

Sophize Mathematics Library

Abhishek Chugh¹[0000–0001–6765–5229]

Sophize Foundation, Bengaluru - 560066, India abc@sophize.org

Abstract. Sophize is a novel mathematics library and discussion platform with a mission to help our users find and organize mathematical proofs. We present Sophize's novel yet familiar knowledge organization scheme that is used to represent a wide variety of proofs. With this scheme at its core, we have engineered a platform to logically aggregate knowledge from multiple sources. The platform curates knowledge from published research, encyclopaedias such as PlanetMath and Wikipedia, informal computer programs, and formal systems such as Metamath; and allows its users to run a federated search over them. To make this happen, two developments were necessary. First, we came up with the concept of 'proof-generating machines' and developed an open plugin architecture that allows proofs from computer programs to be generated as required. Second, we extended the Markdown language to represent the connections between mathematical objects found across various sources of knowledge. In addition, we also utilized this new language to create a novel communication system built specifically to aid mathematicians in solving problems collaboratively.

Keywords: Sophize · knowledge organization · proof-generating machine · belief set · metamath · markdown · argument graph · proof graph · semantic data

1 Introduction

The Sophize Platform

Sophize is a novel mathematics library and discussion platform. The author has developed the platform over the course of the last two years at <https://sophize.org>. Sophize's primary mission is to help users find existing proofs of mathematical statements, to discover new proofs, and to utilize this knowledge in their work. We combine knowledge from multiple resources and have accumulated thousands of definitions, theorems, and proofs from a wide variety of sources, including some published research, the PlanetMath encyclopedia, Wikipedia, and the Metamath formal system [2].

Motivation

In the past couple of decades, the Internet has revolutionized our life by giving us access to seemingly unlimited information. Information from a wide variety of topics ranging from human medicine to rocket science is now just a few

keystrokes away. However, even with this information we seem to have trouble understanding the complex world around us. In fact, dealing with the vast amounts of information thrown at us seems to be one of the primary problems of our generation. While we have access to a wide range of opinions on almost every topic imaginable, it has become increasingly difficult to decide which one to trust. It seems that you can find "proofs" for anything and everything we would like to believe.

Thus the author's primary motivation for this project is to create a library that not just lays out all the information of the World but organizes it in a way that helps us make better sense of it. A library that helps us access justifications from a variety of viewpoints and helps us collaboratively and objectively evaluate issues in the justifications or the viewpoints they arise from. Clearly, this is a challenging goal.

We begin our journey with a focus on pure Mathematics, where knowledge is objectively less complex as compared to other fields. In mathematics, only logically sound arguments are acceptable as justifications. These arguments develop from different foundational theories and logical consistency is the primary criteria of correctness for them. Thus, as a Mathematics library, Sophize's goal is to help our users access to mathematical proofs (chains of arguments) from a variety of foundational theories (extended to 'belief sets' as described in Section 2.5) and help us collaboratively and objectively evaluate issues in the argument or the belief sets they arise from.

With our current scheme we can incorporate almost all mathematics. A natural extension would be to model observational data and develop criteria to quantify empirical correspondence in order to have the ability to incorporate scientific knowledge. But, for now we are focused on building a state-of-the-art Mathematics library that is valuable to our community.

Proofs in Sophize

Mathematical proofs are based on a variety of foundations such as ZFC, intuitionistic logic, and type theory. The arguments used in any proof are considered valid or not based on criteria that can vary. Most academic mathematics is peer-reviewed and published, but some mathematical proofs can be found in community curated sources such as Wikipedia. Proofs can also be algorithmically generated, and at the highest level of verification, they are represented and verified using a formal system.

Sophize combines such an expansive range of proofs into a dense graph of propositions and logical arguments that aggregates knowledge from several documents and other data sources. We use this graph and combine it with the set of foundations and verifications chosen by each user to create proofs tailored to their needs. This introductory video gives an overview of the platform's offerings: <https://youtu.be/Wb1JbW9Otek>.

This work can also be seen as a step towards formalizing the network of information that exists in the connections of mathematical objects. The committee on planning a global library of the mathematical sciences recognized that this

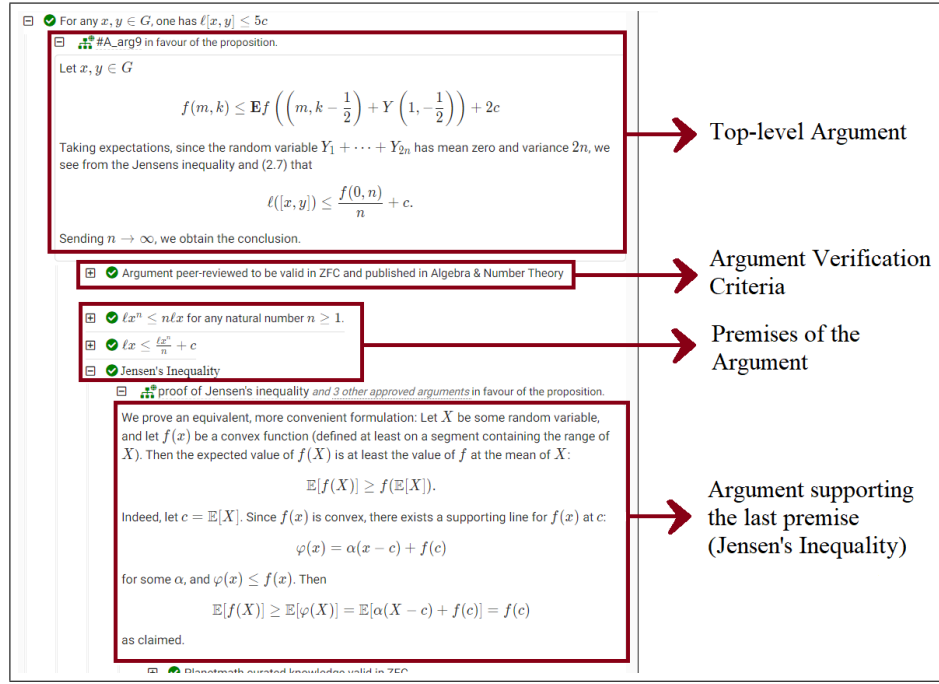


Fig. 1. A proof displayed as a graph (tree) of arguments and propositions on the Sophize platform.

network is largely unexplored, and formalizing it has tremendous potential to accelerate math research [1].

Hence, we required novel knowledge organization techniques to connect mathematical entities from such a wide range of formal and informal knowledge sources. And thus, one of the main challenges towards building the Sophize platform was to create a simple way to embed mathematical entities on the web.

Contribution

In this paper, we present the novel concepts and features of Sophize Mathematics Library. First, we expound Sophize's scheme to build a trust-focused logical network of mathematical knowledge. In section 3, we describe Sophize's proof-generating machines which are computer programs that generate proofs that integrate with Sophize's knowledge graphs.

We then present some features of Sophize Markdown, an extension of the Markdown language that helps present Sophize's knowledge graphs to its users. It is convenient enough to be used for casual discussions of mathematical ideas over the web. It is also powerful enough to embed mathematical entities such as

definitions, theorems, and proofs from various sources, including formal systems. It thus plays an integral role in the two problems that we have mentioned above.

Section 5 discusses some other challenges and contributions made towards making Sophize a modern mathematics library. Finally, Section ?? concludes the paper.

2 Sophize’s Knowledge Organization Scheme

At its core, Sophize’s approach is to simply keep track of all arguments in favour or against any proposition. Use of some form of arguments is the common factor amongst all mathematical (and perhaps all scientific) knowledge. We use this fact to unify knowledge from multiple sources in a meaningful way as described below.

We use the set of all known arguments to track knowledge from a variety of foundational theories. For theoretical knowledge, i.e., knowledge that doesn’t involve empirical data, the primary criteria of validity is internal (logical) consistency. Thus, for each foundational theory (‘belief set’ described below), we also keep track of any contradictions that arise from those foundations.

2.1 Core Concepts

Sophize uses the following concepts to logically organize theoretical knowledge.

*A **resource** is an abstract concept inherited by all other top-level concepts like terms, propositions, arguments, and belief sets. Each resource has a URI and contains fields such as search tags and citations. A **URI** consists of two parts: a namespace-like identifier called its dataset-id, which indicates the data source, and a resource-id that specifies the resource type and its unique name in the data source. The dataset-id may be omitted if it can be inferred from the surrounding context.*

For example, the Pythagorean theorem (a **Proposition**) represented in the Metamath project may have the URI *metamath/P_pythagorean* and the definition of cone (a **Term**) extracted from Wikipedia may have the URI *wiki/T_cone*. When used inside another resource in the same (‘wiki’) dataset, cone’s definition can be referred to simply as *T_cone*.

The same URI scheme will be used in this paper to refer to various terms, propositions, arguments, etc.

*A **term** is a clearly defined entity that can be used to make up a valid proposition. It can be a mathematical object, operator, symbol, data structure, algorithm, or even a person. ‘Meaningless’ primitives in formal theories are also categorized as terms.*

*A **proposition** is a grammatically valid statement that can be either true or false. Axioms, theorems, conjectures, hypotheses, lemmas, corollaries, and converses are all classified as propositions.*

An **argument** is a set of propositions called premises along with a concluding proposition that is claimed to follow from the premises. In addition, most arguments include supporting text that explains how the conclusion follows from the premises. A proof is seen here as a directed graph of arguments and propositions.

A **belief set** is roughly the set of axiomatic propositions that make up the foundations of a theory. Belief sets will be described in more detail in subsection 2.5.

Loosely defined, a **proof-generating machine** (PGM) is a computer program that generates the proof for an input proposition. PGMs are described in more detail in section 3.

2.2 Argument Graph

An argument can be seen as a simple graph (see Fig. 2) with two types of nodes - (a) a single node representing the argument itself and (b) a set of proposition nodes representing the argument's conclusion and premises. The edges of this graph are directed and go from (a) premise nodes to argument node and (b) argument node to conclusion node.

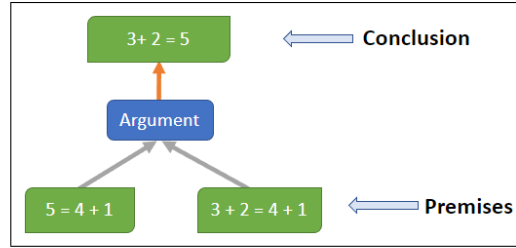


Fig. 2. An argument represented as a graph.

A set of arguments can thus be represented as a single graph that we call an **argument graph** (Fig. 3).

2.3 Generating Proof Graphs

An argument graph along with a set of set of axioms results in a **proof graph**. Any reachability algorithm starting with the axiom nodes is sufficient to create the proof graph (See Fig. 4).

A proof graph is a directed acyclic graph and contains complete proofs (including all alternate proofs) for all its propositions.

Note that an argument graph combines with different sets of axioms to generate different proof graphs. The same argument may be utilized in multiple belief sets (Fig. 6).

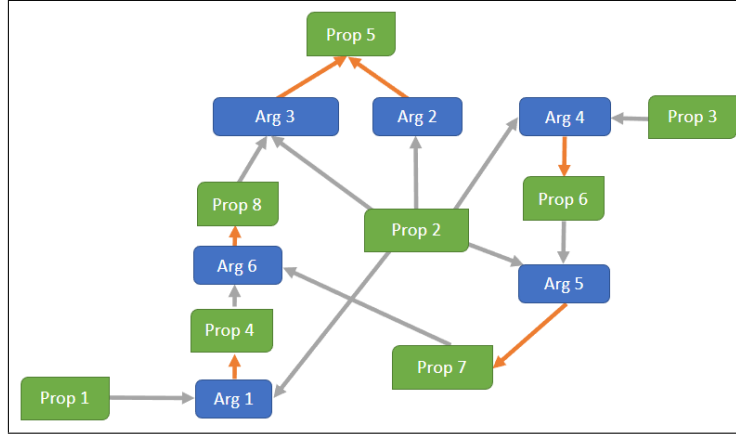


Fig. 3. A set of arguments forming an argument graph.

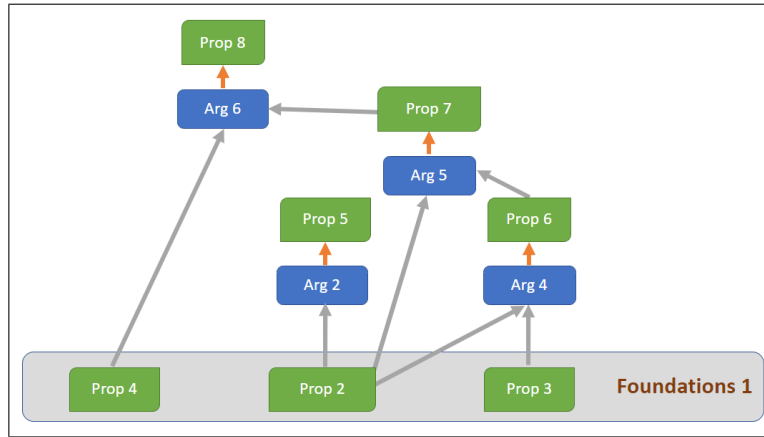


Fig. 4. Proof graph from argument graph (Fig. 3) using 'Prop 2', 'Prop 3', 'Prop 4' are axioms.

2.4 Validity of Arguments

The correctness of proofs obtained by the above process is contingent on the validity of the arguments in the argument graph. The validity of an argument is, in principle, independent of other arguments and objectively verifiable. Formally defined logistic systems indeed have a mechanical procedure to independently verify validity of arguments.

However, only a small section of Mathematics is developed using formally verified arguments. Vast majority of mathematics is verified via peer review, some of this knowledge is curated by experts in books, encyclopaedias. The claim

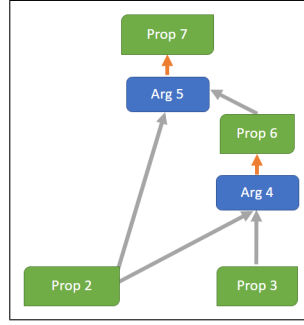


Fig. 5. Proof of 'Prop 7' from proof graph in Fig. 4 uses arguments 'Arg 4' and 'Arg 5'.

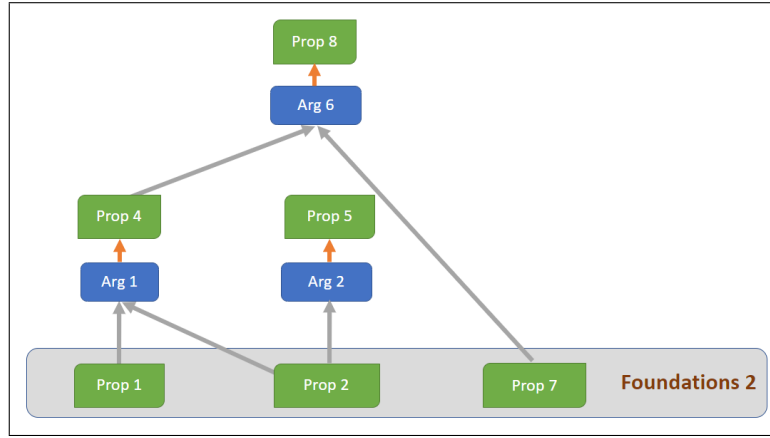


Fig. 6. Proof graph from the same argument graph (Fig. 3) using 'Prop 1', 'Prop 2', and Prop 7' as axioms.

of validity of proofs (arguments) is judged based on the various parameters such as academic standing of the claimant, reputation of the peer reviewing journal, the history of errors of the book or encyclopaedia containing them, etc. Clearly, these criteria are subjective and each individual's criteria of accepting validity of an argument can vary to some extent.

Sophize doesn't promote or endorse any criteria. Instead, it allows its users to choose the criteria they believe in. The validity criterion is modelled as a proposition and is attached to arguments as a premise.

To easily manage validity criteria, the relationships between validity criteria are also maintained in the argument graph using argument-like dependency graphs. Eg. "Published in journal with SJR score > 0.67" → "Published in Annals of Mathematics". By maintaining relationships like above, Sophize allows

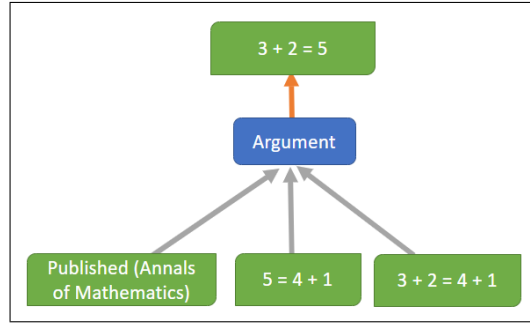


Fig. 7. Validity criteria as an argument premise.

its users to conveniently choose multiple sources they consider to be reliable with a single manually chosen criterion.

2.5 Belief Sets

A proposition is true if it is the conclusion of a valid argument and the premises of the argument are also true. Of course, this means that we need to start with some propositions that are assumed to be true without further justification. Traditionally, these are mathematical axioms and are considered to self-evidently true. In mathematical inquiry, theorems are proven/disproven in one or more multiple theories, each with its own set of axioms. Each theory may have its theoretical advantages and practical uses and there are no universal basis to choose one theory over another when trying to assert a proposition's truth/falsity.

Thus, Sophize allows any set of propositions to be considered true without evidence. Such a set of propositions is called a **belief set**. Propositions representing argument validity criteria are also part of belief sets. All propositions that are considered to be true in a belief set without proof are called its 'unsupported propositions'. In addition, a belief set can also contain a number of 'proof-generating machines' (e.g. for managing axiom-schemas) that will be discussed later. For ease of use, belief sets can also contain other belief sets.

The truth-value of a proposition is only defined within a belief set - there is no notion of absolute or universal truth. Sophize creates proof graphs for all its belief sets.

2.6 Tracking Inconsistencies

If a proposition and its negation is proved within a belief set, it is inconsistent. As noted by the principle of explosion, any proposition can be proved from such a contradiction, thus making the claim of truth of any proposition practically worthless.

Thus, it is important to track and report contradictions in a belief set. To do that Sophize has a semantic notion of negation of a proposition. The premises or

conclusion of an argument can be negations of propositions. When a proposition and its negation are proven, the system reports such contradictions for every belief set. When generating proof graphs from the argument graph, a contradicting proposition cannot be used a premise for any argument.

2.7 Advantages and Limitations

Efficiently modelling different axiomatic systems that have a large overlap of arguments (eg. euclidean/parabolic/hyperbolic geometry, ZF/ZFC etc)

Conjectural knowledge (Riemann hypothesis). Practical utility of conjectural knowledge - eg. a lot of information security systems assume "P is not NP" Proof tailored to needs. Proof graph can be easily converted to prose too.

3 Proof-Generating Machines (PGM)

A lot of mathematical knowledge is computed. Computation machines (e.g. calculators, algebras) can be seen as performing the following tasks

- Parsing the input into a language that can represent math formulae.
- Finding and outputting an equivalent form of the formula.

Thus, their focus is not on finding the proof but only finding the equivalent (simplified) representation of the input formula (Although many computer algebras do show the steps of the computation.). With Sophize, the focus instead is to generate proofs of the input propositions. A **Proof-generating machine** (PGM for short) is a computer program that generates a partial proof graph against or in favour of an argument. PGMs can also have the capability to perform tasks 1 and 2, mentioned above.

PGMs take in some text as input which is parsed by the each PGM as they wish. Typically the text is treated either as a proposition (e.g. ' $3 + 5 = 8$ ') or a formula (eg. ' $5+6$ ') and in the latter case the PGM is responsible for finding an appropriate proposition (eg. ' $5 + 6 = 11$ ') for the given input. The PGM then generates a proof graph for this proposition and returns back to the user.

The Sophize platform itself merely forwards the request to the associated PGM and performs some checks on the output returned. Any http server that can process a proof request can be easily plugged in to Sophize platform using its HTTP address. One server can implement multiple PGMs.

3.1 'Active' PGMs

As with other parts of Sophize, the validity of the output of a PGM is not taken for granted. Each PGM must specify all the propositions (premises) that it will use in the root nodes of the proof graph it returns. In addition, a PGM may also delegate parts of proofs to another PGM. These PGMs must also be specified.

A PGM is considered 'active' in a belief set iff:

1. all of its premises are true in the belief set.
2. all the PGMs it delegates-to are active.

If the PGM returns a proof graph that utilizes *As* with an argument, one or more premises is used to indicate the validity of the PGM (Eg. tested by *X*, verified by some committee, another program etc). Thus, to be considered active in a belief set, the PGM must have a verification criteria that matches the belief set's requirements. The output of a PGM is not used in belief sets in which it is not active.

3.2 PGM Response

The response of a request to any PGM has the following components:

- Truth value, i.e., whether the PGM considers the input proposition (provided or computed) true or false. PGM can also indicate that it doesn't understand the input or prove/disprove the proposition.
- Proof graph (or a subset of proof graph) generated by the PGM in favour or against the input proposition. This component is skipped if only computation results are requested.

Note that unlike other propositions and arguments on Sophize, these are not stored on disk. They have a temporary URI specified in a slightly different format. The leaves of the DAG can have two kinds of arguments:

1. Arguments whose the premises are limited to the premises of the PGM.
2. Special 'Arguments' that indicate that the generation of the remaining proof must be delegated to another PGM. In case of large proofs, a PGM may also delegate to itself after generating some (say 100) arguments.

3.3 Materializing results

Sophize allows saving the fact that a machine has returned a valid proof for a proposition (not the actual proof itself which can be quite large). Then this proposition can be used in further proofs without the need to invoke the proof generation every time. The proof, of course, can be requested by the users whenever required.

4 Sophize Markdown

In the author's view, it is crucial to maintain the proofs graphs for building trustable knowledge. However, the presentation of proofs as graphs to users is perhaps not very convenient. Traditional text based narration remains indispensable when presenting knowledge to the reader.

We have extended the Markdown language - a lightweight markup language for creating formatted text using a plain-text editor - for building a rich narration

interface. It allows users to quickly encode and present semantic data, to show the truth status of propositions available in prose, and to effortlessly access proof graphs.

With Sophize Markdown, we can easily add $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ by enclosing it between two \$ signs (or \$\$ for display math). The detailed specification of Sophize Markdown is available here. Many of its features are demonstrated at: <https://youtu.be/5UYOpQwcjCk>. We highlight the prominent features here.

4.1 Embedding semantic data

With Sophize Markdown we can easily link resources such as terms, propositions, and arguments. Links can be added using the pound sign (#) and the resource's URI. There are multiple resource link options that can be used for changing the behavior and text of the resource link. The following examples demonstrate these options.

Example 1

```
#P_cosine_law|UC generalizes the #P_pythagorean for all kinds of #(T_triangle, 'triangles')
```

The above code is rendered as:

Law of cosines generalizes the Pythagorean theorem for all kinds of triangles.

The link options 'UC/LC' makes the link text upper-cased or lower-cased.

Example 2

```
#T_matrix_multiplication|NO_LINK is a #(T_commutative_property, 'non-commutative'|NAV_LINK)
#T_binary_operation.
```

The above code is rendered as:

Matrix multiplication is a non-commutative binary operation.

This example shows the various link behaviours. The first element shows '#wiki/T_matrix_multiplication' on mouse-hover. The second link navigates the page to a different URL. Clicking on the third link pops up a modal dialog box like so:

Example 3

```
# Conic section
```

A conic section (or simply conic) is a curve obtained as the intersection of the surface of a #wiki/T_cone with a #wiki/T_plane. There are three types of conic sections.

Binary Operation

[#wiki/T_binary_operation](#)

INFORMAL

In mathematics, a binary operation or dyadic operation is a calculation that combines two elements (called operands) to produce another element.

References

[Binary Operation \(Wikipedia\)](#)

Fig. 8. Modal dialog box for the term binary operation. These boxes can also have content that uses all three types of links.

```
## Ellipse
#oxford/T_ellipse|EXPAND #wiki/P_ellipse_area|EXPAND

## Parabola
...
```

The above code is rendered as:

Conic Section

A conic section (or simply conic) is a curve obtained as the intersection of the surface of a cone with a plane. There are three types of conic sections.

Ellipse

An ellipse is a regular oval shape, traced by a point moving in a plane so that the sum of its distances from two other points is constant. The area $A_{ellipse}$ enclosed by an ellipse is

$$A_{ellipse} = \pi ab$$

where a and b are the lengths of the semi-major and semi-minor axes, respectively.

Parabola

...

In the above example, the ‘EXPAND’ option expanded the definition of the term, and statement of the proposition in place. Thus we were able to easily re-use concepts and theorems already extracted from other sources (say, Wikipedia and the Oxford dictionary in this case).

4.2 Truth Status

Sophize markdown adds an icon next to each proposition link. This icon indicates whether the proposition has been proven or dis-proven (or both) in the currently selected belief set using a check-mark or a cross. Clicking on the icon brings up the proof graph as shown in Fig. 1. If the users switches from one belief set to another, the icon is updated based on the truth values in the new belief set but other parts of the article are unchanged.

Similarly, an icon is added next to a proof generating machine which indicates whether the PGM is active or not.

4.3 Formal Language Support

In formal languages, definitions of terms, statements of propositions, and supporting argument text may be parse-able by an external parser. We can convert the externally parsed output into Markdown in such a case, where each term is automatically linked to the appropriate resource. This provides a convenient interface, where the user types in the native language, and the final output automatically allows users to explore all concepts that make up the input statement in depth. Currently, this is demonstrated in the use of Sophize Markdown with the Metamath language.

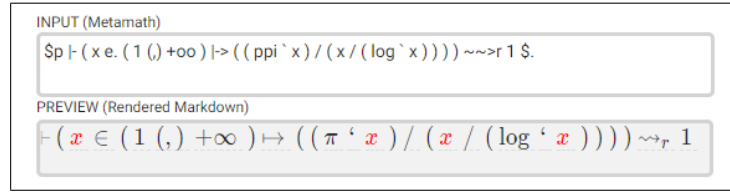


Fig. 9. The prime number theorem written in Metamath language and rendered as Markdown in real time.

5 Building a Modern Mathematics Library

The argument and proof graphs in help us build a network of information that spans across a wide variety of sources (including live results from PGMs). Using

different belief sets, users can view mathematical knowledge arising from different theoretical foundations and choose what they consider to be reliable. With Sophize Markdown, users can easily embed semantic information, thus making it convenient to explore the network of knowledge that is being built. However, a few more practical issues need to be overcome to build a modern scalable library with reliable information.

5.1 Semantic Data Extraction

Utilizing the full range of features offered by the Sophize platform requires the availability of semantic information. However, extracting semantic information from unstructured or semi-structured literature is a difficult problem. This problem can be seen as composed of several sub-problems:

- Finding definitions and propositions in prose.
- Associating the appropriate entity with a term used in prose for quick lookup.
- Identifying arguments and along with premises and conclusion.
- Identifying duplicate definitions and propositions used in different sources.

Solving these problems would need manual effort (e.g. crowd-sourcing) combined with automated extraction tools. Recent progress in machine learning and natural language processing techniques does make this problem seem tractable.

Connecting Across Datasets The last sub-problem mentioned above is crucial to help make connections across datasets. For example, once we have identified that the ‘Jensen’s Inequality’ in a published paper is same as the Planetmath proposition with the same name, Planetmath’s proof is automatically made available to the reader of the published paper.

However, figuring out if two sources are referring to the same proposition or a different version of the same proposition is often unclear. When working with formal systems using different languages, another related problem arises. There is no general way to claim equivalence of propositions across different formal systems. There are some projects, like Formal Harmony and theory morphisms, that are trying to understand in what sense these propositions are same and how we can translate one to another. But these projects are still in early stages and do not provide a general solution.

However, we can utilize curated translations, such as those available from Logipedia, by representing them using arguments. The conclusion of such an argument is the proposition in target language and the premise would be the proposition in source language. For bi-directional translations two such arguments will be required. Such translations do not stand on a very firm foundation and that should be indicated in the validation criteria of the argument. A user can choose not to accept such criteria. In that case, these translations will not be used in the generating proof graphs. Thus we can create proof graphs that have arguments from different languages.

There are some cases where translations occur on firmer grounds, e.g., use of theory morphisms or translations of HOL-light proofs to Metamath. The author has not studied these cases in detail but expects that it to be possible to formulate such cross-language translations using the argument graphs.

5.2 Data Management

Sophize divides its data into groups called datasets for copyright management and data access rules. For example, the 'wiki' dataset that extracts knowledge from Wikipedia mathematics glossaries releases its data under 'CC-BY-SA' license, whereas the 'metamath' dataset is public ('CC0'). Our signed-in users are free to create their own datasets and we allow them to choose who can read, write or comment in these datasets. Another interesting access control is the restriction of use of resources (especially argument-validity criteria) outside an otherwise open-to-read dataset.

Information can be added to Sophize platform using intuitive web forms. However, it is infeasible to extract information from large repositories in this manner. Sophize publishes its data schema and helper libraries (explained in Section ??) in multiple languages. The extracted information is can thus be stored as text (JSON) files in a repository. Sophize provides dataset owners the ability to sync knowledge such repositories to the central Sophize database.

5.3 Search Interface

Sophize allows its users to run a federated search over its resources using an Elasticsearch server. We allow users to search a space-insensitive and case-insensitive search over L^AT_EX formulae by encoding L^AT_EX formulae as lexemes and an Elasticsearch analyzer (See section ?? for sources) that removes spaces and lower cases the encoding.

For example the, latex formula $E = mc^2$ is converted to *Ej9rjmc9v2mMm*. Note that the encoder converts spaces and new-lines into the character 'j' or 'J' which are removed by our Elasticsearch analyzer at the time of searching the documents.

5.4 Collaboration

The Sophize platform includes an innovative mathematics communication interface designed to help mathematicians discuss mathematics and collaboratively discover new proofs. The design focuses on making the collaboration more productive and enjoyable by building the right set of technical tools that aid in effective organization and summarization of existing progress. With Sophize Markdown at its core, Sophize's collaboration interface[?] ([demo](#)) is able to provide key features such as easy linking of existing math content, L^AT_EX support, and live preview comment drafts.

6 Conclusion

TODO: What criteria are met based on the 2013 report

TODO: How much math can be represented. Eg. three valued logic

TODO: Sophize's limitations. In "Potential Value of a Digital Mathematics Library" in 2013 report. Sophize offers no way to more easily find analogies or weaker forms of reasoning like "If A is true, then B is true. B is true; therefore, A becomes more plausible."

Acknowledgements A part of the research presented here was supported by the International Mathematical Union.

References

1. on Planning a Global Library of the Mathematical Sciences, C.: Developing a 21st century global library for mathematics research (2014)
2. Megill, N.D.: Metamath: A Computer Language for Mathematical Proofs. Lulu Press, Morrisville, North Carolina (2019), <http://us.metamath.org/downloads/metamath.pdf>