
Maple-PVS Superuser Guide

Version 0.1 ■ December 2005

Clare M. So and Hanne Gottliebsen

Department of Computer Science
Queen Mary, University of London

{cmso,hago}@dcs.qmul.ac.uk

Abstract

Maple is a popular and powerful computer algebra system, but it could not guarantee the correctness of all answers. PVS provides formal proofs to guarantee the correctness of answers, but it cannot be used efficiently without specific knowledge of the system. To combine the advantages of PVS and Maple, an interface is built between the systems so that a Maple user can access the functionalities of PVS. This document discusses the implementation and installation instructions of the interface.

Contents

1	Introduction	1
2	Overview	2
2.1	PVS Verification System	2
2.2	Maple	2
2.3	Maple-PVS Interface	3
3	Implementation	4
3.1	Filter	5
3.2	GUI Front End	5
3.3	Wrapper	6
4	Installing Maple-PVS	7
4.1	System Requirements	7
4.2	Running the Installation Script	8
4.3	Troubleshooting	9
4.3.1	Changing Paths	9
4.3.2	Setting Access Priviledges	10
4.3.3	Compiling Source Code	10
4.3.4	PVS installation	10
5	Frequently Asked Questions	11

Chapter 1

Introduction

Maple is a popular and powerful computer algebra system for solving many mathematical problems, but it lacks of formal methods to ensure the correctness of answers. PVS is a verification system that use formal methods to ensure the correctness of answers, but it cannot be used efficiently without specific knowledge of the system. To combine the advantages of Maple and PVS, we enable a Maple user to access certain functionalities by our Maple-PVS interface. In this interface, a Maple user can access the functionalities of PVS from a Maple session.

This document first present the software architecture and the implementation of Maple-PVS, then it presents the installation and configuration instructions of the software. It is not intended to be a tutorial of the Maple-PVS interface. For information on the usages of the interface, please refer to the *User Guide*.

Chapter 2

Overview

This chapter briefly describes the Maple-PVS interface itself as well as the software components connected by the interface. There are three main components of the Maple-PVS interface: PVS, Maple and the interface (Figure 2.1). PVS and Maple are the two mathematical software packages we are interested in connecting together. The interface's main task is to relay messages between Maple and PVS: All communications between Maple and PVS are always done through the interface. Without the interface, Maple and PVS cannot communicate between each other.

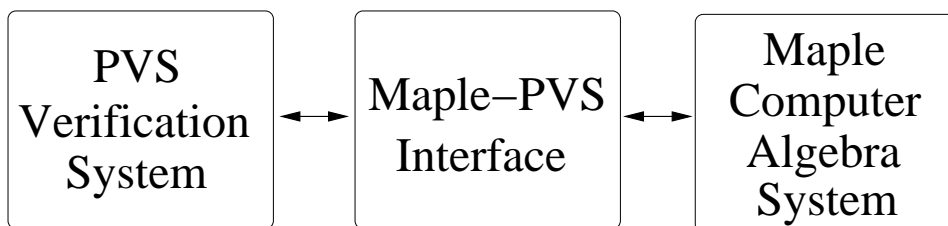


Figure 2.1: Communications between components of the Maple-PVS architecture

2.1 PVS Verification System

PVS Verification System [1] is a tool for specifying and proving theorems. **Please don't be confused with an unrelated Linux/Unix utility named pvs.** Many regular PVS users normally use PVS through (X)Emacs (Figure 2.2). In this implementation of the Maple-PVS interface PVS raw, which is the command line version of PVS, is used. (Figure 2.3).

2.2 Maple

Maple [2] is the computer algebra system (Figure 2.2) to be used in this interface. It does not only perform computations: A programming environment is available. One of

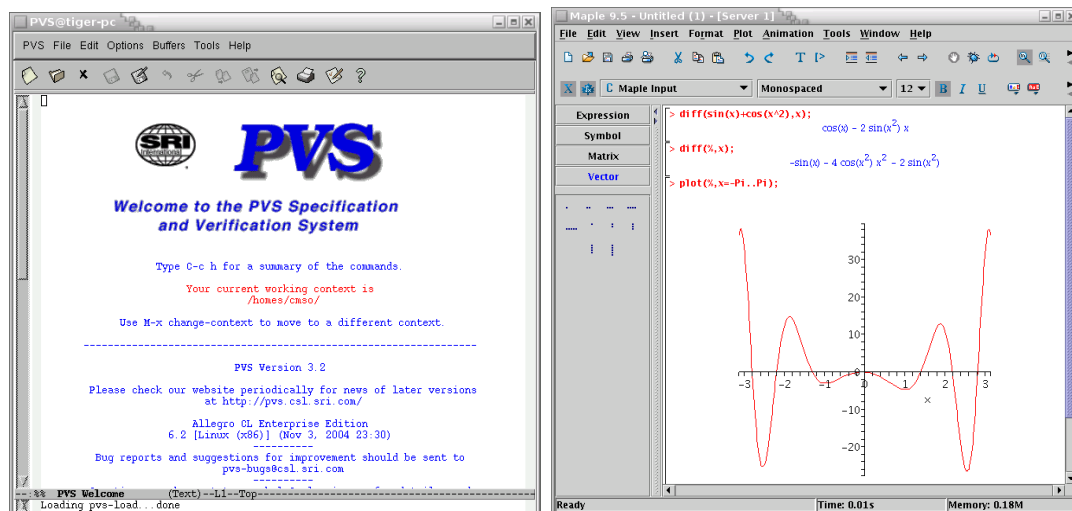


Figure 2.2: A PVS Emacs session welcome screen (left) and a typical (X)Maple session (right)

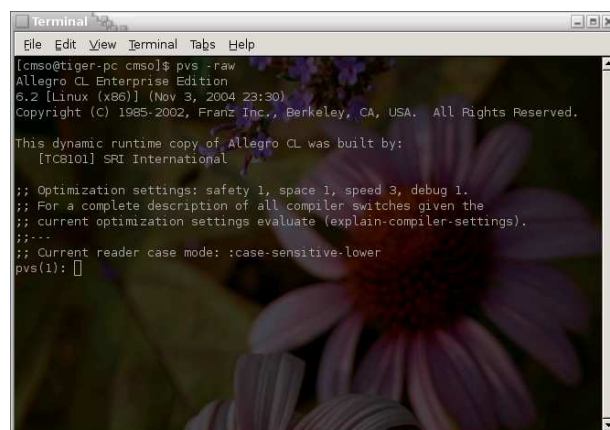


Figure 2.3: A PVS raw session

the functionalities of the programming environment is to allow Maple to access programs written in another language. We are utilizing this feature to enable Maple to be exposed to PVS.

2.3 Maple-PVS Interface

The Maple-PVS Interface is responsible to handle the communications between Maple and PVS. Its tasks include initializing PVS, assembling query commands and filtering PVS messages. The implementation of this interface is discussed in this document.

Chapter 3

Implementation

This chapter discusses the implementation of the Maple-PVS interface. There are three components in the interface: Filter, GUI front end and wrapper (Figure 3.1).

Each component is responsible for relaying, processing or assembling messages for Maple and PVS. PVS commands are first assembled accordingly by the wrapper. Then the wrapper sends the command to PVS via C. PVS processes the commands and generates output. The PVS output messages are first categorized by the filter. Then each message is discarded, or is distributed accordingly to the GUI front end and Maple.

Each section in this chapter describes the functionalities of each component. A list of source files that belong to a particular component can be found along with the description. All source files of the software can be found in `<maplepvs-dir>/src`.

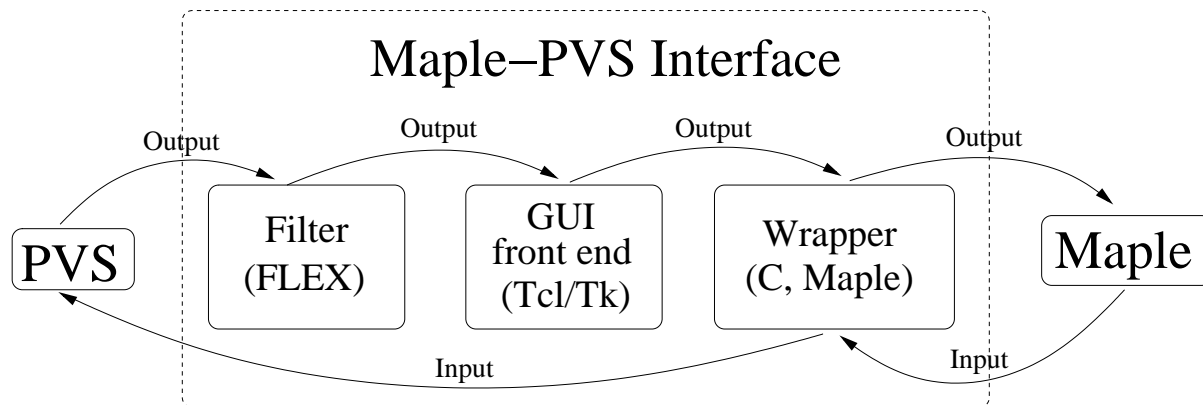


Figure 3.1: Software components and tools used in the Maple-PVS interface's implementation

3.1 Filter

Source file(s) required:	Language:
<ul style="list-style-type: none"> • pvs-filter.lex 	<ul style="list-style-type: none"> • FLEX

This component put PVS output messages into different categories. PVS-raw generates many messages regarding to the state of the system. Not all of the messages are equally important to all users. In PVS-Emacs, the messages are usually either shown in the mini-buffer, displayed in the main window or discarded.

In this component, each type of messages is described by a regular expression. Upon matching a message to a certain description, a two-letter tag is attached before the message. This tag is for identifying the type of the PVS messages. This component does not ignore or discard any messages. Instead, it relays all messages to the GUI front end.

3.2 GUI Front End

Source file(s) required:	Language:
<ul style="list-style-type: none"> • pvs-ctl 	<ul style="list-style-type: none"> • Tcl/Tk

This component does not only provide a simplified graphical front end to Maple-PVS (Figure 3.2), it also distributes messages to the Tcl/Tk window and Maple. The destination of a particular output message depends on the message type specified by the filter. Among the messages to be displayed in the main window, different colours highlight important messages.

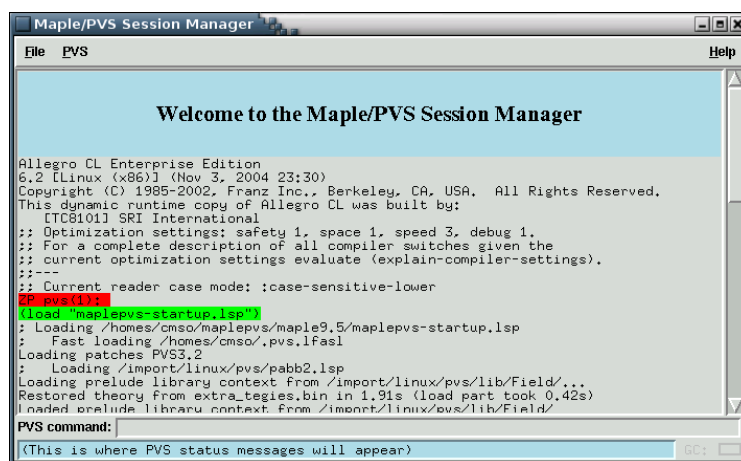


Figure 3.2: GUI front end of Maple-PVS as a Tcl/Tk window.

3.3 Wrapper

Source file(s) required: <ul style="list-style-type: none"> • <code>debug.h</code>, <code>defs.h</code>, <code>err.h</code>, <code>mem.h</code>, <code>proc.h</code>, <code>pvs.h</code>, <code>str.h</code>, <code>defines.h</code>, <code>types.h</code> • <code>err.c</code>, <code>mem.c</code>, <code>proc.c</code>, <code>pvs.c</code>, <code>str.c</code> 	Language: <ul style="list-style-type: none"> • C
<ul style="list-style-type: none"> • <code>pvsmodule.mpl</code> 	<ul style="list-style-type: none"> • Maple Programming Language

Maple accesses the interface via this component. This component's main functionalities are to initialize the interface, assemble PVS commands, get inputs/outputs and shut down the interface

The C functions handle the communications between Maple and rest of the interface. They are responsible for initializing and closing the interface (`proc.c`, `pvs.c`), and relaying messages to/from Maple (`proc.*`).

Normal Maple-PVS users see the interface accessible to them as a Maple package. This package does not only link Maple to the C functions, it also contains procedures to assemble PVS proof commands and view PVS proofs.

Chapter 4

Installing Maple-PVS

This chapter presents the system requirements and the installation instructions of the Maple-PVS interface. Satisfying the system requirements is essential. An installation script is included in the tarball to simplify the steps to check system requirements and configuring the software. If the installation script does not execute successfully, a troubleshooting section is available to guide the manual configuration.

4.1 System Requirements

The system requirements of the Maple-PVS interface are the following:

- Maple Computer Algebra System 9.5 or 10
- PVS ¹ Specification and Verification System 3.2
- Linux/Unix (Example: Debian, Fedora) or Solaris operating system
- gcc
- FLEX (Fast Lexical Analyzer Generator)
- Tcl/Tk

In addition, the installation script (`install.sh`) requires the following:

- bash
- sed

If your system does not satisfy all of the requirements, please install any missing software before installing Maple-PVS. Note that the tarball only includes the software for connecting Maple and PVS. For information about getting a copy of Maple, please see <http://www.maplesoft.com>. For information about obtaining and installing PVS, please see <http://pvs.csl.sri.com/>.

¹Please don't be confused with an unrelated Linux/Unix utility called `pvs`

4.2 Running the Installation Script

The installation script (`install.sh`) is found in the Maple-PVS tarball. Before running the script, you **must** change the `PVS_HOME` variable. It is found near the beginning of the script. This variable contains the path of your PVS installation.

You are now ready to install the Maple-PVS interface. Before running the script, be sure to change directory to the Maple-PVS installation. This directory should be where `install.sh` is located. Run `install.sh`. Upon a successful installation you should see messages similar to the following:

```
$ ./install.sh

*****
Checking system requirements
*****
/homes/cmso/.pvs found
Maple installation detected
gcc detected
flex detected
sed detected

*****
Maple-PVS is now being installed in /homes/cmso/maplepv
*****
Configuring paths in pvsmodule.mpl
Configuring paths in maplepvs-startup.lsp
Configuring paths in pvs-ctl
Configuring paths in pvs.c
Ensuring the users have executing privilege to pvs-ctl
Compiling C source code
/bin/rm -f *~ *.o *.so
/bin/rm -f mwrap_*
/bin/rm -f lex.yy.c
/bin/rm -f pvs-filter
gcc -g -Wall -c -o err.o err.c
gcc -g -Wall -c -o mem.o mem.c
gcc -g -Wall -c -o proc.o proc.c
gcc -g -Wall -c -o pvs.o pvs.c
gcc -g -Wall -c -o str.o str.c
gcc -o maple_pvslib.so -shared -g -Wall err.o mem.o proc.o pvs.o str.o -lutil
flex pvs-filter.lex
gcc -o pvs-filter lex.yy.c -lfl
Done!
```

This script first detects if the system requirements are met. If certain requirements are missing, a warning message is displayed. Then it changes the paths in the source

files and ensures the users to have read or execute privilege for certain files. Lastly, it compiles the C and FLEX source code.

You may be able to use the Maple-PVS interface now even if the installation script gives you some warning messages. For example, you have installed Maple in your system, but the script cannot detect the Maple installation via the `PATH` variable. If you still cannot run the Maple-PVS interface, you should read the troubleshooting guide in the next section and the frequently asked questions in chapter 5.

4.3 Troubleshooting

If you have run `install.sh` and cannot get the interface to run properly, you should consider setting up the interface manually. Before continuing to read this section, please be sure that all of the system requirements described in section 4.1.

4.3.1 Changing Paths

Check if the following paths are changed properly in these specific files:

- `src/pvsmodule.mpl`
 - Line 69: `dir := "/homes/cmso/maplepvs";`
 - * Change the `/homes/cmso/maplepvs` to the location of your Maple-PVS installation
- `src/maplepvs-startup.lsp`
 - Line 1: `(setq *pvs-path* "/import/linux/pvs/")`
 - * Change `/import/linux/pvs/` to the location of your PVS installation
 - Line 4: `(load-prelude-library "/import/linux/pvs/lib/Field")`
 - * Change `/import/linux/pvs/` to the location of your PVS installation
- `src/pvs-ctl`
 - Line 1356: `set PvsHandle [open "| pvs -raw | /homes/cmso/maplepvs/src/pvs-filter" "r+"]`
 - * Change `/homes/cmso/maplepvs/` to the location of your PVS installation
- `src/pvs.c`
 - Line 100: `"/homes/cmso/maplepvs/src/pvs-ctl",`
 - * Change `/homes/cmso/maplepvs/` to the location of your PVS installation

4.3.2 Setting Access Priviledges

- `install.sh`
 - The administrator must have priviledge to execute this file (ie. `chmod u+x install.sh`)
- `src/pvs-ctl`
 - All users must be given execute priviledge
- `src/maple_pvslib.so`, `src/pvs-filter`, `src/maplepvs-startup.lsp`,
`src/pvsmodule.mpl`
 - All users must have read priviledge to these files

4.3.3 Compiling Source Code

A `Makefile` is found under the `src/` directory. Change to this directory. Compile the FLEX and C source files by typing `make`.

4.3.4 PVS installation

The command to execute PVS must be called `pvs`. Any other variant of this name will not work. This executable must be directly accessible from the default `PATH` variable.

Chapter 5

Frequently Asked Questions

1. PVS is behaving strangely. What can I do?
Never execute more than one PVS session in your computer, even if the sessions are from PVS-Emacs, PVS raw or different Maple worksheets. Doing so may cause unpredictable results and loss of proof information.
2. I got a repeated “`excp() failed`” message after I typed `PvsStart`. What does this mean?
This means PVS is not started successfully. A possibility is that Maple-PVS cannot find the local installation of PVS. Check if all paths are set correctly. Please refer to section 4.3 for details.
3. Maple is frozen after the Maple-PVS interface is initialized.
See answer of (2).
4. There is no Tcl/Tk window!
Please check the access permission of `pvs-ctl` found under `src/`. The current user needs “execute” privilege for it. Another possibility is that the source files are not compiled. Try to issue the `make` command in the `src/` directory.
5. PVS is installed in my system for sure, but it complains about something I don’t understand.
Be sure that you are running the PVS verification system described in this document. There is an unrelated Linux utility called `pvs`.

Bibliography

- [1] PVS Verification and Specification System. <http://pvs.csl.sri.com>
- [2] Maple Computer Algebra System. <http://www.maplesoft.com>
- [3] Monagan, Michael B., et al. *Maple Advanced Programming Guide*. Toronto: Maplesoft, a division of Waterloo Maple Inc., 2005.
- [4] Monagan, Michael B., et al. *Maple Introductory Programming Guide*. Toronto: Maplesoft, a division of Waterloo Maple Inc., 2005.
- [5] Andrew Adams, Martin Dunstan, Hanne Gottliesen, Tom Kelsey, Ursula Martin, Sam Owre. *Computer Algebra meets Automated Theorem Proving: Integrating Maple and PVS*. TPHOLS 2001. (?)