Interactive Theorem Proving and Applications at NASA

J V Siratt

Formal Methods Research Team
Safety Critical Avionics Systems Branch
NASA Langley Research Center

February 11, 2025

NASA Langley has been at the forefront of government use of formal methods for more than 35 years.

In this time the Formal Methods Team has used the Prototype Verification System (PVS) extensively, applying it to various problems in aerospace.

We will give examples of the PVS specification language and discuss it's capability to perform interactive proofs.

We will see examples of how PVS has been used by the NASA Langley Formal Methods team in the past, and briefly discuss current work.

PVS Specification and Verification System

"PVS consists of a **specification language**, a large number of **predefined theories**, a **type checker**, an **interactive theorem prover** that supports the use of several **decision procedures** and a **symbolic model checker**, various utilities including a **code generator** and a random tester, documentation, formalized libraries, and examples that illustrate different methods of using the system in several application areas."

– from `pvs.csl.sri.com`

Specification language: *classical typed higher-order logic*
Proof system: *interactive sequent calculus*

The Martian

If you've seen this movie, you've seen part of our PVS library.

From a computer screen on the *Hermes*:

```
(""(induct "n")
 (("1" (expand "expt") (("1" (propax) nil nil)) nil)
  ("2" (skosimp*)
   (("2" (expand "expt" 1)
     (("2" (inst - "px!1")
       (("2"
         (lemma "both_sides_times_pos_le1"
          ("pz" "px!1" "x" "1" "y" "expt(1+px!1,j!1)"))
         (("2" (rewrite "expt_gt1_bound1" -1)
           (("2" (assert) nil nil)) nil))
         nil))
       nil))
     nil))
   nil))
 nil)
```

This is the PVS internal representation for a proof of

$$\forall n \in \mathbb{N}, x \in \mathbb{R}^{+},\ 1 + nx \leq (1+x)^{n}$$

The Martian

This is NOT code to shutdown or startup the Mars Habitat and the MAV.

```
mpoly         : VAR MultiPolynomial
mdeg          : VAR DegreeMono
mcoeff        : VAR Coeff
nvars,terms   : VAR posnat
rel           : VAR RealOrder
Avars,Bvars   : VAR Vars
boundedpts,
intendpts     : VAR IntervalEndpoints

MPoly : TYPE = [#
  mpoly  : MultiPolynomial,
  mdeg   : DegreeMono,
  terms  : posnat,
  mcoeff : Coeff
#]

mk_mpoly(mpoly,mdeg,terms,mcoeff) : MACRO MPoly = (#
  mpoly  := mpoly,
  mdeg   := mdeg,
  terms  := terms,
  mcoeff := mcoeff
#)
```

It's actually specifying a data structure for representing multivariate polynomials.

A PVS Theory

```
sum: THEORY
BEGIN

  sum(n:nat): RECURSIVE nat =
    IF n=0 THEN 0 ELSE n+sum(n-1) ENDIF
  MEASURE n

  closed_form: THEOREM
    FORALL (n:nat): sum(n) = (n*(n+1))/2

END sum
```

Typechecking

The first correctness test in PVS involves typechecking.

The PVS typechecker will generate a Type Correctness Condition (TCC) if something is not obviously correct or incorrect.

The sum function might generate TCCs to the effect:

$$\forall (n : nat) : n \neq 0 \rightarrow n - 1 \geq 0$$

$$\forall (n : nat) : n \neq 0 \rightarrow n - 1 \leq n$$

Often TCCS are discharged by PVS itself.
Other times they are easily proven by the user.

Conjecture
If a TCC is difficult to prove it's because I've messed up.

Interactive Prover

Invoking the prover on our theorem, we enter a separate environment.

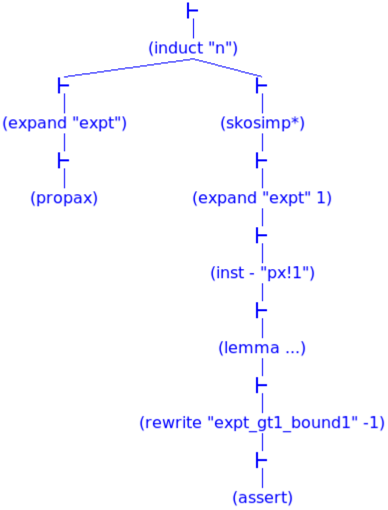
```
closed_form:

  |-------
{1} FORALL (n:nat): sum(n) = (n*(n+1))/2

Rule?
```

Using PVS proof rules we build a proof tree which is complete when every leaf is a trivial sequent.

Proof of $\forall n \in \mathbb{N}, x \in \mathbb{R}^{+},\ 1 + nx \leq (1 + x)^{n}$ displayed as tree.



Types Depending on Something

PVS supports predicate subtyping and theory parameterization.

```
example[T:TYPE,k:posint]: THEORY
BEGIN

  mytype: TYPE = {ls:list[T] | length(ls)<=k}

  ex1: TYPE = [# first : nat, second : {m:nat | m>first} #]
  ex2: TYPE = [# second : nat, first : {n:nat | n<second} #]
  ex3: TYPE = {x:[# first : nat, second : nat #] |
                   x`first<x`second}

END example
```

Replacing theorems with subtypes

The subtype of *list_max* claims it does its job.

```
nonempty_list_of_reals: TYPE+ = {ls:list[real] | cons?(ls)}

list_max(ls:nonempty_list_of_reals): RECURSIVE
  {x:real | member(x,ls) AND
            (FORALL (r:real): member(r,ls) IMPLIES r<=x}) =
  COND length(ls)=1 -> car(ls),
       length(ls)=2 -> max(car(ls),cadr(ls)),
       ELSE -> list_max(cons(max(car(ls),cadr(ls)),cddr(ls))
  ENDCOND
MEASURE length(ls)
```

PVS Support

PVS was created and developed by SRI.

Langley FM has used, supported, and helped fund PVS for many years. Some of our contributions:

- NASA PVS Library – Collection of formal developments in PVS maintained by NASA Langley FM Team.
  - $\sim$62 top-level libraries containing $\sim$38,000 proven formulas
  - many (but not all) of these libraries have been written by the Langley FM team
- PVSio – Animation tool of PVS functional specifications.
- VSCode-PVS – Visual Studio Code plugin supporting an ever-increasing subset of PVS functionality.

The future of PVS

Expected near the end of May:

- PVS 8.0 - Transition from Allegro to SBCL
- NASAlib 8.0
- VScode-PVS 8.0
- PVS2C

NASA Langley
Formal Methods Research Program

Who are we and what do we do?

Safety-Critical Avionics Systems Branch
Research Directorate

- Major goals:
  - Advance the state-of-the-art in formal methods,
  - Orchestrate the transfer of this technology to industry.

- Basic strategy: apply formal methods to challenging areas of digital flight-control systems design and initiate demonstration projects.

An Incomplete Specification

The Compact Position Reporting (CPR) algorithm enables aircraft to share their position and velocity with other aircraft in their vicinity.

What Langley FM did:

- Formal proof that published requirements for decoding were insufficient (even assuming exact real arithmetic).
- New requirements formally proven to guarantee correct decoding under exact real arithmetic.
- Equivalent but computationally simpler forms of expressions used in CPR functions to reduce imprecision.
- Fixed point and floating point implementations in C which became the reference in international standard ED-102B/DO-260C.

How we use PVS (some examples)

- **Algorithm verification**

  DAIDALUS - Detect and Avoid Alertic Logic for Unmanned Systems
- **Formal certificates**

  PRECiSA - Program Round-off Error Certifier via Static Analysis
- **Operational embeddings**

  Plaidypvs - differential dynamic logic for hybrid program verification
- **Differential testing**

  PVSio - animation tool of PVS functional specifications
- **Proof strategies (tactics)**

  Manip - automate algebraic manipulations, and much more
- **Computational reflection**

  Tarski - prove results for systems of inequalities using verified ground evaluation.
- **Formal semantics**

  PLEXIL5 - formal semantics and metaproperties in PVS

We currently have people working on ...

- Preparing NASAlib, VSCode-PVS, etc., for the release of PVS 8.0.
- Improving Plaidypvs with a library of hybrid programs realizing vehicle dynamics, modeling communications between systems, and more.
- Applying a verified implementation of Bellman-Ford in determining paths for best navigation quality.
- Exploring in-time path verification
- Applications of machine learning to formal methods (e.g. using LLMs for lemma suggestion).
- Developing assurance techniques for machine learning.
- Applying model checking (Spin) to service-oriented software architectures (namely ICAROUS).
- Using PRECiSA to verify numerically intensive algorithms for data-fusion and association.

Aerial AId

Drone First Response (DFR)

Exploratory phase:

- engage with first responders using drone tech
- identified automation as an area stakeholders wanted developed
- test flights for preliminary data

Execution phase (begun Oct 2024):

- building and testing visual perception models
- future flight tests
- develop suitable assurance techniques

Goal: enable long-term vision of safe autonomous DFR

Aerial AId – Assurance Team

Quality Data

Little public aerial data from correct height

Existing data usually has strings attached

Must not introduce bias

Well-behaved Model

Volatility (surprise adequacy, Lipschitz, etc.)

Runtime assurance frameworks

Formally verified components (or analogs)

Operator Considerations

Human-in-the-loop

Avoid information overload

Transparency – don't hide info!

NASA Formal Methods Symposium

From 1990 to 2008 the Langley Formal Methods Team held a series of formal methods workshops.

This series evolved into the NASA Formal Methods Symposium (NFM) organized by the NASA Formal Methods Research Group, which consists of researchers at six NASA centers.

The 17th NASA Formal Methods Symposium is being organized by the Langley Formal Methods Team and will be held June 11-13 in Hampton Roads, VA.