Library Management for PVS

Miguel Romero[1] Camilo Rocha[2]

[1]Escuela Colombiana de Ingeniería Julio Garavito
[2]Pontificia Universidad Javeriana, Cali

11CCC
September 29, 2016

Context

- The Prototype Verification System (PVS) is a verification system comprising a specification language integrated with support tools and a theorem prover
- The PVS system has been used in state-of-the-art formal methods projects as a productive environment for constructing and maintaining collections of theories, both in industry and in research.
- In `pvslm`, the description language is designed to represent a PVS library as a collection of packages, with a package consisting of a collection of PVS theories.

The Problem

Given the increasing size of mathematical developments in the form of libraries and the growing need for and adoption of mechanized proof environments, such as PVS, it is important to have tools for managing such libraries.

Main Contribution

- pvslm is an utility for assisting in the management of PVS libraries, comprising:
  - a description language for annotating PVS libraries
  - an implementation for managing PVS libraries annotated with this language
- The pvslm implementation is written in the Python programming language and it can manage any (annotated) PVS library that is publicly available from a pvslm server through the internet.

Roadmap

1. Description Language
2. Obtaining the `pvslm` Implementation
3. Available Commands
4. Conclusion

Roadmap

1. Description Language

2. Obtaining the `pvslm` Implementation

3. Available Commands

4. Conclusion

Description Language

The `pvslm` tool distinguishes three levels of aggregation for PVS sources. A PVS theory is the building block of a library managed by `pvslm`. A *package* is a collection of theories. A *library* is at the top level of aggregation, comprising a collection of packages.

Description Language: BNF-like notation

⟨metadata⟩	::=	⟨header⟩ ⟨body⟩
⟨header⟩	::=	‘/’ ⟨theorylist⟩
⟨theorylist⟩	::=	⟨theory⟩ | ⟨theory⟩ ‘,’ ⟨theorylist⟩
⟨body⟩	::=	(⟨packagedep⟩ | ⟨theorydep⟩)∗
⟨packagedep⟩	::=	⟨package⟩ ‘/’ ⟨theorylist⟩
⟨theorydep⟩	::=	⟨theory⟩ ‘:’ ⟨qualtheorylist⟩?
⟨qualtheorylist⟩	::=	⟨qualtheory⟩ | ⟨qualtheory⟩ ‘,’ ⟨qualtheorylist⟩
⟨qualtheory⟩	::=	(⟨package⟩ ‘@’)? ⟨theory⟩

Table: Syntax of the `top.dep` metadata file in NASALib.

Description Language: Example

```
/top,trig_doc,trig,trig_basic,trig_values,trig_ineq,trig_full,trig_extra,atan2
structures/for_iterate
reals/real_fun_preds,factorial,binomial,abs_lems,sign,sqrt_exists,root
analysis_ax/continuous_functions_props,derivatives,sqrt_derivative
finite_sets/finite_sets_minmax,finite_sets_inductions
ints/factorial
top:trig_doc,trig,trig_full,trig_basic,trig_values,trig_ineq,trig_extra
trig_doc:
trig:trig_basic,reals@sqrt,trig_values,trig_ineq
trig_basic:reals@sqrt
trig_values:trig_ineq
trig_ineq:trig_basic
```

Figure: Overview of metadata file for package trig in NASALib.

Roadmap

1 Description Language

2 Obtaining the `pvslm` Implementation

3 Available Commands

4 Conclusion

How to obtain pvslm

The pvslm tool can be installed automatically from the terminal in unix systems by issuing the following command:

```
curl
http://migueleci.github.io/pvslm/downloads/pvslm-conf.py
-o pvslm-install && chmod +x pvslm-install && python
./pvslm-install
```

Roadmap

1 Description Language

2 Obtaining the `pvslm` Implementation

3 Available Commands

4 Conclusion

Available Commands: Libraries

- Adding a library source (i.e., command `-a`) with a name, a short description, and a `git` URL.
- Deleting a library source (i.e., command `-d`) with the given name.
- Cloning a library source (i.e., command `-c`) with the given name.
- Updating a library source (i.e., command `-u`) with the given name.
- Removing the clone of a library source (i.e., command `-r`) with the given name.

Available Commands: Packages

- Installing and updating a given package (i.e., command `-i`) from a given library source, including *all* its dependencies.
- Updating a given package (i.e., command `-u`) from a given library source, including *all* its dependencies.
- Deleting a given package (i.e., command `-d`) from a given library source (local copy), including all packages that depend on it.
- Listing the contents (i.e., command `-l`) from a given library source.

Roadmap

1. Description Language

2. Obtaining the `pvslm` Implementation

3. Available Commands

4. **Conclusion**

Conclusion

- This tool features support for different library sources, libraries with several theories, and dependencies among the theories (within the same library source).
- The tool is freely available for download and it is distributed under GNU's GPLv3 license.
- It uses a small footprint language for annotating libraries

Thank you.