Introduction to OpenAI's Proof Assistant for *Metamath*

Stanislas Polu
spolu@openai.com

Abstract

This document's primary goal is to serve as a tutorial for OpenAI's Proof Assistant for *Metamath*. It can also serve as a basic introduction to *Metamath* for people curious to discover it with the assistance of our models.

1 Introduction

The proof assistant is available at `https://mmproofassistant.azurewebsites.net/`. Please send an email to spolu@openai.com to request access.

Section 2 presents the main functionalities of the proof assistant. Section 3 provides an introduction to Metamath using the proof assistant. Section 4 explains how proofs can be shared and splitted among lemmas. Section 5 presents a search tool for theorems in Metamath's *set.mm* database and finally, section 6 propose a few additional exercises and their solutions.

The reader completely unfamiliar with Metamath can skip to section 3 for an introduction to its main concepts along with the capabilities of the proof assistant.

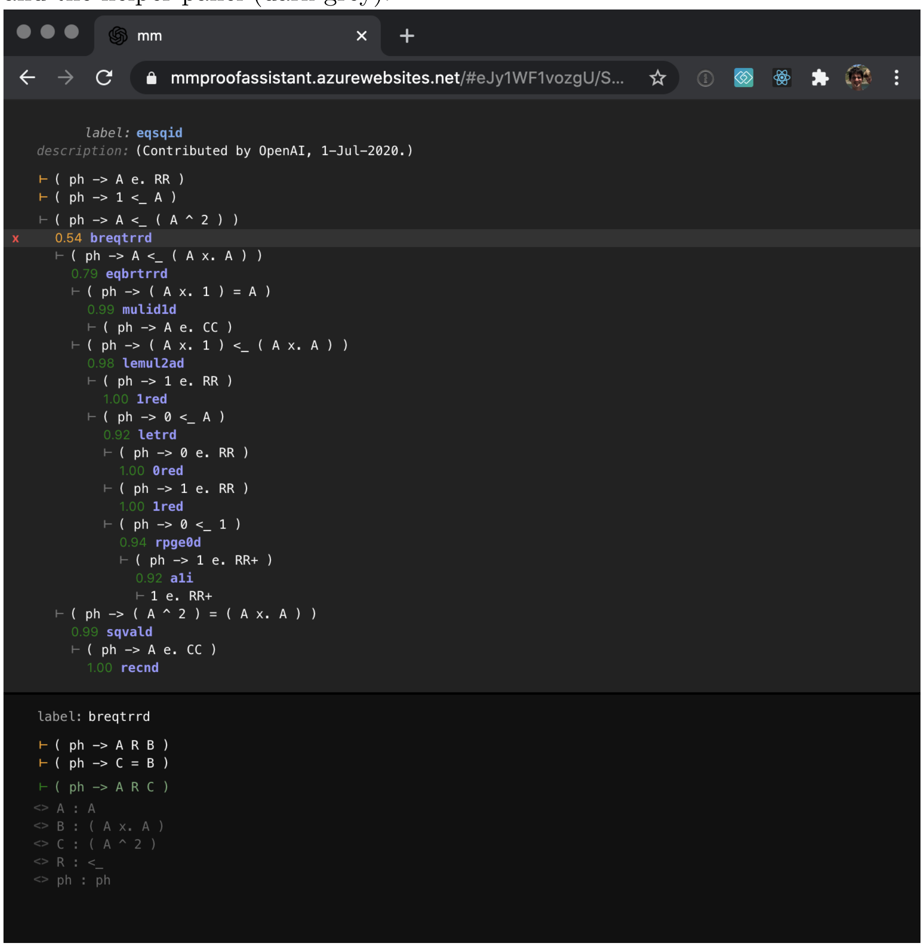
2 Main functionalities

The proof assistant lets you define a goal and prove it backward with the assistance of our model, providing manual overrides to steer it when necessary. As shown in Figure 1, there are two panels, the main proof panel and the helper panel. All commands are contextual to the currently selected line and accessible with the mouse through the icons appearing on its left-hand side. Line selection can be altered by `<Up>` and `<Down>` arrows.

We describe in the following list all the different line types in the main proof panel and their associated commands:

- *Label*: Lets you provide a name for the current theorem and proof. Available actions are:

Figure 1: Interface of the proof assistant with the main proof panel (light grey) and the helper panel (dark grey).



  - *Check Proof* (`<Ctrl>+<U>`): Checks the current proof highlighting the top goal or hypotheses in red if not grammatically correct, goals and tactics in green when closed. Once the proof is complete it triggers the generation of the proof in Metamath uncompressed format.
  - *Add Lemma* (`<Shift>+<Enter>`): Opens a popup to add a lemma by URL. The proof linked by the URL must be complete for the lemma to be accepted. The proof is extracted from the URL and the theorem added to the current context so that it can be used in the current proof. The model is not trained on it so it has to be entered or used manually.

- *Description*: Lets you provide a description for the current theorem and proof. The description is used to create the proof in Metamath uncompressed format once it is complete, so the description is intended to follow Metamath's conventions.

- *Lemmas*: Displays a lemma added to the context of the proof by name. When selected the helper displays the full statement. Available actions are:

  - *Discard Lemma* (`<Backspace>`): Discards the lemma *Open Lemma* (`<Enter>`): Opens the lemma's proof in a new window.

- *Hypotheses*: Prefixed by a yellow turnstile, lets you define one or more hypotheses for the current theorem. Hypotheses can be altered as the proof is constructed.

- *Goals*: Prefixed by a white turnstile, lets you initially define the statement you would like to formalize. A goal can be expanded by applying a tactic (a theorem and a set of substitutions), possibly producing new (sub-)goals lines. Available actions are:

  - *Automatically Expand* (`<Enter>`): Call the model to generate a set of tactics (theorem + substitutions) applicable to the current goal.
  - *Manually Expand* (`<Shift>+<Enter>`): Opens the helper in edition mode where a tactic (a theorem and a set of substitutions) can be defined manually.

  Note that goals and their sub-trees can be collapsed by clicking on the turnstile or alternatively using the `<Left>` and `<Right>` arrows.

- *Tactics*: Prefixed by a value representing the estimated confidence of the model for it, it displays the theorem being applied by name. Details about the tactic (theorem statement and substitution) are shown in the helper when the tactic's line is selected. Available actions are:

  - *Discard Tactic* (`<Backspace>`): Discards the tactic and the associated sub-tree.

The helper has three possible states:

- *Tactic Viewer*: Displays the currently selected tactic's details (theorem name, statement and substitutions). This state is also used to show lemmas's statements.
- *Tactic Input*: Allows a user to expand a goal by manually inputting a tactic. The manual expansion procedure starts by entering the theorem name and pressing `<Enter>`. Substitutions can then be specified. Once the tactic is fully specified `<Shift>+<Enter>` can be sued to submit the manual expansion.
- *Metamath Proof Viewer*: Once the proof is complete and checked, its associated Metamath uncompressed format is displayed in the helper, ready to be exported.

Exercise 2.1 *Open this proof, check it and verify it with metamath-exe.*

3 Crash course in Metamath using OpenAI's Proof Assistant

Before you get started you can get familiar with the *Metamath* formalism by reading: `http://us.metamath.org/mpeuni/mmset.html`. You can also skim through the *Metamath* book available at `http://us.metamath.org/downloads/metamath.pdf`.

Metamath is a formal system that implements a very simple substitution based kernel. The main database of the system is based on ZFC set theory and called set.mm. It contains 38k theorems with among many other things 74 of the Formalizing 100 theorems list.

Proofs can be explored through the proof explorer. Here's an example proof of $2 + 2 = 4$ from the definition of 2, 3, and 4: `http://us.metamath.org/mpeuni/2p2e4.html`.

3.1 Syntax

The *Metamath* / *set.mm* formalism relies on the fact that the *set.mm* grammar is unambiguous, meaning that there is only one way to write a given term. As an example, the theorem above is written:

```
|- ( 2 + 2 ) = 4
```

This is the only valid way to write this term, the syntaxes `2 + 2 = 4` or `( ( 2 + 2 ) ) = 4` being both out of the grammar.

OpenAI's proof assistant provides a way to check the syntaxic validity of the top-level term you wish to demonstrate.

Exercise 3.1 *Navigate to the proof assistant and authenticate. Enter the following term in from of the white turnstile:*

```
( 2 + 3 + 4 ) = 9  # Make sure to insert spaces between each token
```

Then press `Ctrl+U`*. You'll notice that the term will be highlighted in red, meaning that the term is not grammatically correct. Indeed, the only valid syntax for addition is* `( A + B )`*. Solution.*

Exercise 3.2 *Find the correct syntax for the term. The term should turn white as you hit* `Ctrl+U`*.*

Notes on the Interface The *label* is the name you give to your proof so it can be referred to by others. You can change it for whatever you want. As an example you can rename it here to `2p3p4e9`. The *description* is a free-form text input letting you describe the proof you're working on.

There is a loading indicator next to the label as you request a check of your current proof with `Ctrl-U`.

Next Steps You should review proofs online as well as in the ressources linked above to get familiar with the syntax. Here's a list of examples that may be useful to get you going.

```
|- ( ph <-> ph )
|- ( ( ph -> ps ) -> ( ( ps -> ph ) -> ( ph <-> ps ) ) )
|- ( 2 + 3 ) = 5
|- ; ; ; 1 2 3 4 = ( ( ; 1 2 + ; ; 1 2 3 ) + ; ; ; 1 0 9 9 )
|- 3 e. NN
|- ( ( A e. NN /\ B e. NN ) -> ( ( A + B ) - 1 ) e. NN )
|- ( K e. ( M ... N ) -> K e. ZZ )
|- ( N e. NN -> 3 || ( ( 4 ^ N ) + 5 ) )
|- ( A e. RR -> 0 <_ ( A x. A ) )
|- ( A e. CC -> ( * ‘ A ) = ( ( Re ‘ A ) - ( _i x. ( Im ‘ A ) ) ) )
```

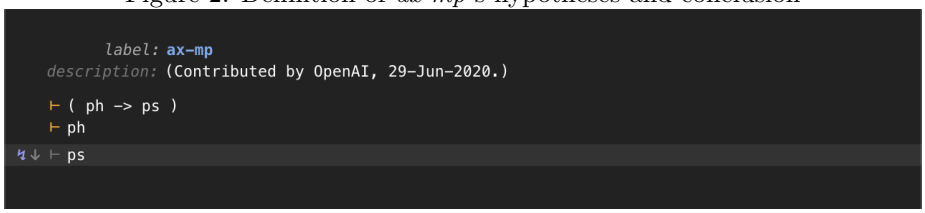
3.2 Hypotheses and Conclusion

A Metamath theorem consists of an ordered set of hypotheses and one conclusion. As an example, the following theorem has 6 hypotheses and 1 conclusion: `http://us.metamath.org/mpeuni/imo72b2.html`. Similarly, the modus ponens axiom has 2 hypotheses and 1 conclusion: `http://us.metamath.org/mpeuni/ax-mp.html`.

In the interface, the orange turnstile is used to define hypotheses. The modus ponens axiom can be represented as shown in Figure 2.

Exercise 3.3 *Create an hypothesis that supposes* `ps` *is true. Then add* `ps` *as the goal to prove. Press* `<Ctrl>+<U>`*, what happens? why? Solution.*

Figure 2: Definition of *ax-mp*'s hypotheses and conclusion



3.3 Proving propositions

Proving propositions in Metamath consists in the following:

- Selecting a theorem to apply from the database.
- Generate variable substitutions for the theorem conclusion and hypotheses such that the conclusion unifies to the current goal.
- Add the substituted hypotheses to the list of goals to prove.
- Finish the proof once there is no more subgoals open.

Exercise 3.4 *Let's attempt to prove the following simple tautology with the proof assistant:*

```
|- ( ( T. \/ T. ) \/ F. )
```

Enter the term as the top goal and check it is valid with `<Ctrl>+<U>`*. While the top goal is selected, press* `<Enter>`*. You'll notice there is a loading indicator next to the goal. The model is generating theorem statements and unifying substitutions to match the goal.*

The predictions of the model are stochastic (we sample from it), so sampling multiple times may not yield the same results. Sampling multiple times is actually a perfectly valid technique to force the model to explore more solutions.

Generally, in the current situation, the model samples `orci` *and* `orri`*. You can introspect the statement of these theorems as well as the substitutions that the model generated to unify to the goal by selecting the theorems' labels.*

A quick inspection of the remaining subgoals clearly demonstrates that `orci` *is the way to go. You can discard the* `olci` *completion by hitting* `<Backspace>` *while it is selected. You should now be in this state.*

From inspecting `orci` *it is easy to see that it simply proves a disjunction by proving its left-hand side. We probably want to reuse the same tactic on the remaining goal:*

```
|- ( T. \/ T. )
```

Let's do so manually. Select the sub-goal and press `<Shift>+<Enter>` *to manually input your tactic. The focus is moved to the helper panel. Enter the theorem you want to use:* `orci` *and press* `<Enter>`.

The substitutions are pre-filled by the system to unify to the current goal and you have nothing else to do. You can submit your manual tactic by hitting `<Shift>+<Enter>`.

The proof is not finished there's one sub-goal left to prove. Use the model to prove it.

Check the proof. Solution.

4 Sharing Proofs and Working with Lemmas

The proof assistant web application is entirely stateless. The entire proof state is stored in the URL which can be copied and shared to others.

This system also allows defining and proving lemmas for a main proof. To refer to a lemma, simply select the proof label and click the `+` button on the left. A pop-up appears and you can refer to another proof by URL. The proof must be complete and checkable to be accepted as lemma (otherwise we wouldn't be able to enforce distinct variables properly).

Once added, the lemma can be used by its name. The underlying model not being trained on it, the lemma will never be used as part of automated goal expansions, but you can refer to it in the helper when manually expanding a goal.

Exercise 4.1 *Prove that* $A + 2 \leq A + 3$. *Use it as a lemma to prove that* $5 + 2 \leq 5 + 3$. *Solution.*

You'll note that in the exercise solution above, we use a non-obvious format for the statement involving a `ph`:

```
|- ( ph -> ( 5 + 2 ) <_ ( 5 + 3 ) )
```

This is called a natural deduction form; it is a convenient way to format proofs in Metamath. More information is available here: `http://us.metamath.org/mpegif/mmnatded.html`

5 Searching for theorems

The proof assistant provides a tool to search for theorems from the *set.mm* database. It is available at `https://mmproofassistant.azurewebsites.net/search`.

This tool provides two ways to search for theorems: using labels, or pattern matching on the hypotheses or conclusion of the theorems. In both cases, results are ordered by relevance.

To search for a theorem based on its label, you can use the label search input. To search for a theorem based on its expression, use the hypotheses

Figure 3: Interface of the search page

inputs (prefixed by a yellow turnstil) or the conclusion input (grey turnstile). For example, to find *ax-mp*, you can use `ph` and `( ph -> ps )` as hypotheses, and `ps` as conclusion, or `ax-m` as label.

When you search for a theorem by its expression, you rarely know it exactly. To help with this, two RegExp-inspired wildcards can be used:

- `.` : replaces any Metamath constant or variable (ex: `Scalar`, `A`, `i^i`, ...)
- `.*` : replaces any sequence of space separated Metamath variable or constants

So, for example, to find *ax-mp*, you can simply search for `.` and `( . - > . )` as hypotheses, and `.` as conclusion.

Some remarks about the tool:

- The number of results displayed is limited to 20.
- The order of the results for the labels is determined by the Levenshtein distance.
- To use `.` and `.*` as defined in metamath, you have to escape them. For example, to find *dpval*, you can use `( . \. . ) = _ . .`
- The order of the hypotheses does not matter. The order that matches best is automatically detected.
- If there are several theorems that contain exactly the input expression, the shortest result appears first.
- If no theorem in *set.mm* matches the search expression, the theorems that best match the prefix are returned.

6 Additional Exercises

Exercise 6.1 *Prove* $10 + 12 = 22$. *Do so using only the model automated expansions. Solution.*

Exercise 6.2 *Prove* p *prime*, $a \in \mathbb{N}, b \in \mathbb{N}, p|(a*b) \implies (p|a) \vee (p|b)$. *Solution.*

Exercise 6.3 *Prove* $A^2 - B^2 = (A - B) * (A + B)$. *Solution 1 (by hand). Solution 2 (short)*