Intelligent Lemma Selection for Formal Methods Proofs

Connor T. Baumler[1], Mariano M. Moscato[2], J. Tanner Slagel[2]

[1] Department of Computer and Data Science
Case Western Reserve University

[2] Safety Critical Avionics Systems Branch
NASA Langley

Aug 2020 @ NASA Langley
Intern exit presentation

Outline

Outline

Motivation

- 30,000 lemmas in NASA's PVS (Prototype Verification System) library
- Using one lemma in the proof of another is a common occurrence
  - `lemma`, `rewrite`, and `use` command used 70,000+ times in NASAlib proofs
  - Manual or string-based search often used to find lemmas

Motivation

- 30,000 lemmas in NASA's PVS (Prototype Verification System) library
- Using one lemma in the proof of another is a common occurrence
  - `lemma`, `rewrite`, and `use` command used 70,000+ times in NASAlib proofs
  - Manual or string-based search often used to find lemmas

How can we make finding a relevant lemma an easier task?

Outline

Fact Selection Process

- To decide which NASAlib lemmas are relevant, each fact selector compares the current proof state to the initial proof state from the library lemmas
- The proof information from each state is transformed into a set of features
- Lemmas with features that are determined to be more similar to the features in the current proof state are selected

Feature Extraction

- Basic features and patterns are pulled from proof states
- These simple features come from the variables (x), constants (a), and functions (g, h) in each formula
- Variable names are replaced by their types
- For function calls with arguments, patterns of arguments up to depth 2 are considered. Logical connectives are ignored
- e.g. The features for the expression $g(h(x : nat, a))$ are:

$$\begin{array}{ccc}
nat & h(\_, a) & g(h(nat, \_)) \\
a & h(nat, a) & g(h(\_, a)) \\
h(\_, \_) & g(\_) & g(h(nat, a)) \\
h(nat, \_) & g(h(\_, \_)) &
\end{array}$$

- The features of each proof state are collected with no separation based on which formula the feature appears in (cf. Kyle's counting vectors)

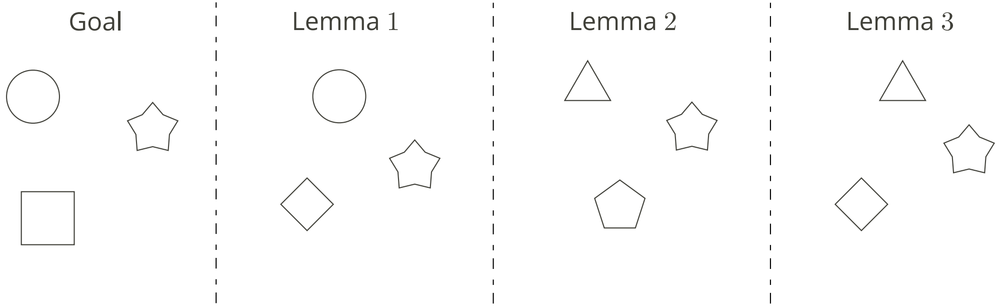
MePo

MePo is a simple, relevancy filtering based fact selector (Meng and Paulson 2009)

Relevance is scored in relation to a passmark or a minimum relevance grade

MePo

To start, the so-called “relevant set” of features contains only goal features

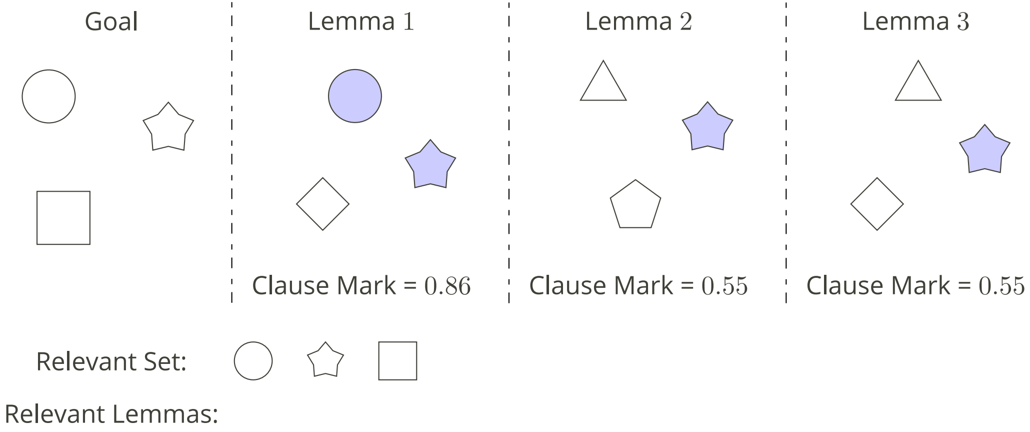


Relevant Set: ○ ☆ □

Relevant Lemmas:

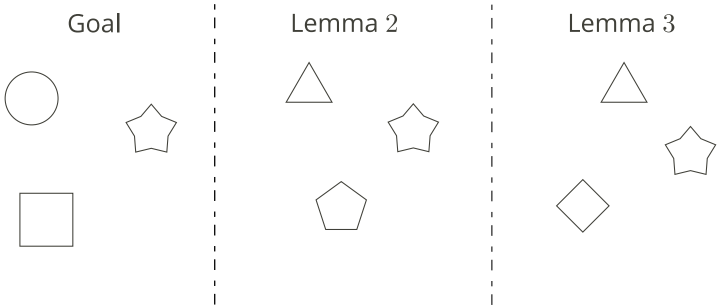
MePo

Features of each lemma are partitioned by the relevant set. The clause mark is calculated based on the proportion of relevant features and how frequently the relevant features occur in the whole library



MePo

Lemmas with a clause mark above the passmark ($.6$) are marked as relevant, and their features are added to the relevant set. The passmark is increased ($.8$, in this case)

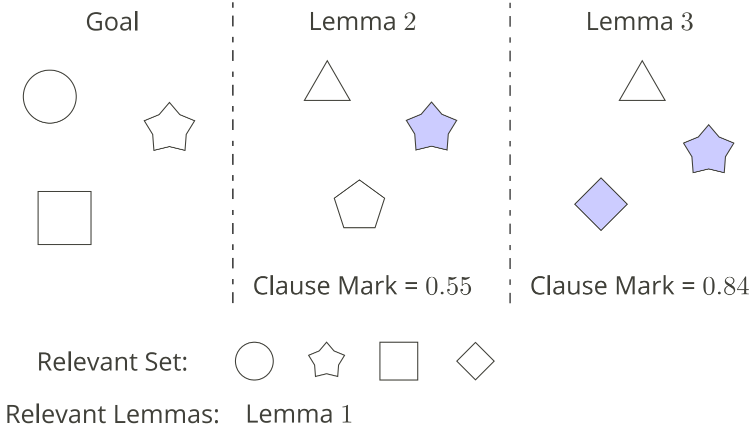


Relevant Set: ○ ☆ □ ◇

Relevant Lemmas: Lemma 1

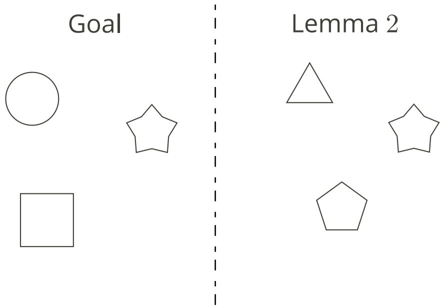
MePo

The scoring repeats



MePo

The passmark and the relevant lemmas and features are updated. (The new passmark is $.9$)

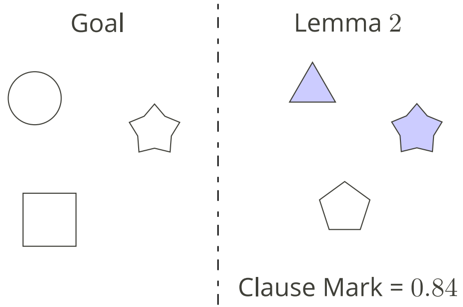


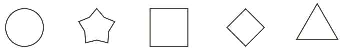
Relevant Set: ○ ☆ □ ◇ △

Relevant Lemmas: Lemma 1, Lemma 3

MePo

After scoring, we see that the final lemma doesn't make the current passmark ($.9$). Since no new lemmas are selected, lemmas 1 and 3 are returned as relevant



Relevant Set: 

Relevant Lemmas: Lemma 1, Lemma 3

MePo

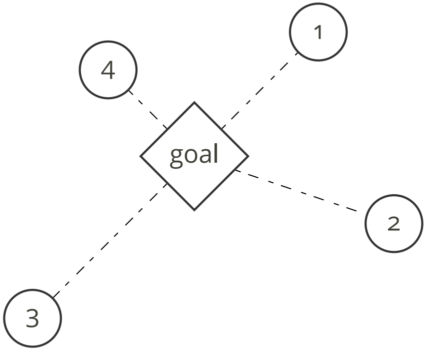
Drawback: Meng and Paulson 2009 found that this approach does well when the goal contains rare features, but falls apart when all the goal features are common

MaSh

- MaSh is a learning-based fact selector that builds on MePo (Blanchette et al. 2016)
- This approach utilizes the proof of each fact to find a dependency structure
- The facts are partially ordered by visibility based on their dependencies
- We use MaSh’s k-nearest neighbors method

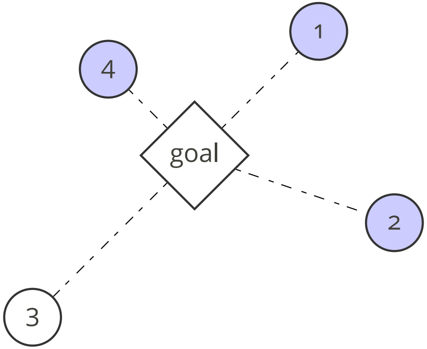
MaSh

Consider the lemmas visible from the goal



MaSh

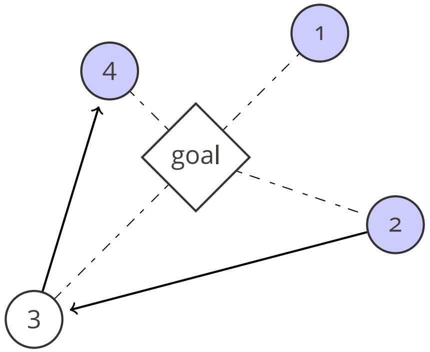
Up to k such lemmas selected based on the features they have in common and their rarity in the entire library



MaSh

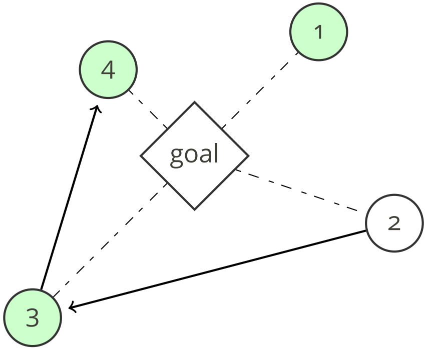
Instead of updating the set of relevant features, this approach considers each goal-visible lemma's other goal-visible dependencies. They are scored based on:

- How close each dependency of the lemma is to the goal
- How many dependencies the goal has
- If it was a chosen neighbor, the original nearness of the lemma to the goal



MaSh

Given this second scoring, up to k lemmas are selected and ordered



MaSh

- In practice, the number of initially selected neighbors starts at 0 and is gradually increased to make sure enough lemmas have non-zero relevance
- Blanchette et al. 2016 found that this approach generally outperforms MePo, with the difference being more pronounced when the number of candidate lemmas is smaller

MeSh

- Since MePo can do better than MaSh on certain problems, the hybrid approach MeSh combines the two (Blanchette et al. 2016)
- The rankings of each fact selector is combined with equal weight
- Blanchette et al. 2016 found that while this does improve overall performance when being used in certain provers, the addition of MePo often hurt results more than it helped

Outline

Implementation in PVS

- I implemented the three fact selectors themselves in python. The feature extraction (done by Mariano) was implemented in lisp and pulls directly from PVS’s internal representation
- MePo, MaSh, and MeSh are being integrated into the PVS VSCode extension
    - This will allow users to receive suggestions of potentially relevant NASALib lemmas as they work to prove novel lemmas

“Goals” in PVS

- MePo, MaSh, and MeSh were originally created for Isabelle (Meng and Paulson 2009, Blanchette et al. 2016)
- In Isabelle, they were able to easily compare the features of the goal of the current proof (or sub-proof) to the final state in facts’ proofs
- In PVS, a lemma in proven when a terminal state is reached
    - This happens when certain formulae in the state are true, false, or the same
- Since the objective is then to transform a formula in the current state in a useful way, we use the current state as the goal
- This also means that similarity to the final state of a NASAlib lemma’s proof (the leaves of the proof tree) is likely a less informative metric than similarity to the initial state
- So, we ultimately compare the current proof state to the initial state of each NASAlib lemma

Outline

Evaluation on NASAlib

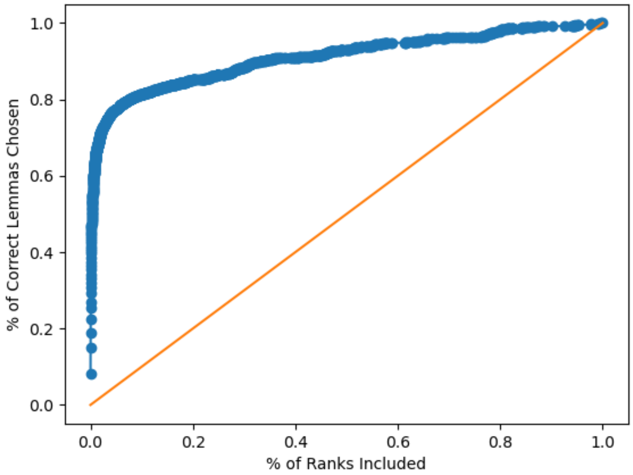
- To test the three methods, we utilized the existing PVS proofs in NASA's library
- For each lemma's proof, we found the steps where other NASAlib lemmas were applied
- Treating that point in the proof as the current state, we saw what lemmas each method thought was relevant
- This method likely underestimates the efficacy of these approaches since:
    1. The lemma that was actually applied at the given state might not be the only lemma that leads to a successful proof
    2. Some of the candidate lemmas being considered are dependent on the lemma being proven and thus wouldn't be in the set of proven lemmas before the current one has been proven

Evaluation on NASAlib

Method	Accuracy	Avg relative lemma rank in set (when included)	Avg Passing set size	Avg eval time
MePo ($p = .9$)	8.4%	56.7%	3.45	0.061
MePo ($p = .6$)	38%	14.8%	116.5	0.179
Mepo ($p = .3$)	79%	5.40%	3101.1	0.478
MaSh (max 1000)	89.0%	11.27%	1000	0.067
Mepo ($p = 0$)	n/a	7.24%	n/a	1.49
MaSh (no cutoff)	n/a	7.33%	n/a	0.084
MeSh ($p = 0$, no cutoff)	n/a	6.19%	n/a	1.23

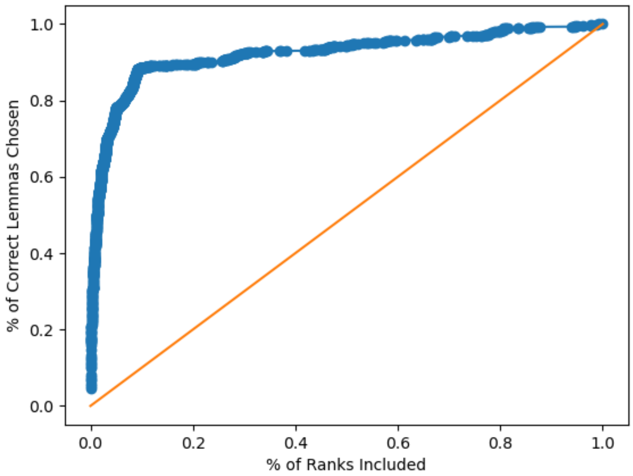
Evaluation on NASAlib

MePo with $p = 0$ (Area = $.919$)



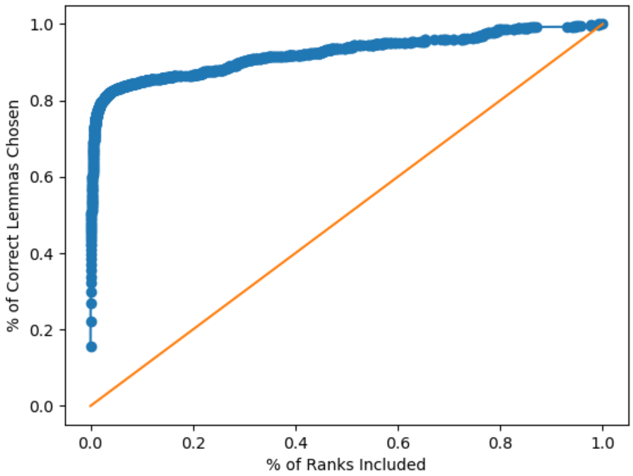
Evaluation on NASAlib

MaSh with no maximum suggestion cutoff (Area = $.923$)



Evaluation on NASAlib

MeSh with $p = 0$ and no maximum suggestion cutoff (Area = $.924$)



Outline

Conclusion

- Three Lemma selectors implemented for use in PVS and for use as baselines for further experimentation
  1. MePo: Simple, passmark-based relevancy scoring
  2. MaSh: K-nearest neighbors
  3. MeSh: Ensemble of MePo and MaSh
- Comparison of MePo vs MaSh vs MeSh performance
  - We find overall similar performance between the three methods, with MeSh having the best AUC.
  - Since NASAlib has so many lemmas, this is unsurprising. Blanchette et al. 2016 reported MaSh and MeSh to have the greatest improvement over MePo in problems considering between 32 and 512 facts
  - This may suggest that a preliminary pruning of candidate lemmas outside the domain of the current proof could increase MaSh and MeSh's performance

Future Work

Function and Constant names

- Currently, variable names are replaced by their types since their names are quite arbitrary
- Two constants of the same type may be interchangeable
- Two functions of the same type signature may also be interchangeable
- A system that can determine the similarity functions/constants (independent of their names), will be better able to determine the similarity of lemmas

Command-level Suggestions

- Instead of suggesting existing lemmas, we may find success looking for relevant previously-used proof commands
- Using a similar architecture, we could look at the features in each library lemma before a command was used and comparing it to the current state

MeSh ensemble

- Currently, MeSh combines MePo and MaSh with a simple, equal weight vote
- Using an ensemble learning technique like bagging or boosting as a base, we can try to find a more appropriate weighting

References

Blanchette, Jasmin Christian et al. (2016). “A learning-based fact selector for Isabelle/HOL”. In: *Journal of Automated Reasoning* 57.3, pp. 219–244.

Meng, Jia and Lawrence C Paulson (2009). “Lightweight relevance filtering for machine-generated resolution problems”. In: *Journal of Applied Logic* 7.1, pp. 41–57.