Lemma Suggesting in PVS using Machine Learning

Kyle Nguyen[1], Mariano Moscato[2,3], J. Tanner Slagel[2]

[1] Biomathematics Graduate Program
North Carolina State University

[2] Safety Critical Avionics Systems Branch
NASA Langley Research Center

[3] Formal Methods Team
National Institute of Aerospace

August 5, 2020

Overview

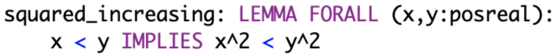
- Motivation
- Proof procedure in PVS
- Implementation
- Machine learning model
- Prenex normal form conversion
- Empirical evaluation
- Integration of suggester to vscode-pvs IDE

Motivation

- Safety-critical systems require the highest possible degree of verification and validation.
- Theorem proving offers such kind of warranties.
  - NASA LaRC use Prototype Verification System (PVS) for verification of safety-critical systems.
- Downside: it is a very time-consuming activity.
- Goal: Speed up formal verification time in PVS using machine learning (ML) by adding a lemma suggester to PVS.

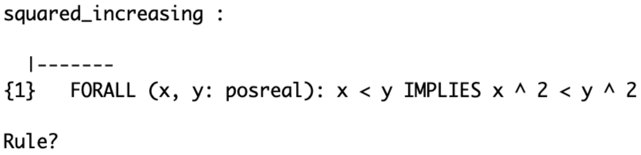
Proof procedure in PVS

- Defining a lemma:



```
squared_increasing: LEMMA FORALL (x,y:posreal):
    x < y IMPLIES x^2 < y^2
```

- Lemma presentation in PVS:



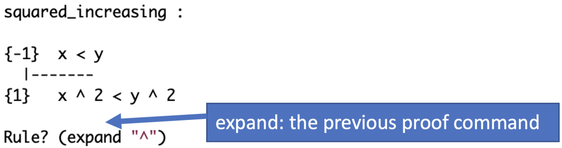
```
squared_increasing :

  |-------
{1}   FORALL (x, y: posreal): x < y IMPLIES x ^ 2 < y ^ 2

Rule?
```

Proof procedure in PVS

- Step $n - 1$:



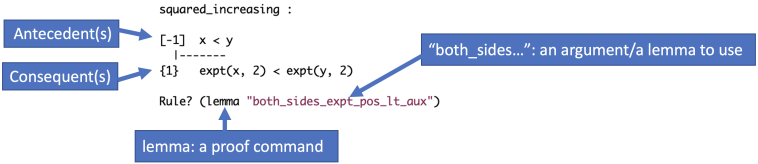
```
squared_increasing :

{-1}  x < y
  |-------
{1}   x ^ 2 < y ^ 2

Rule? (expand "^")
```

expand: the previous proof command

- Step n:



```
squared_increasing :

[-1]  x < y
  |-------
{1}   expt(x, 2) < expt(y, 2)

Rule? (lemma "both_sides_expt_pos_lt_aux")
```

Antecedent(s)

Consequent(s)

“both_sides…”: an argument/a lemma to use

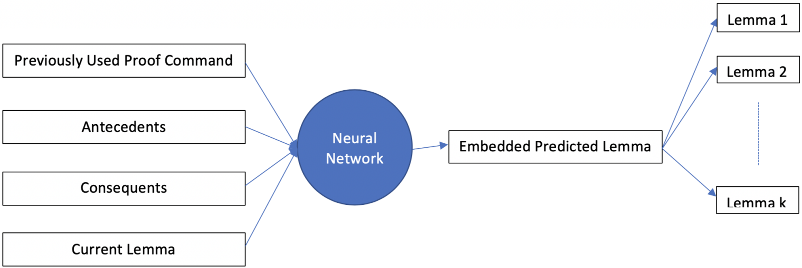
lemma: a proof command

Proof procedure in PVS

- Usage of lemmas in PVS:
  - Pros: using the correct lemma could speed up the proof process.
  - Cons: user needs to locate it among the existing library (NASALib has $\approx$ 30,000 available lemmas).
- Need a new feature that could provide users the right lemma during proof process.

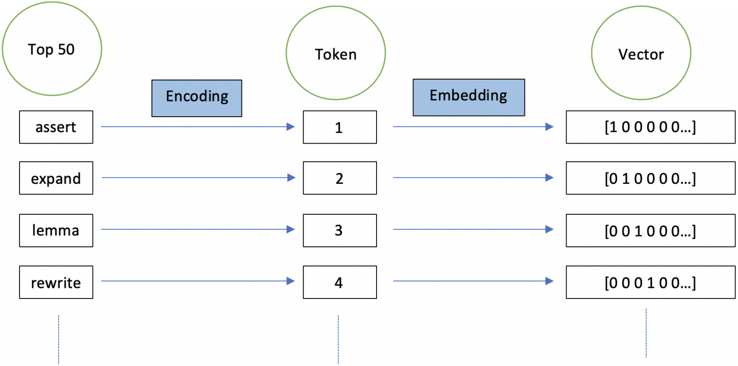
Implementation

- Case study: real library from the NASALib.
- Number of lemmas: 1048.
- Data size: 2167.
- Inputs: Previously used proof command, antecedents, consequents, and current lemma.
- Neural network model: A combination of convolution and long-short term memory.
- Model output: An embedded vector of predicted-to-use lemma.
- Output: A list of “useful” lemmas.



Embedding Previously Used Tactic Command

- Take the top 50 mostly used proof commands.
- Encoding: assigning a number to each proof command.
- Embedding: using one-hot encoding.



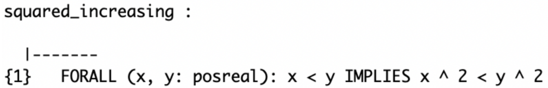
Encoding

Embedding

Top 50	Token	Vector
assert	1	[1 0 0 0 0 0...]
expand	2	[0 1 0 0 0 0...]
lemma	3	[0 0 1 0 0 0...]
rewrite	4	[0 0 0 1 0 0...]

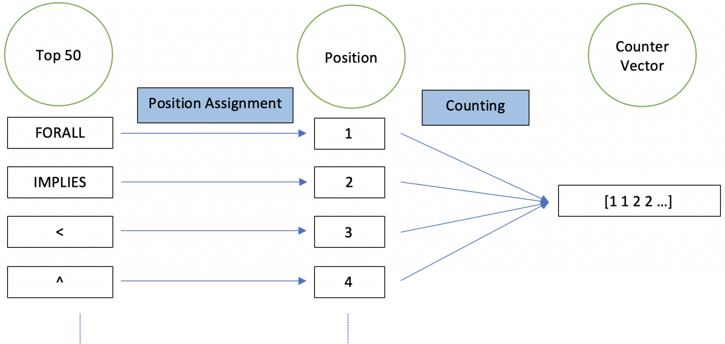
Counter for Antecedents, Consequents, Lemmas

- Take the top 50 most important keywords/symbols/data types.
- Assign a position.
- Count the number of their appearance.



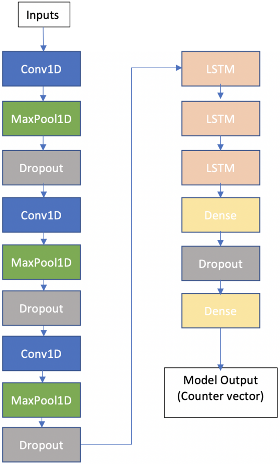
```
squared_increasing :

  |-------
{1}   FORALL (x, y: posreal): x < y IMPLIES x ^ 2 < y ^ 2
```



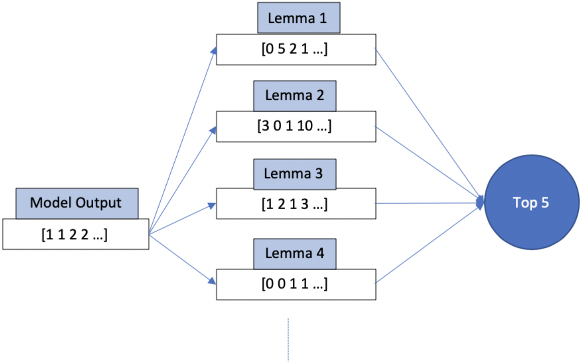
Neural Network Model

- Loss function: Mean square error.
- Optimizer: Adam [1].
- Training/Validation/Testing ratio: 70/20/10.



Suggesting Top 5 Lemmas

- Compute the difference between model output and each lemma in the real library.
- Choose the top 5 (out of 1048) lemmas with the smallest difference.



Result of the First Attempt

- Test size: 217.
- Predict correctly when the actual lemma used is in the top 5 lemmas predicted.
- Model accuracy $\approx 6.5\%$ (14 out of 217).
- Potential explanations for low accuracy:
  - Model overfitting.
  - Have not explored hyperparameters.
  - Encoding was not rich enough to capture important information.
  - Proof data from PVS are not in a normal form.

Prenex Normal Form Conversion

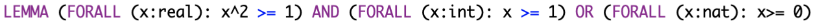
- Why prenex normal form?
  - To canonize PVS formulas for accurate comparison.
  - Example:

$$A \iff B \equiv (A \implies B) \land (B \implies A)$$

- Implemented in Common Lisp.
- Prenex normal form examples:

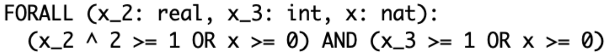
$$A \implies B \to \neg A \lor B$$
$$\neg \forall x A(x) \to \exists x \neg A(x)$$
$$\forall x A(x) \land \forall x B(x) \to \forall x \forall y A(x) \land B(y)$$

Result of Prenex Normal Form Conversion



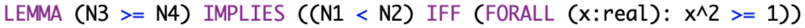
```
LEMMA (FORALL (x:real): x^2 >= 1) AND (FORALL (x:int): x >= 1) OR (FORALL (x:nat): x>= 0)
```

Output:



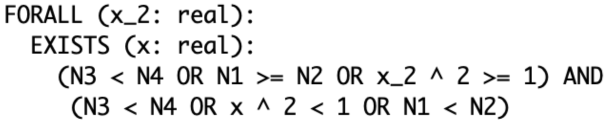
```
FORALL (x_2: real, x_3: int, x: nat):
  (x_2 ^ 2 >= 1 OR x >= 0) AND (x_3 >= 1 OR x >= 0)
```

Result of Prenex Normal Form Conversion



```
LEMMA (N3 >= N4) IMPLIES ((N1 < N2) IFF (FORALL (x:real): x^2 >= 1))
```

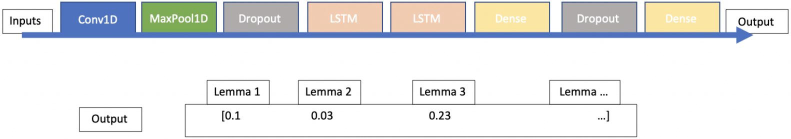
Output:



```
FORALL (x_2: real):
  EXISTS (x: real):
    (N3 < N4 OR N1 >= N2 OR x_2 ^ 2 >= 1) AND
     (N3 < N4 OR x ^ 2 < 1 OR N1 < N2)
```

Empirical Evaluation

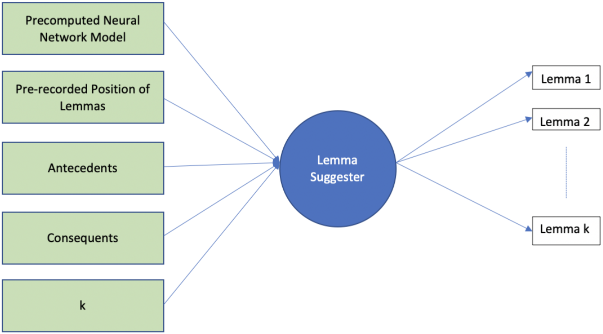
- What did we change?
  - Removed: previously used proof command, current lemma.
  - Counter vector size: 50 → 25.
  - Simplified the model.
  - Converted to a traditional classification problem.
  - Chose top 5 (out of 434) lemmas.



- Model accuracy ≈ 42% (91 out of 217).

Integration of Suggester to Vscode-pvs IDE

- Load pre-computed neural network model and pre-recorded position of the 434 lemmas.
- Receive queries as JSON format from Vscode.
- Return JSON output file containing the top k relevant lemmas.


Precomputed Neural Network Model
Pre-recorded Position of Lemmas
Antecedents
Consequents
k
Lemma Suggester
Lemma 1
Lemma 2
Lemma k

Future Direction

- Try different neural network architectures: WaveNet, Graph Neural Networks.
- Explore different tokenizers.
- Increase the size of dataset.
- Predict proof commands.
- Fully automated formalization.

Summary

- To NASA:
  - Introduced an initial framework for lemma suggesting feature in PVS using machine learning.
  - Developed a prenex normal form conversion feature in PVS.
- To me:
  - Learned how to use PVS.
  - Improved Common Lisp skill while trying to build prenex normal form conversion feature.
  - Applied machine learning to formal verification.
  - Applied object oriented visitor design pattern to a real-life implementation of a higher order logic language.
  - Worked with non-trivial logic concepts (higher order languages and sequent calculus) and their implementation in an object oriented setting.

Acknowledgment

- Mariano Moscato and J. Tanner Slagel.
- César Muñoz.
- NASA Langley Research Center.

Thank you! Questions?

References

D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.