.
%In SICSTUS, reads at points other than the first column
%seem to cause the prompt to disappear.
%COMMENT===================================================
        nl,
%PORT:END:NEWCODE######################################/\#
```

6.51.2 Starting Line: 94

```
%PORT:BEGIN:REPLACECODE##################################\/#
%In SICSTUS read(...) does not consume white space. To do
%this it must be made explicit.
%COMMENT===================================================
%             read(XX),                                          /* CFR015 */
%REMOVETONEW===============================================
    %The ideal:
    %read_term(XX, [consume_layout(true)]),
    %Requires SICSTUS 3.9. The following is used instead:
    read(XX),
    skip_line,
%PORT:END:REPLACECODE###################################\/#
```

6.51.3 Starting Line: 235

```
%PORT:BEGIN:REMOVECODE###################################\/#
%Use SICSTUS built in append predicate. This is likely
%to be much more efficient. The SICSTUS manual describes
%its append/3 as:
%
%append(?Prefix, ?Suffix, ?Combined)
%
```

```
%Combined is the combined list of the elements in
%Prefix followed by the elements in Suffix. It can be
%used to form Combined or it can be used to find Prefix
%and/or Suffix from a given Combined.
%COMMENT=====================================================
%/*** APPEND(L1,L2,LR) - deterministically join L1 & L2 to get LR ***/
%append([],X,X) :- !.
%append([H|X],Y,[H|Z]) :- append(X,Y,Z), !.

%PORT:END:REMOVECODE#######################################/\#
```

6.51.4 Starting Line: 253

```
%PORT:BEGIN:REMOVECODE#####################################\/#
%The space between atom| |(arguments) is
%not allowed in SICSTUS.
%The code using this predicate is also removed,
%thus no replacment is required. Note that
%the new numericAtomMonthToString/2 is similar
%but is indexed by numbers rather than strings.
%COMMENT=====================================================
%/* On NT, we get months as two-character numeric strings, which need to */
%/* be converted into 3-character month names as follows: */
%numeric_month_to_string ("01", "JAN").
%numeric_month_to_string ("02", "FEB").
%numeric_month_to_string ("03", "MAR").
%numeric_month_to_string ("04", "APR").
%numeric_month_to_string ("05", "MAY").
%numeric_month_to_string ("06", "JUN").
%numeric_month_to_string ("07", "JUL").
%numeric_month_to_string ("08", "AUG").
%numeric_month_to_string ("09", "SEP").
%numeric_month_to_string ("10", "OCT").
%numeric_month_to_string ("11", "NOV").
%numeric_month_to_string ("12", "DEC").
%PORT:END:REMOVECODE#######################################/\#
```

6.51.5 Starting Line: 277

```
%PORT:BEGIN:REPLACECODE####################################\/#
%The date facilities of POPLOG-PROLOG are quite differnt to
%those in SICSTUS. They are replaced with POPLOG code.
%
%This predicate extracts the date from the operating system
%and presents it in the following form:
%DATE=<space padded day number>-
%     <month 3 char abbrevation>-
%     <year as 4 char number>
%TIME=<2 digit hours>:<2 digit minutes>:<2 digit seconds>
%
%For example: fetch_date_and_time( 1-OCT-2001, 00:48:45)
%
%SICSTUS has a predicate portable accross different
%operating systems for accessing this information.
%However, the information is in a differnt format.
%
%The key predicate is: datime(-Datime)
%From library: library(system).
%
%Datime is a timestamp of the form
%datime(Year,Month,Day,Hour,Min,Sec) containing the
%current date and time. All fields are integers.
%
%Previously this predicate had an unusual formats catch-all
%facility. If the SICSTUS code works, it should always work
%accross different platforms. Thus the actions to take in a
%catch all situation are rather unclear. It is left out.
%COMMENT=====================================================
/*** fetch_date_and_time(DATE, TIME) - get date & time from system call ***/
%fetch_date_and_time(DATE, TIME) :-
%       machine(vax),                                           /* CFR048 */
%       prolog_eval(sysdaytime([]), DAYTIME),
%       name(DAYTIME, DAYTIMELIST),
%       gen_append([D|DATELIST], [32|TIMELIST], DAYTIMELIST),
%       name(DATE, [D|DATELIST]),
%       name(TIME, TIMELIST),
%       !.
%fetch_date_and_time(DATE, TIME) :-                          /* CFR048 */
%       machine(sun),                                           /* CFR048 */
%       prolog_eval(sysdaytime([]), DAYTIME),                   /* CFR048 */
%       name(DAYTIME, [_, _, _, 32, M1, M2, M3, 32, D1, D2,    /* CFR048 */
%                      32, H1, H2, 58, Mi1, Mi2, 58, S1, S2,    /* CFR048 */
%                      32 | REST]),                            /* CFR048 */
%       gen_append(_, [32, Y1, Y2, Y3, Y4], REST),             /* CFR048 */
%       toupperstring([M1,M2,M3], [MU1,MU2,MU3]),  /* C1.41 added */
%       /* C1.41: next line altered to use MU? instead of M? */
%       name(DATE, [D1,D2,45,MU1,MU2,MU3,45,Y1,Y2,Y3,Y4]),      /* CFR048 */
%       name(TIME, [H1, H2, 58, Mi1, Mi2, 58, S1, S2]),             /* CFR048 */
%       !.                                                     /* CFR048 */
%
%fetch_date_and_time(DATE, TIME) :-
%    machine(nt),
%    prolog_eval(sysdaytime([]), DAYTIME),
%    name(DAYTIME, [D1, D2, _, M1, M2, _, Y1, Y2, Y3, Y4,
%              32, H1, H2, 58, Mi1, Mi2, 58, S1, S2]),
%       numeric_month_to_string([M1, M2], [MU1, MU2, MU3]),
%    name(DATE, [D1,D2,45,MU1,MU2,MU3,45,Y1,Y2,Y3,Y4]),
%    name(TIME, [H1, H2, 58, Mi1, Mi2, 58, S1, S2]),
%    !.
%
%fetch_date_and_time(DATE, '00:00:00') :-                   /* CFR048 */
%       prolog_eval(sysdaytime([]), DATE),                     /* CFR048 */
%       !.       /* Unusual formats catch-all */                      /* CFR048 */
%REMOVETONEW=================================================
fetch_date_and_time(DATE6, TIME6) :-
    datime(Datime),
    datime(Year,Month,Day,Hour,Min,Sec)=Datime,

    %Year always has 4 digits.
    %(Until the year 10000, anyway..)
    %The number is used intact.
    name(Year, YearAsCharList),

    %Month may have 1 or 2 digits. (1-9 and 10-12)
    %A caps three letter name is used.
    numericAtomMonthToString(Month, MonthAsCharList),

    %Day may have 1 or 2 digits. (1-9 and 10-31)
    %The number is space padded.
    lessHundredPaddedName(Day, 32, DayAsCharList),

    %Hour may have 1 or 2 digits. (0-9 and 10-59)
    %The number is zero padded.
    lessHundredPaddedName(Hour, 48, HourAsCharList),

    %Min may have 1 or 2 digits. (0-9 and 10-59)
    %The number is zero padded.
    lessHundredPaddedName(Min, 48, MinAsCharList),

    %Sec may have 1 or 2 digits. (0-9 and 10-59)
    %The number is zero padded.
    lessHundredPaddedName(Sec, 48, SecAsCharList),

    %Build Date
    DATE1=DayAsCharList,
    append(DATE1,[45],DATE2),
    append(DATE2,MonthAsCharList,DATE3),
    append(DATE3,[45],DATE4),
    append(DATE4,YearAsCharList,DATE5),
    name(DATE6,DATE5),

    %Build Time
    TIME1=HourAsCharList,
    append(TIME1,[58],TIME2),
    append(TIME2,MinAsCharList,TIME3),
    append(TIME3,[58],TIME4),
    append(TIME4,SecAsCharList,TIME5),
    name(TIME6,TIME5),

    %Mirror previous cut behaviour.
    !.

%Convert Number known to be less than a hundred
%into a character list containig two elements.
%If Number is less than 10, the first element
%will be padded with the provided atom.
lessHundredPaddedName(Number, Padding, NumberAsCharList):-
    (
          Number < 10,
          name(Number, SmallNumberAsCharList),
          NumberAsCharList = [Padding | SmallNumberAsCharList]
        ;
          %Known: Month >= 10
          name(Number, NumberAsCharList)
    ).

%Return the three leter name for different
%month numbers.
numericAtomMonthToString(1,  "JAN").
numericAtomMonthToString(2,  "FEB").
numericAtomMonthToString(3,  "MAR").
numericAtomMonthToString(4,  "APR").
numericAtomMonthToString(5,  "MAY").
numericAtomMonthToString(6,  "JUN").
numericAtomMonthToString(7,  "JUL").
numericAtomMonthToString(8,  "AUG").
numericAtomMonthToString(9,  "SEP").
numericAtomMonthToString(10, "OCT").
numericAtomMonthToString(11, "NOV").
numericAtomMonthToString(12, "DEC").
%PORT:END:NEWCODE##########################################/\#
```

6.51.6 Starting Line: 466

```
%PORT:BEGIN:NEWCODE########################################\/#
%A new line is inserted to give a cleaner command prompt.
%In SICSTUS, reads at points other than the first column
%seem to cause the prompt to disappear.
%COMMENT=====================================================
        nl,
%PORT:END:NEWCODE##########################################/\#
```

6.51.7 Starting Line: 513

```
%PORT:BEGIN:NEWCODE########################################\/#
%Get a character code as in POPLOG-PROLOG. The only
%difference between POPLOG-PROLOG get0(...) and SICSTUS
%get_code(...) is the return of 26 rather than -1 on
%end of file.
%COMMENT=====================================================
poplogPrologGetCharacterCode(PoplogCharacterCode):-
        get_code(SicstusCharacterCode),
        (
              %Returns -1 on end-of-file.
              %POPLOG-PROLOG uses the number 26
              %(ascii end of file).
              SicstusCharacterCode=(-1),
              PoplogCharacterCode=26
            ;
              PoplogCharacterCode=SicstusCharacterCode
        ),
        !.
%PORT:END:NEWCODE##########################################/\#
```

6.51.8 Starting Line: 537

```
%PORT:BEGIN:REPLACECODE####################################\/#
%A single change is made to this predicate to replace
%the use of get0(...). The POPLOG-PROLOG behaviour of this
%predicate is slightly different to that seen in SICSTUS.
%A new predicate poplogPrologGetCharacterCode(...) is used
%to gain the behaviour of POPLOG-PROLOG get0(...).
%COMMENT=====================================================
%       get0(CH),                                                  /* CFR014 */
%REMOVETONEW=================================================
        poplogPrologGetCharacterCode(CHAR).
%PORT:END:REPLACECODE######################################/\#
```

6.51.9 Starting Line: 879

```
%PORT:BEGIN:REMOVECODE####################################\/#
%Use SICSTUS built in sublist instead.
%The SICSTUS manual describes its sublist/2 as:
%
%sublist(?Sub, ?List)
%
%Sub contains some of the elements of List, in the same order.
%sublist(SUB, LIST) :- append(SUB, _, LIST).
%sublist(SUB, [_|LIST]) :- sublist(SUB, LIST).
%COMMENT=====================================================
%/*** sublist(SUB, LIST) -- SUB is a sublist of LIST ***/
%sublist(SUB, LIST) :- append(SUB, _, LIST).
%sublist(SUB, [_|LIST]) :- sublist(SUB, LIST).
%PORT:END:REMOVECODE#######################################/\#
```

6.51.10 Starting Line: 1067

```
%PORT:BEGIN:REMOVECODE####################################\/#
%Use SICSTUS built in reverse instead.
%The SICSTUS manual describes its reverse/2 as:
%
%reverse(?List, ?Reversed)
%
%Reversed has the same elements as List but in a
%reversed order.
%COMMENT=====================================================
%/*** reverse(LIST, REVERSED_LIST) - reverse a list ***/               /* CFR049 */
%reverse(LIST, REVERSED_LIST) :-                                       /* CFR049 */
%       !,                                                     /* CFR049 */
%       reverse_acc(LIST, [], REVERSED_LIST),                  /* CFR049 */
%       !.                                                     /* CFR049 */
%
%
%/* reverse_acc(LIST, ACC, REVLIST) - reverse + accumulator */  /* CFR049 */
%reverse_acc([HEAD|TAIL], ACCUMULATOR, LIST) :-                /* CFR049 */
%       !,                                                     /* CFR049 */
%       reverse_acc(TAIL, [HEAD|ACCUMULATOR], LIST),           /* CFR049 */
%       !.                                                     /* CFR049 */
%reverse_acc([], REVERSED_LIST, REVERSED_LIST) :- !.           /* CFR049 */
%PORT:END:REMOVECODE#######################################/\#
```

7 declarations.pl

```prolog
%######################################################################
%The NuSPADE Project
%
%Bill J Ellis
%======================================================================
%declar.pl
%----------------------------------------------------------------------
%The single location for all of NuSPADE operator declerations.
%######################################################################


%**********************************************************************
%Initilisation*********************************************************
%**********************************************************************


%##Operator declerations############################################\/##
%:- systemOp(+Precedence, +Type, +Name).
%
%==Description=====================================================\/==
%Prolog operators are global.
%It is necessary to ensure that the operators from each system
%within NUSPADE are consistient.
%
%The purpose of this predicate is to document the operators seen
%within systems which must be supported by NUSPADE.
%
%Note that this predicate serves no computational role. it is
%series of formal comments.
%######################################################################

systemOp(poplog, 10, xfy, ^).
systemOp(poplog, 25, yfx, *).
systemOp(poplog, 25, yfx, /).
systemOp(poplog, 25, yfx, div).
systemOp(poplog, 25, yfx, mod).
systemOp(poplog, 31, yfx, /\).
systemOp(poplog, 31, yfx, +).
systemOp(poplog, 31, yfx, -).
systemOp(poplog, 31, yfx, \/).
systemOp(poplog, 40, xfx, is).
systemOp(poplog, 40, xfx, <).
systemOp(poplog, 40, xfx, =<).
systemOp(poplog, 40, xfx, ==).
systemOp(poplog, 40, xfx, =).
systemOp(poplog, 40, xfx, =:=).
systemOp(poplog, 40, xfx, =..).
systemOp(poplog, 40, xfx, =\=).
systemOp(poplog, 40, xfx, >=).
systemOp(poplog, 40, xfx, >).
systemOp(poplog, 40, xfx, @<).
systemOp(poplog, 40, xfx, @=<).
systemOp(poplog, 40, xfx, @>=).
systemOp(poplog, 40, xfx, @>).
systemOp(poplog, 40, xfx, \==).
systemOp(poplog, 40, xfx, \=).
systemOp(poplog, 253, xfy, ->).
systemOp(poplog, 249, fy, \+).
systemOp(poplog, 252, xfy, ',').
systemOp(poplog, 254, xfy, ;).
systemOp(poplog, 255, fx, :-).
systemOp(poplog, 255,  fx, ?-).
systemOp(poplog, 255, xfx, -->).
systemOp(poplog, 255, xfx, :-).

systemOp(sicstus, 200, xfy, ^).
systemOp(sicstus, 200, xfx, **).
systemOp(sicstus, 500, yfx, #).
systemOp(sicstus, 200, fy, -).
systemOp(sicstus, 200, fy, +).
systemOp(sicstus, 400, yfx, *).
systemOp(sicstus, 400, yfx, /).
systemOp(sicstus, 400, yfx, mod).
systemOp(sicstus, 400, yfx, /\).
systemOp(sicstus, 500, yfx, +).
systemOp(sicstus, 500, yfx, -).
systemOp(sicstus, 500, yfx, \/).
systemOp(sicstus, 550, xfy, :).
systemOp(sicstus, 700, xfx, is).
systemOp(sicstus, 700, xfx, <).
systemOp(sicstus, 700, xfx, =<).
systemOp(sicstus, 700, xfx, ==).
systemOp(sicstus, 700, xfx, =).
systemOp(sicstus, 700, xfx, =:=).
systemOp(sicstus, 700, xfx, =..).
systemOp(sicstus, 700, xfx, =\=).
systemOp(sicstus, 700, xfx, >=).
systemOp(sicstus, 700, xfx, >).
systemOp(sicstus, 700, xfx, @<).
systemOp(sicstus, 700, xfx, @=<).
systemOp(sicstus, 700, xfx, @>=).
systemOp(sicstus, 700, xfx, @>).
systemOp(sicstus, 700, xfx, \==).
systemOp(sicstus, 700, xfx, \=).
systemOp(sicstus, 900, fy, \+).
systemOp(sicstus, 1100, xfy, ->).
systemOp(sicstus, 1000, xfy, ',').
systemOp(sicstus, 1100, xfy, ;).
systemOp(sicstus, 1200, fx, :-).
systemOp(sicstus, 1200, fx, ?-).
systemOp(sicstus, 1200, xfx, -->).
systemOp(sicstus, 1200, xfx, :-).

systemOp(spadepc, 20, xfx, **).
systemOp(spadepc, 20, xf, ~).
systemOp(spadepc, 20, yfx, #).
systemOp(spadepc, 21, fx, +).
systemOp(spadepc, 21, fx, -).
systemOp(spadepc, 25, yfx, *).
systemOp(spadepc, 25, yfx, /).
systemOp(spadepc, 25, yfx, div).
systemOp(spadepc, 25, yfx, mod).
systemOp(spadepc, 25, yfx, @).
systemOp(spadepc, 25, yfx, /\).
systemOp(spadepc, 31, yfx, +).
systemOp(spadepc, 31, yfx, -).
systemOp(spadepc, 31, yfx, \/).
systemOp(spadepc, 31, yfx, \).
systemOp(spadepc, 40, fy, set).
systemOp(spadepc, 40, yfx, <>).
systemOp(spadepc, 40, xfx, subset_of).
systemOp(spadepc, 40, yfx, strict_subset_of).
systemOp(spadepc, 40, xfx, iss).
systemOp(spadepc, 40, yfx, less_than).
systemOp(spadepc, 40, yfx, <=).
systemOp(spadepc, 100, yfx, in).
systemOp(spadepc, 100, yfx, not_in).
systemOp(spadepc, 100, yfx, '..').
systemOp(spadepc, 200, fy, not).
systemOp(spadepc, 210, yfx, and).
systemOp(spadepc, 220, yfx, or).
systemOp(spadepc, 220, yfx, xor).
systemOp(spadepc, 230, xfy, ->).
systemOp(spadepc, 230, yfx, requires).
systemOp(spadepc, 235, yfx, <->).
systemOp(spadepc, 238, yfx, ':=').
systemOp(spadepc, 240, fx, consult).
systemOp(spadepc, 240, fx, load).
systemOp(spadepc, 240, yfx, &).
systemOp(spadepc, 240, fx, rule_family).
systemOp(spadepc, 243, yfx, where).
systemOp(spadepc, 245, yfx, by).
systemOp(spadepc, 245, yfx, with).
systemOp(spadepc, 245, yfx, to).
systemOp(spadepc, 245, yfx, may_be_deduced_from).
systemOp(spadepc, 245, yfx, may_be_replaced_by).
systemOp(spadepc, 245, xf, are_interchangeable).
systemOp(spadepc, 245, yfx, if).
systemOp(spadepc, 245, xf, may_be_deduced).
systemOp(spadepc, 247, yfx, =>).
systemOp(spadepc, 247, yfx, using).
systemOp(spadepc, 247, yfx, for).
systemOp(spadepc, 247, yfx, on).
systemOp(spadepc, 250, xfy, :).
systemOp(spadepc, 250, yfx, from).

systemOp(nuspade, 800, fy, for_all).
systemOp(nuspade, 800, fy, for_some).
systemOp(nuspade, 900, xfy, ==>).
systemOp(nuspade, 950, xfy, :::).
systemOp(nuspade, 950, fy, @@@).
systemOp(nuspade, 830, xfy, '[]-->').
systemOp(nuspade, 700, xfx, '===').
%######################################################################


%##Operators########################################################\/##
%:- op(+Precedence, +Type, +Name).
%
%==Description=====================================================\/==
%
%All of the operators are integrated into a single uniform manner.
%Where clashes with operator orderings exist, reasonable comprises
%are made.
%######################################################################

%sicstus overrides poplog.
:-op(200        ,xfy  ,^                  ).

%spadepc is adjusted to sicstus ordering.
:-op(200        ,xf   ,~                  ).

%spadepc is adjusted to sicstus ordering.
:-op(200        ,xfx  ,**                 ).
:-op(200        ,xfx  ,**                 ).
:-op(200        ,yfx  ,'i**i'             ).
:-op(200        ,yfx  ,'r**i'             ).

%sicstus clashes with spadepc.
%Use spadepc adjusted to sicstus ordering.
%
%In SICSTUS X#Y is the bitwise exclusive or of the
%integers X and Y. In SPADE-PC this is used for selecting
%hypothesis and conclusions (h#3, c#1). The SICSTUS
%operator is considered not very valuable, and is
%not even ISO standard. Thus the SPADE-PC
%definition for X#Y is taken instead, changing
%the precedence accordingly.
:-op(200        ,yfx  ,#                  ).

%In normal SICSTUS mode, the following extra operators
%are added:
%:- op(  500,  fx, [ +, - ]).
%:- op(  300, xfx, [ mod ]).
%Although SICSTUS allows fx and fy operators to coexist,
%it is worth noting this unusual behaviour here.

%spadepc is adjusted to sicstus ordering.
:-op(200        ,fy   ,-                  ).

%spadepc is adjusted to sicstus ordering.
:-op(200        ,fy   ,+                  ).

%spadepc is adjusted to sicstus ordering.
:-op(400        ,yfx  ,*                  ).
:-op(400        ,yfx  ,'i*i'              ).
:-op(400        ,yfx  ,'i*r'              ).
:-op(400        ,yfx  ,'r*i'              ).
:-op(400        ,yfx  ,'r*r'              ).

%spadepc is adjusted to sicstus ordering.
:-op(400        ,yfx  ,/                  ).
:-op(400        ,yfx  ,'i/i'              ).
:-op(400        ,yfx  ,'i/r'              ).
```

```
:-op(400        ,yfx  ,'r/i'               ).
:-op(400        ,yfx  ,'r/r'               ).

%spadepc is adjusted to sicstus ordering.
:-op(400        ,yfx  ,div                 ).

%spadepc is adjusted to sicstus ordering.
:-op(400        ,yfx  ,mod                 ).

%spadepc is adjusted to sicstus ordering.
:-op(400        ,yfx  ,@                   ).

%spadepc is adjusted to sicstus ordering.
:-op(400        ,yfx  ,/\                  ).

%spadepc is adjusted to sicstus ordering.
:-op(500        ,yfx  ,+                   ).
:-op(500        ,yfx  ,'i+i'               ).
:-op(500        ,yfx  ,'i+r'               ).
:-op(500        ,yfx  ,'r+i'               ).
:-op(500        ,yfx  ,'r+r'               ).

%spadepc is adjusted to sicstus ordering.
:-op(500        ,yfx  ,-                   ).
:-op(500        ,yfx  ,'i-i'               ).
:-op(500        ,yfx  ,'i-r'               ).
:-op(500        ,yfx  ,'r-i'               ).
:-op(500        ,yfx  ,'r-r'               ).

%spadepc overrides poplog and is adjusted to sicstus ordering.
:-op(500        ,yfx  ,\/                  ).

%Sole spadepc is adjusted to sicstus ordering.
:-op(500        ,yfx  ,\                   ).

%Sole spadepc is adjusted to sicstus ordering.
:-op(700        ,fy   ,set                 ).

%nuspade ordering.
:-op(700        ,xfx  ,'==='               ).

%Sole spadepc is adjusted to sicstus ordering.
:-op(700        ,yfx  ,<>                  ).
:-op(700        ,yfx  ,'i<>i'              ).
:-op(700        ,yfx  ,'r<>r'              ).
:-op(700        ,yfx  ,'e<>e'              ).
:-op(700        ,yfx  ,'o<>o'              ).
:-op(700        ,yfx  ,'a<>a'              ).

%Sole spadepc is adjusted to sicstus ordering.
:-op(700        ,xfx  ,subset_of           ).

%Sole spadepc is adjusted to sicstus ordering.
:-op(700        ,yfx  ,strict_subset_of    ).

%Sole spadepc is adjusted to sicstus ordering.
:-op(700        ,xfx  ,iss                 ).

%Sole spadepc is adjusted to sicstus ordering.
:-op(700        ,yfx  ,less_than           ).

%Sole spadepc is adjusted to sicstus ordering.
:-op(700        ,yfx  ,<=                  ).
:-op(700        ,yfx  ,'i<=i'              ).
:-op(700        ,yfx  ,'i<=r'              ).
:-op(700        ,yfx  ,'r<=i'              ).
:-op(700        ,yfx  ,'r<=r'              ).
:-op(700        ,yfx  ,'e<=e'              ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,is                  ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,<                   ).
:-op(700        ,xfx  ,'i<i'               ).
:-op(700        ,xfx  ,'i<r'               ).
:-op(700        ,xfx  ,'r<i'               ).
:-op(700        ,xfx  ,'r<r'               ).
:-op(700        ,xfx  ,'e<e'               ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,=<                  ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,==                  ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,=                   ).
:-op(700        ,xfx  ,'i=i'               ).
:-op(700        ,xfx  ,'r=r'               ).
:-op(700        ,xfx  ,'e=e'               ).
:-op(700        ,xfx  ,'o=o'               ).
:-op(700        ,xfx  ,'a=a'               ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,=:=                 ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,=..                 ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,=\=                 ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,>=                  ).
:-op(700        ,xfx  ,'i>=i'              ).
:-op(700        ,xfx  ,'i>=r'              ).
:-op(700        ,xfx  ,'r>=i'              ).
:-op(700        ,xfx  ,'r>=r'              ).
:-op(700        ,xfx  ,'e>=e'              ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,>                   ).
:-op(700        ,xfx  ,'i>i'               ).
:-op(700        ,xfx  ,'i>r'               ).
:-op(700        ,xfx  ,'r>i'               ).
:-op(700        ,xfx  ,'r>r'               ).
:-op(700        ,xfx  ,'e>e'               ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,@<                  ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,@=<                 ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,@=<                 ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,@>=                 ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,@>                  ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,\==                 ).

%sicstus overrides poplog.
:-op(700        ,xfx  ,\=                  ).

%spadepc is adjusted to sicstus ordering.
:-op(750        ,yfx  ,in                  ).

%spadepc is adjusted to sicstus ordering.
:-op(750        ,yfx  ,not_in              ).

%spadepc is adjusted to sicstus ordering.
:-op(750        ,yfx  ,..                  ).

%spadepc is adjusted to sicstus ordering.
:-op(800        ,fy   ,not                 ).

%nuspade ordering.
:-op(800        ,fy   ,for_all             ).

%nuspade ordering.
:-op(800        ,fy   ,for_some            ).

%spadepc is adjusted to sicstus ordering.
:-op(810        ,yfx  ,and                 ).

%spadepc is adjusted to sicstus ordering.
:-op(820        ,yfx  ,or                  ).

%spadepc is adjusted to sicstus ordering.
:-op(820        ,yfx  ,xor                 ).

%spadepc overrides sicstus and is adjusted to sicstus ordering.
%
%The operator -> is base in POPLOG and SICSTUS as the
%conditional operator.
%(Where: (P -> Q) ; R is analogous to (if P then Q else R))
%However, SPADE-PC uses -> for implication, with an incompatible
%precedence.
%As changing the SPADE-PC precedence would mean substantial
%rule changes, its precedence must remain. Thus where used
%as a conditional operator brackets may be required.
%In many cases the SPADE-PC precedence is unlikely to require
%any changes. However, if warranted a special -> operator might
%be declared in SICSTUS.
:-op(830        ,xfy  ,->                  ).

%nuspade ordering.
:-op(830        ,xfy  ,-+->                ).

%spadepc is adjusted to sicstus ordering.
:-op(830        ,yfx  ,requires            ).

%spadepc is adjusted to sicstus ordering.
:-op(835        ,yfx  ,<->                 ).

%spadepc is adjusted to sicstus ordering.
:-op(838        ,yfx  ,:=                  ).

%spadepc is adjusted to sicstus ordering.
:-op(840        ,fx   ,consult             ).

%spadepc is adjusted to sicstus ordering.
:-op(840        ,fx   ,load                ).

%spadepc is adjusted to sicstus ordering.
:-op(840        ,yfx  ,&                   ).

%spadepc is adjusted to sicstus ordering.
:-op(840        ,fx   ,rule_family         ).

%spadepc is adjusted to sicstus ordering.
:-op(843        ,yfx  ,where               ).

%spadepc is adjusted to sicstus ordering.
:-op(845        ,yfx  ,by                  ).

%spadepc is adjusted to sicstus ordering.
:-op(845        ,yfx  ,with                ).

%spadepc is adjusted to sicstus ordering.
:-op(845        ,yfx  ,to                  ).

%spadepc is adjusted to sicstus ordering.
:-op(845        ,yfx  ,may_be_deduced_from ).

%spadepc is adjusted to sicstus ordering.
:-op(845        ,yfx  ,may_be_replaced_by  ).

%spadepc is adjusted to sicstus ordering.
:-op(845        ,xf   ,are_interchangeable ).

%spadepc is adjusted to sicstus ordering.
:-op(845        ,yfx  ,if                  ).

%spadepc is adjusted to sicstus ordering.
:-op(845        ,xf   ,may_be_deduced      ).

%spadepc is adjusted to sicstus ordering.
```

```
:-op(847        ,yfx  ,=>                 ).

%spadepc is adjusted to sicstus ordering.
:-op(847        ,yfx  ,using              ).

%spadepc is adjusted to sicstus ordering.
:-op(847        ,yfx  ,for                ).

%spadepc is adjusted to sicstus ordering.
:-op(847        ,yfx  ,on                 ).

%spadepc overrides sicstus and is adjusted to sicstus ordering.
%
%In Sicstus X:Y is used to work with modules.
%In SPADE-PC this is used in rule files, with a different
%precedence ordering. Both of these are key operations.
%For the moment the SPADE-PC precedence is used.
%It may be necessary to dynamically change the precedence
%of this operator.
:-op(950        ,xfy  ,:                  ).

%nuspade ordering.
:-op(950        ,xfy  ,:::                ).

%nuspade ordering.
:-op(950        ,fy   ,@@@                ).

%spadepc is adjusted to sicstus ordering.
:-op(950        ,yfx  ,from               ).

%nuspade ordering.
:-op(900        ,xfx  ,==>                ).

%sicstus overrides poplog.
:-op(900        ,fy   ,\+                 ).

%sicstus overrides poplog.
%
%The ',' operator can not be redeclared in sicstus.
%:-op(1000       ,xfy  ,','                ).

%sicstus overrides poplog.
:-op(1100       ,xfy  ,;                  ).

%sicstus overrides poplog.
:-op(1200       ,fx   ,:-                 ).

%sicstus overrides poplog.
:-op(1200       ,fx   ,?-                 ).

%sicstus overrides poplog.
:-op(1200       ,xfx  ,-->                ).

%sicstus overrides poplog.
:-op(1200       ,xfx  ,:-                 ).
%######################################################################


%######################################################################
%End of file
```