Formal Premise Selection With Language Models

Szymon Tworkowski*, Maciej Mikula*, Tomasz Odrzygódz*, Konrad Czechowski*,
Szymon Antoniak*, Albert Q. Jiang3, Christian Szegedy2, Łukasz Kuciński4, Piotr Miłoś4,
Yuhuai Wu2

1 University of Warsaw, 2 Google Research
3 University of Cambridge, 4 Polish Academy of Sciences, 5 IDEAS NCBR

Abstract
Premise selection, the problem of selecting a useful premise to prove a new theorem, is an essential part of theorem proving. Existing language models cannot access knowledge beyond a small context window, and therefore are unsatisfactory at retrieving useful premises (i.e., premise selection) from large databases for theorem proving. In this work, we provide a solution to this problem, by combining a premise selection model with a language model. We first select a handful (e.g., 8) of premises from a large theorem database consisting of 100K premises, and present them in the context along with proof states. The language model then utilizes these premises to construct a proof step. We show that this retrieval-augmented prover achieves significant improvements in proof rates compared to the language model alone.

1 Introduction

Language models have been recently applied to theorem proving [17, 25, 12, 14, 24] and program synthesis [7, 2, 20], achieving impressive results. Premise selection is a fundamental aspect of formal mathematics [28, 1, 4]. Early works in this domain often relied on symbolic [19, 5] or hybrid [15] approaches. Classical ML algorithms [30, 29, 9] have also proven effective, frequently outperforming symbolic methods by significant margins. More recently, graph neural networks mimicking the symbolic structure of mathematical expressions have shown promising results [22, 33, 10, 18].

Effective retrieval of premises from large databases is still an open challenge. In this work, we propose to approach it with a two-stage procedure, which, to the best of our knowledge, is the first method to do the selection process globally over the whole corpus. Firstly, a premise selection model (PSM) picks a handful (e.g. 8) of premises from a database. These are then presented, along with a proof state, to a premise selection guided language model (PGLM) responsible for generating a proof step. Importantly, our PSM can efficiently query large databases; in our case, we use over 100K lemmas from the entire Isabelle corpus. By providing a relatively small number of premises in context, we allow the PGLM to efficiently retrieve the correct ones, aiming to leverage its in-context learning capabilities [16].

2 Method

Premise selection model (PSM) is based on a batch contrastive learning approach similar to [1, 3, 26, 11]. It encodes proof state and premise text into embeddings. The cosine similarity of a given premise embedding and a proof state embedding estimates their mutual relevance. Premise embeddings can be precomputed and cached, allowing for the use of large databases.

*Equal Contribution
Premise-guided language model (PGLM) is a model for proof step generation. It takes as an input the current proof state \( s \) and premises (names and statements) selected by PSM. These are e.g. \( k = 8 \) premises from the whole database with the highest relevance to \( s \).

We first train the PSM, then freeze the weights and use it in the training process of PGLM. The PGLM is designed to perform premise-aware proof step generation. By design, given the (small) context of \( k \) premises, the model selects the relevant ones to be applied in the generated proof step. This setup is motivated by recent findings [16] showing that LMs can grasp dependencies in the text within the same input much better than ones occurring across different training examples. The latter is how the state-only (our baseline model, described below) approach works. We hope that in-context learning helps the model focus on premise selection instead of memorization of frequently-occurring premises (as we hypothesize the state-only models do).

The State-only model is a language model that, given a proof state (goal), predicts the proof step. This is the most common setup found in prior work [14, 12, 24], used here as a baseline.

3 Experiments
We conduct our interactive theorem proving experiments on a dataset collected in Isabelle [23] which is one of the largest corpora of formal proofs. To interact with the formal environment, we use PISA [14]. The proof rates are presented in the table below.

<table>
<thead>
<tr>
<th>Method</th>
<th>Proof rate, full</th>
<th>Proof rate, ( \geq 1 ) premise</th>
<th>Proof rate, 0 premises</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sledgehammer [5] (baseline)</td>
<td>22.4%</td>
<td>17.7%</td>
<td>27.5%</td>
</tr>
<tr>
<td>State-only (baseline)</td>
<td>39.8%</td>
<td>14.7%</td>
<td>67.1%</td>
</tr>
<tr>
<td>PGLM+PSM (ours)</td>
<td>43.1%</td>
<td>19.6%</td>
<td>68.6%</td>
</tr>
<tr>
<td>PGLM+PSM ( \cup ) State-only</td>
<td>47.2%</td>
<td>22.6%</td>
<td>73.9%</td>
</tr>
</tbody>
</table>

Table 1: Proof rate is evaluated using a best-first search solver, similar to the one mentioned in [14], on a test set of 1000 theorems. We split the test dataset into proofs originally using and not using premises; denoted \( \geq 1 \) premise and 0 premises, respectively. For the sledgehammer baseline we use 50s timeout per proof.

Our method, PGLM+PSM, performs significantly better on theorems that require at least one premise and fares well on the entire test set. This indicates that the proposed two-stage method is efficient in premise retrieval. Furthermore, a significant improvement is observed when PGLM+PSM and the state-only model are combined. This is especially visible on the full test set, indicating that both methods have complementary strengths.

4 Conclusion and future work
We present a simple method integrating premise selection with language models, which is guided by an external retriever model. We show proof rate improvements when compared to a state-only baseline and demonstrate that our model is capable of generating novel proofs that utilise premises.

We speculate that scaling up our approach will further increase its capabilities. In particular, we hypothesise that in-context premise selection performance will improve due to better generalisation to unseen premises. If true, it would indicate better reasoning potential of the underlying language model, and as such is an attractive research direction.
References


[18] Zhaoyu Li, Binghong Chen, and Xujie Si. Graph contrastive pre-training for effective theorem reasoning, 2021.


A Experimental setup

A.1 LM setup

For language modeling, we use a decoder-only transformer [31] with 30M non-embedding parameters. The setup (weight initialization, positional embeddings, and other architectural hyperparameters) is exactly the same as in GPT-J [32]. We use a pretrained BPE tokenizer from [27]. Similarly to GPT-f [25], the loss function is calculated only on the proof step tokens. As a context for proof step generation we use one sentence representing the proof state for state only model, and premises + proof state sentence for PGLM+PSM setup.

For the PGLM+PSM model, in our main result, we provide it with top $k = 4$ premises from the premise selection model.

All of the models are pretrained on The Pile [8] - GitHub + arXiv dataset for 500k steps with context length of 2048 as in [6] and total batch size of $2^{17}$ tokens per update.

A.2 PSM setup

We modify the InfoNCE [21] loss by only using row-wise softmax (column-wise softmax is ablated). Batch size of 512 proof states is used. We also randomly sample 1536 additional negative premises within a batch (512 proof states and 2048 premises in each batch, for each proof state there is exactly one positive premise and 2047 negatives), and we find it helpful to the score (see Tab. 2). We use a non-pretrained, 6-layer decoder-only transformer (15M non-embedding parameters).

B Dataset and Environment

Isabelle [23] is an interactive theorem prover (ITP). It allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas, which are verified by a logical kernel. Its main application is the formalization of mathematical proofs and in particular formal verification, which includes proving the correctness of computer hardware or software and proving properties of computer languages and protocols. Each Isabelle library is composed of theories. A proof for a given theorem is a sequence of proof steps, with each step being a proof tactic or part of a declaration. Each subsequent proof step changes the state (referred to as proof state) of the current proof. A proof step can make use of premises, which are simply references to definitions, axioms, or previously proven theorems. This theorem proving setting constitutes a Partially Observable Decision Process and thus can be represented by a sequential decision process in a certain environment. An example of such an environment is the PISA environment [14], which we used for all the experiments. We trained models using a dataset mined from the Archive of Formal Proofs (AFP)[13] and all the standard libraries available in Isabelle. The dataset consists of 220K lemmas, with a total of 2.4M (proof state, proof step) pairs. For the premise selection task, we chose the proof steps that utilised at least one premise, which resulted in 400K training examples.

C Premise selection - ablation study

We investigate what contributes to the performance of our retrieval model by reducing its expressive power to a 1-layer transformer (first experiment), as well as removing our negative sampling strategy (second experiment). We observe a significant drop in recall with the changes.
Table 2: Retrieval metrics (top-k recall) comparison. On the test dataset, we measure percentage of situations, where given a proof state, the ground truth premise has been retrieved among top-k according to the PSM model. The 6L transformer + neg. entry refers to a model utilizing our negative sampling strategy with 1536 additionally sampled negatives (see A.2 for details).

<table>
<thead>
<tr>
<th>Model</th>
<th>recall@1</th>
<th>recall@4</th>
<th>recall@8</th>
<th>recall@16</th>
<th>recall@64</th>
<th>recall@128</th>
</tr>
</thead>
<tbody>
<tr>
<td>1L transformer</td>
<td>0.168</td>
<td>0.347</td>
<td>0.447</td>
<td>0.565</td>
<td>0.663</td>
<td>0.809</td>
</tr>
<tr>
<td>6L transformer</td>
<td>0.203</td>
<td>0.408</td>
<td>0.516</td>
<td>0.621</td>
<td>0.781</td>
<td>0.832</td>
</tr>
<tr>
<td>6L transformer + neg.</td>
<td>0.230</td>
<td>0.446</td>
<td>0.561</td>
<td>0.656</td>
<td>0.793</td>
<td>0.839</td>
</tr>
</tbody>
</table>

Theorem 1: lemma reachable_steps: "<exists> xs. steps xs <and> hd xs = s<s> <and> last xs = x" if "reachable x"

Original proof:
using that unfolding reachable_def proof induction case base then show ?case by (inst_existentials "[s<$>]; force) next case (step y z) from step.IH guess xs by clarify with step.hyps show ?case apply (inst_existentials "xs @ [z]") apply (force intro: graphI) by (cases xs; auto)+ qed

Our proof:
using that unfolding reachable_def by (fastforce dest: reaches_steps)

Proof 1: Our model is capable of proposing short and neat proofs when compared to the original.
Theorem 2:
lemma (in wf_digraph) iapath_dist_ends: 
\(\langle\text{And}\rangle u \ p \ v. \ iapath \ u \ p \ v \ \rightarrow u <noteq> v\)

Original proof:
unfolding pre_digraph.gen_iapath_def
by (metis apath_ends)

Our proof:
by (unfold gen_iapath_def) (auto dest: apath_nonempty_ends)

Proof 2: Exemplary proof that state-only model failed to close, whereas our PGLM+PSM managed to derive a fundamentally different proof without using metis - in contrast to original proof.

E Inputs comparison

Input 1: Exemplary input for PGLM+PSM model with top-4 premises. Input is a single sentence, here, for readability, split into multiple lines.

Input 2: Exemplary input for classical State-only model. Input is a single sentence, here, for readability, split into multiple lines.