ProofLite: Batch Proof Checking and Literate Proving in PVS

César A. Muñoz

NASA Langley Research Center
Cesar.A.Munoz@nasa.gov

The PVS Theorem Prover

- PVS is a powerful interactive theorem prover.
- Standard PVS provides a powerful batch mode too (but mainly for expert users).
- Why do *normal users* need a batch mode ?

Scenario 1

After several weeks we have finished the development of `interval_arith`: 10 files, 322 lemmas.

- We want to double check the status of all lemmas.
- A new version of PVS is available. We want to recheck all the proofs.

Scenario 2

- We want to write proof scripts in the same file where we have the PVS specification.
- We are working on a third party application that generates PVS specifications *and PVS proofs*.

The ProofLite Package

- Package for non-interactive proof checking and proof scripting in PVS:
  - Utility for running the theorem prover in batch mode: `proveit`.
  - A proof scripting notation where proof scripts reside in `.pvs` files.
- Suitable for batch generation of PVS specifications and inlined proof scripts.
- Pre-installed in PVS 5.0

The `proveit` Utility

```
$ proveit interval_arith/top.pvs
```

The `proveit` Utility

```
$ proveit --importchain interval_arith/top.pvs
```

The `proveit` Utility

```
$ proveit --importchain --clean interval_arith/top.pvs
```

The `proveit` Utility

```
$ proveit -a interval_arith/top.pvs
```

The proveit Utility

```
$ proveit interval_arith
Processing interval_arith/top.pvs.
Writing output to file top.summary.
Grand Totals: 322 proofs, 322 attempted, 322 succeeded

$ more top.summary
 Proof summary for theory interval
    IMP_sigma_TCC1.....................proved - complete
    sharp_Proper.......................proved - complete
    ...
    Theory totals: 156 formulas, 156 attempted, 156 succeeded
 ...
```

ProofLite Scripts

- ProofLite scripts are written in PVS files using the special comment form:

```
l1: LEMMA a*a >= 0
%|- l1 : PROOF (grind) QED
```

- ProofLite scripts can extend to multiple lines:

```
l2: LEMMA (nza/2)*(2/nza) = 1
%|- l2 : PROOF
%|-   (then (skosimp)
%|-         (grind))
%|- QED
```

Sharing ProofLite Scripts

Several lemmas can share the same ProofLite script:

```
13: LEMMA a*a >= 0

14: LEMMA (nza/2)*(2/nza) = 1

%|- 13 : PROOF
%|- 14 : PROOF
%|-   (grind)
%|- QED
```

ProofLite Scripts for Name-Matching Formulas

- Name-matching formulas can share the same ProofLite script.
- The symbol * stands for an arbitrary sequence of one or more characters, e.g.,

```
%|- *TCC* : PROOF
%|-   (grind)
%|- QED
```

Macro Scripts

- Name-matching lemmas can be used to create macro scripts.
- The symbol $0 refers to the name of the lemma and the symbol $n refers to *n*-th matching string from left to right, e.g.,

```
l_5_6 : LEMMA EXISTS (a) :  5 < a AND a < 6

l_6_7 : LEMMA EXISTS (a) :  6 < a AND a < 7

%|- l_*_* : PROOF
%|-   (then (skip-msg "Proving Lemma: $0")
%|-         (inst 1 "$1 + ($2 - $1)/2")
%|-         (grind))
%|- QED
```

Parametric Scripts

- Parametric scripts have the form:

```
%|- <script_name>[e1;...;en]: PROOF
%|-   <steps>
%|- QED
```

- The symbol #*n* is substituted by e_n, e.g.,

```
l_8 : LEMMA EXISTS (a,b) :  a+b = 8
l_9 : LEMMA EXISTS (a,b) :  a+b = 9
%|- l_8[2;6] : PROOF
%|- l_9[4;5] : PROOF
%|-   (then (skip-msg "Proving Lemma: $0")
%|-         (inst 1 "#1" "#2")
%|-         (grind))
%|- QED
```

Installing ProofLite Scripts

Interactively

- ProofLite scripts in the current theory.
  - Without overriding old proofs:
    `M-x install-prooflite-scripts-theory` (`C-c it`).
  - Overriding old proofs:
    `M-x install-prooflite-scripts-theory!` (`C-c !t`).

- ProofLite scripts at the cursor position.
  - Without overriding old proofs:
    `M-x install-prooflite-script` (`C-c ip`).
  - Overriding old proofs:
    `M-x install-prooflite-script!` (`C-c !p`).

Installing ProofLite Scripts

In batch mode

`proveit` automatically installs ProofLite scripts on untried formulas (and on tried formulas if the option `--force` is used).

Creating ProofLite Scripts from Proofs

ProofLite scripts can be created from proofs in two ways:

- Place the cursor on the formula for which you want to create a ProofLite script and issue the Emacs command: `M-x insert-prooflite-script` (`C-c 2p`). The ProofLite script is automatically inserted after the formula.

- Issue the command: `M-x display-prooflite-script` (`C-c dp`) and enter the name of a formula. The ProofLite script of that formula is displayed in the buffer "ProofLite".

Conclusion

- The basic capabilities provided by ProofLite are already available in proof assistants such as Coq, HOL, etc.
- The ProofLite scripting notation also supports several forms of proof sharing and proof reuse.
- Proof scripts provide a simple mechanism to write user defined strategies.
- *Batch Proving and Proof Scripting in PVS*, César Muñoz, NASA Contract Report, `http://hdl.handle.net/2060/20070012333`.