Graph based analysis of mathematical knowledge structure on Metamath

Reuel R. Ribeiro, Valmir C. Barbosa, and Flavio B. Gonzaga

Núcleo de Ciência da Computação, Universidade Federal de Alfenas,
37133-840, Alfenas - MG , Brazil
{reuel,fbgonzaga}@unifal-mg.edu.br
{valmir}@cos.ufrj.br

Abstract. Axioms and thousands of theorems compose the modern mathematical body. Software assisted proofs is a reality and Metamath serves this purpose. We extract from Metamath proofs and represent them through a directed acyclic graph. Modeling mathematical knowledge in this way makes it possible to study the intrinsic characteristics of the network formed by axioms and theorems. We perform a degree distribution study on the graph, decompose the graph into layers, look the length of mathematical proofs, and a study about the reachability of axioms to theorems in order to determine important axioms. This work proposes to analyze this organic structure and how elements relate to each other.

Keywords: metamath, graph analysis, mathematics, lognormal, power-law

Introduction

Aristotle was an important person to give to Logic a status of science of ideas and the mind process [1]. He is the one who establishes elements such as axioms, definitions, theorems and hypotheses, which work as tools to describe the knowledge in a methodical and deductive way. According to [2], "Aristotle saw a deductive science as one structured building of truths chained together through logical relationships, founded over some fundamental per-conceptual ideas that cannot be demonstrated but rather taken from granted".
It is necessary to have a start point to begin crafting theorems; such elements are called axioms, and they do not need to be proved from previous axioms because they are essentially simple on themselves and can be easily verified to be true. "For the sake of proof, the first principles must be indemonstrable, since otherwise regression would proceed to infinity. And how do you know the truth of the first principles? By induction." [1, p. 39].

1 Metamath

"Metamath is a computer language and a computer program for achieving, verifying, and studying mathematical proofs at a very detailed level" [3]. It works through simple rules of replacement to prove theorems. Its relevancy for this work relies on its knowledge database, which is an ASCII text file, whence axioms, theorems and their relationships can be extracted.

Metamath is also available at `http://us.metamath.org/` to be explored. Each axiom, theorem, and some other elements can be accessed at the address `http://us.metamath.org/mpegif/<name>.html` where **<name>** is the name of the desired elements. A thorough explanation about how to read the mathematical proofs is available on the website itself.

Metamath not only deals with fundamental mathematics such as *theorem set*, but also has some other major areas of knowledge. The present work is concerned exploring the *Metamath Proof Explorer* area of Metamath, where the foundation of mathematics is represented. The other sections such as *Hilbert Space Explorer* and *Higher-Order Logic Explorer* are endowed by the *Metamath Proof Explorer* section.

Metamath is readily available to download; it comes with a group of **.mm** ASCII files where the mathematical knowledge is contained. The file used for this work is the **set.mm** one and it is indiscriminately referred throughout this paper as "metamath database", "database" or "knowledge base". More instructions about the file format, organization, syntax summary and other helpful topics are found within the file itself and in [3, pp. 33, 91, 155].

1.1 Metamath structure

To give the reader a glimpse on the metamath database and how to interpret it, table 1 summarizes three concrete examples taken directly from it.

Metamath has more elements such as definitions, syntax definitions, constants and variables, but those are pertinent to metamath itself. The aim of this work is to only analyze how the axioms and theorems relate to each other, so it is enough to know, for a given theorem, which are the other axioms and/or theorems necessary to directly prove it.

Hitherto, the Metamath database contains 72 axioms [1] and 17505 theorems [2], not accounting for the users' mathboxes, which is explained and justified in section 5.4. In order to manipulate such information, it is convenient first to represent it on an adequate form with which it can be easily queried, manipulated and analyzed. For such purpose, based on all the details found on [3], a grammar has been written to parse the database file into a graph structure. It makes the knowledge extraction easier and less error prone compared to crawling the metamath website for instance. A set of lexical rules and syntactical rules in appendices A.1 and A.2, respectively, describe the relevant information contained in the metamath database.

[1] Considering `$a elements within metamath file whose name starts with "ax-"`

[2] Considering `$p statements within metamath file`

Metamath database (relevant lines)	**Mathematical representation**	**Interpretation**
`ax-1 $a |- (ph -> (ps -> ph)) $.`	Axiom (**ax-1**): $\vdash (\phi \to (\psi \to \phi))$	The first of three axioms of propositional calculus.[a]

`min $e |- ph $.`
`maj $e |- ( ph -> ps ) $.`
`ax-mp $a |- ps $.`
	Hypotheses:
	$\vdash \phi$ (min)
	$\vdash (\phi \to \psi)$ (maj)
	
	Axiom (**ax-mp**):
	$\vdash \psi$
		Inference rule known as *Modus Ponens*. The rule says that if ϕ happens, and the condition says that $\phi \to \psi$, than ψ holds true.[b]

`mp2b.1 $e |- ph $.`
`mp2b.2 $e |- ( ph -> ps ) $.`
`mp2b.3 $e |- ( ps -> ch ) $.`
 	
`mp2b $p |- ch $=`
`wps wch wph wps mp2b.1`
`mp2b.2 ax-mp mp2b.3 ax-mp`
`$.`
	Hypotheses:
	$\vdash \phi$ (mp2b.1)
	$\vdash (\phi \to \psi)$ (mp2b.2)
	$\vdash (\psi \to \chi)$ (mp2b.3)
	
	Theorem (**mp2b**):
	$\vdash \chi$
	
	Proof:
	1. $\vdash \psi$
	2. $\vdash (\phi \to \psi)$
	3. $\vdash \psi$ (using steps 1 and 2)
	4. $\vdash (\phi \to \psi)$
	5. $\vdash \chi$ (using steps 3 and 4)
		Theorem that extends the *Modus Ponens* in order to use two inferences.[c]
		The first two steps of proof use of hypotheses 1 and 2.
		The third step uses the previous steps with the axiom ax-mp as it uses two hypotheses as "inputs".
		Steps 4 and 5 follow a similar approach.
		The last step finishes the proof.

[a] http://us.metamath.org/mpeuni/ax-1.html
[b] http://us.metamath.org/mpeuni/ax-mp.html
[c] http://us.metamath.org/mpeuni/mp2b.html

Table 1: Elements from metamath database and their interpretation

2 Representation

This work proposes a graph-like representation of Metamath. To understand how the data is modeled from it, consider the arbitrarily chosen theorem **mpd** [3]. The portion of the raw metamath file necessary to describe it is:

```
1  ${
2    mpd.1 $e |- ( ph -> ps ) $.
3    mpd.2 $e |- ( ph -> ( ps -> ch ) ) $.
4    $( A modus ponens deduction.  (Contributed by NM, 5-Aug-1993.) $)
5    mpd $p |- ( ph -> ch ) $=
6      wph wps wi wph wch wi mpd.1 wph wps wch mpd.2 a2i ax-mp $.
7      $( [5-Aug-1993] $)
8  $}
```

Lines 1 and 8 begin and end, respectively, a Metamath scope [3, pp. 113-114].

Lines 2 and 3 describe hypotheses, **mpd.1** and **mpd.2**, respectively. A hypothesis in metamath is "a logical truth (such as 'assume x is prime') that must be established in order for an assertion requiring it to also be true." [3, pp. 105-106]. One analogy to grasp how hypotheses work is to visualize a theorem as a computer method which requires certain input parameters and then returns some result. As long as the parameters are correct, the result is also guaranteed; hypotheses are mandatory parameters and the theorem assertion is the return of the function.

Line 4 is a comment left by the author of the contribution.

Line 5 is the most important one with the theorem assertion (between the `$p` and `$=` symbols). It asserts that $\vdash (\phi \rightarrow \chi)$ and its proof is represented, in reverse Polish notation, between the `$=` and `$.` symbols. The names **wph**, **wps**, **wi**, **wph**, **wch**, **mpd.1**, **wch**, **mpd.2**, **a2i**, and **ax-mp** are all the necessary elements to construct a proof, which metamath then uses to assert $\vdash (\phi \rightarrow \chi)$ through adequate substitutions, which are thoroughly explained in [3, p. 114].

[3] http://us.metamath.org/mpegif/mpd.html

wph, **wps**, and **wch** are all variables necessary for metamath to know how to make substitutions and verify a proof. They stand for the Greek letters ϕ, ψ and χ, respectively. These variables represent well-formed formulas (wff) in metamath, meaning that they can represent other logical assertions [3, pp. 20].

wi is a syntax definition, and it says that "if ϕ and ψ are wffs each, so is the construction $\phi \rightarrow \psi$", which can be read as "ϕ implies ψ". Syntax definition is a way to produce a new valid wff from other wffs.

mpd.1 and **mpd.2** are hypotheses necessary only to prove the **mpd** theorem. No other theorem can use them because they are scoped to the **mpd** theorem. The scope is created by the `${` and `$}` symbols, although it does not affect the axioms and theorems defined within it. Axioms and theorems can be referenced later on in the file by any other theorem, contrary to hypotheses that can only be referenced within their own scope.

a2i is another theorem used in one of the proof's steps. It also has its own hypotheses. In order to **a2i** be used on the proof of **mpd**, **a2i** must have be previously proved.

ax-mp is, within metamath, an axiom which is similarly used as the **a2i** theorem to prove **mpd**, but it does not depend on a previous proved assertion; it stands on its own since it is an axiom.

How the substitutions are made and more thorough details regarding the file syntax, its elements and their meaning can be consulted on [3].

After describing all the main elements of such small portion of the metamath database file, the graph in figure 1 is proposed. Further elements for the **ax-mp** axiom and the **a2i** theorem are also shown on it; they do not appear on the file excerpt but are found within **set.mm** file. Notice that in figure 1 more elements could have been represented as nodes connecting or being connected by arcs, such as syntax definitions and constants, but that would litter the representation with nodes that are not important for the intents of this work. Even the hypotheses nodes will be discarded for the herein proposed analyses because they serve the internal mechanisms of metamath for making substitutions in order to verify a proof. They ultimately form satellites around their theorem nodes; therefore, they do not contribute in a special way to this research.

Axioms, hypotheses, and (already proved) theorems can be used to prove a new theorem. Axioms and theorems can be represented as nodes and the usage of one by the other on its proof can be indicated as an arc going from the supporting axiom/theorem to the theorem being proved. The hypothesis on figure 2a was shown only to illustrate how the graph could have been modeled. Since they are not of interest for this work, the model is going to be simplified even further as figure 2b shows, taking only axioms and theorems into consideration.

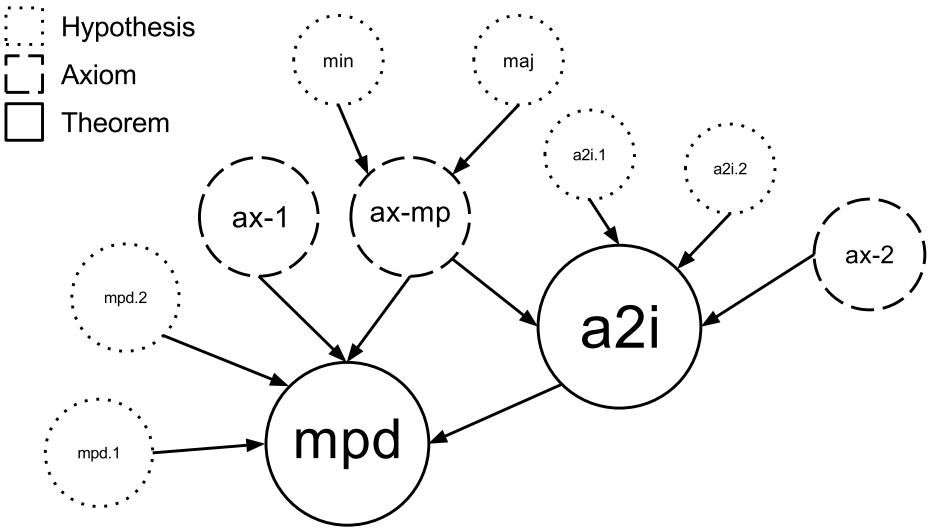
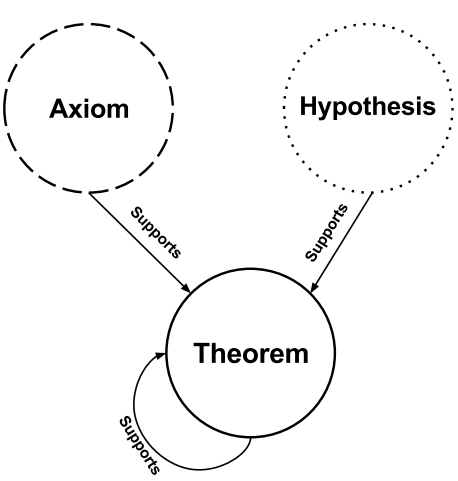
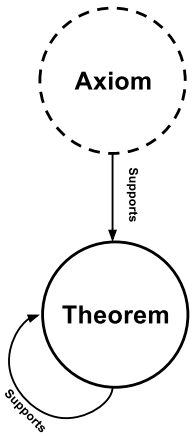


Fig. 1: Metamath database represented as a directed graph, with the arcs showing that ancestor nodes assist the construction of a proof for a new theorem, yet the arcs do not show *how* the proof is constructed nor do they show how many times one element has been used on a proof. Other elements such as variables, constants, and definitions are not contemplated for this illustrative modeling.



(a) More general schema to model the metamath database as a graph



(b) Simplified schema to model the metamath database as a graph of axioms and theorems.

Fig. 2: General schema for modeling Metamath as a graph

3 Graph storage

As the metamath database is fairly large (+140 Mb), it would be inadvisable to parse it every time or work with it directly. It becomes evident that a proper solution has to be determined to persist the parsed information. Using relational databases would not be appropriate to store a graph-like related information because relational databases work with foreign keys to relate information and costly join operations [4]. Besides, after storing it is necessary a proper way of traversing and analysing the graph. To take advantage of the already available graph database technology, Neo4j has been chosen to support this work. Neo4j is an avail for both storing the parsed information and querying for data in a more adequate manner, since it is a graph database with all the Create, Recovery, Update and Delete (CRUD) functionalities one would expect from a Database Management System (DBMS). To query data and manipulate the graph, it offers both a Java API and a high-level language called Cypher. Neo4j is thought to assign importance to the relationships more than the entities themselves, as it is usually done in relational databases. It means that to model and relate information is more flexible and expressive than in other systems that work with foreign keys or reduction maps to indicate how the information is related. To make relationships first class citizens, neo4j exposes the data throughout a graph model [4].

4 Graph structure

As axioms and theorems are modeled by the aforementioned methodology in section 2, it is quite easy to see that the output graph is a Directed Acyclic Graph (DAG): Axioms do not need to be proved from other axioms or theorems, so they are the source nodes of our graph. Theorems can only be proved from axioms or other erstwhile proved theorems. Also, there could not be cycles as it would mean that one theorem would need its own consequent to prove itself, which would be an illogical scenario. The metamath file structure does not allow for a theorem to refer to a not yet existing theorem; that means that lines on the metamath **set.mm** file can only refer to previous lines, which makes it impossible to form cycles in the proof structure. As explained before, the arcs go out from axioms to new theorems and from already proved theorems to new ones. If the arcs are reversed, then the graph can be seen as a dependency graph, where axioms would be sink nodes rather than source nodes and new theorems out point out to other already proven theorems.

5 Metamath intricacies

Parsing the file and analyzing the graph is not as straightforward as one might think. Some theorems within metamath are actually not useful theorems, or maybe some of them are just helping mechanisms that exist for very specific reasons which are not practical for this work. Such cases will be listed and briefly described. This is not a claim to include all pitfalls, but the ones that have been discovered while investigating metamath.

5.1 Theorems which are actually not theorems

idi - This theorem has been ignored from analysis as its page states "This inference rule, which requires no axioms for its proof, is useful as a copy-paste mechanism during proof development in mmj2. It is normally not referenced in the final version of a proof, since it is always redundant and can be removed [...]".

weq, **wel** and **wsb** - Those three elements once existed within metamath as axioms, but they have been reformulated as theorems. They have been constructed in that way to avoid overloading the $=$, $\in$, and $[x/y]\phi$ connectives, respectively. They work as a special construction within metamath, and their pages [4] [5] [6] state that each element is to be considered as a primitive syntax, even though each one "derives" from **wceq**, **wcel**, and **wsbc** syntax definitions, respectively. Therefore, they have been ignored for this work.

The **dummylink**[7] theorem serves only the purpose of filling a gap during the construction of a proof. It has been discarded as its description states: "This is a technical inference to assist proof development. It provides a temporary way to add an independent subproof to a proof under development, for later assignment to a normal proof step."

5.2 ax-meredith axiom and meredith theorem

A tree has been found beginning from the axiom **ax-meredith**. Performing a *depth-first search* (DFS) from this axiom led to a tree with the follow theorems as leaves: **ax1**, **ax2** and **ax3**. The purpose of this axiom is to demonstrate that the three axioms from the propositional calculus (**ax-1**, **ax-2**, and **ax-3**), the Carew Meredith's theorem, and Lukasiewicz's axioms are all equivalent systems. The reason for ignoring all this tree is that all tree elements state that "proof modification is discouraged" and "new usage is discouraged". Since they only serve for a very specific reason and because no new theorems are likely to sprout from them, it was decided to exclude them from analyses. The DFS nodes are listed in table 2.

[4] http://us.metamath.org/mpeuni/weq.html
[5] http://us.metamath.org/mpeuni/wel.html
[6] http://us.metamath.org/mpeuni/wsb.html
[7] http://us.metamath.org/mpegif/dummylink.html

ax-meredith	luk-2	luklem4	ax3
luklem6	ax2	luklem7	luklem8
luklem5	ax1	luk-1	luklem2
luklem3	luklem1	merlem13	merlem11
luk-3	merlem12	merlem10	merlem9
merlem8	merlem7	merlem5	merlem4
merlem6	merlem3	merlem2	merlem1

Table 2: DFS components found starting from ax-meredith.

5.3 Isolated nodes

Within metamath, there are six axioms which are not used by anyone. They are: **ax-7d**, **ax-8d**, **ax-9d1**, **ax-9d2**, **ax-10d**, and **ax-11d**. They were not used to construct the graph.

5.4 Users' mathboxes

Metamath also has the concept of users' mathboxes. It allows contributors to have a workspace to develop their own proofs that later may be incorporated into the official body of metamath. Metamath has an internal mechanism for identifying user's mathboxes. A dummy theorem called mathbox[8] exists to identify them. The **mathbox** theorem can be found within the set.mm file as

```
mathbox $p |- x = x $= vx equid $.
```

All theorems after this line belong to someone's mathbox. It has been observed that many theorems hereafter contain some description similar to "Moved to *<theorem name>* in main set.mm and may be deleted by mathbox owner", meaning that many theorems counted for the main body of **set.mm** file also exists within someone's mathbox. All mathboxes have been ignored. It would be complicated to manually scan through theorems to decide which should be kept and which should be removed. As of the writing of this work, 8 axioms and 8258 theorems exist in different users' mathboxes.

[8] http://us.metamath.org/mpegif/mathbox.html

6 Graph Metrics

Having described metamath, how it has been proposed as a graph, and which peculiarities have been left out, the following sections will describe a series of concepts and analysis that have been performed upon the graph. The main purpose is to investigate how axioms and theorems relate to each other, and if important or essential axioms and theorems can be discovered purely from graph algorithms. The terms "*graph*" and "*network*" are interchangeably used for the following sections. The term "*arc*" is preferred over "*edge*" since the graph is a directed one.

6.1 Node degrees

Based on [5], for a network with n nodes and m arcs, each node i has a set I_i of incoming arcs and a set O_i of outgoing arcs connecting i to other neighbour nodes. For a node i, its indegree is $\delta_i^+ = |I_i|$, its outdegree is $\delta_i^- = |O_i|$, and its total degree is $i = |I_i \cup O_i| \leq \delta_i^+ + \delta_i^-$. It is also known that $max(\delta_i^+, \delta_i^-) \leq \delta_i$. The present work further defines source nodes n_s as having $\delta_s^+ = 0$, meanwhile sink nodes n_t have $\delta_t^- = 0$. Nodes whose $\delta_i = 0$ are isolated nodes and they have been removed as previously explained.

The straightforward global metrics for a graph [5] are its indegree and outdegree mean, which are necessarily equal and are given by

$$\delta^- = \delta^+ = \frac{1}{n}\sum_i \delta_i^+ = \frac{m}{n} \tag{1}$$

The average degree denoted by δ is given as

$$\delta^+ \leq \delta = \frac{1}{n}\sum_i \delta_i \leq \frac{1}{n}\sum_i (\delta_i^+ + \delta_i^-) = 2\delta^+ \tag{2}$$

6.2 Basic distributions

The degree distribution of networks have been the concern of many studies regarding both natural phenomena and man-made structures, such as power grid [6] , solar flares [7] , mobile phone calls [8] , Internet routers and web pages [9–11] . More examples can be found in [12, p. 2].

The degrees of these aforementioned structures tend to follow a power-law distribution. Sometimes, though, structures are better described by a lognormal distribution or other distributions like the double Pareto [8,13]. Both power-law and lognormal distributions have been described in the literature to behave in some similar ways and to have similar generative models [13].

A non-negative random variable X is said to follow a power-law distribution if [13]:

$$P[X > x] \sim cx^{-\alpha}, c > 0, \alpha > 0 \tag{3}$$

Equation 3 means that the probability of a value x to occur is inversely proportional to itself with an appropriate α exponent. A random variable X follows a lognormal distribution if $Y = \ln X$ follows a normal (Gaussian) distribution. The normal distribution Y is given by the density probability function (PDF) [13]:

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(y-\mu)^2/2\sigma^2} \tag{4}$$

The PDF for a lognormal distribution is [13]:

$$f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\ln x-\mu)^2/2\sigma^2} \tag{5}$$

6.3 Network topology

The distributions of δ^+, δ^-, and δ describe the topology of a network. Networks can be classified into two broad groups depending on their distribution: either as a **random network** or as a **scale-free network** [11]. A random network describes graphs whose degrees follow a Poisson distribution, whereas a scale-free network is characterized by a power-law distribution.

In a random network, "a key prediction of random network theory is that, despite the random placement of links, most nodes are assigned approximately the same number of links" [11, p. 34]. It means that most of the nodes in a random network will have a similar degree; the average degree of the network in this case. In contrast, in a scale-free network, "the power-law distribution implies that nodes with few links are abundant, while a small number of nodes have a large number of links" [11]. In other words, scale-free networks usually have hubs with a high connectivity to other nodes. While those hubs are scarce, nodes with a few number of connections are abundant.

One last remark about scale-free networks is that "[...] a power-law distribution does not possess an intrinsic scale, and its average degree does not convey much information about the network structure. The absence of an intrinsic scale in degree in networks with a power-law degree distribution motivates the concept of a scale-free network" [11, p. 36]. This can be seen in figure 3.

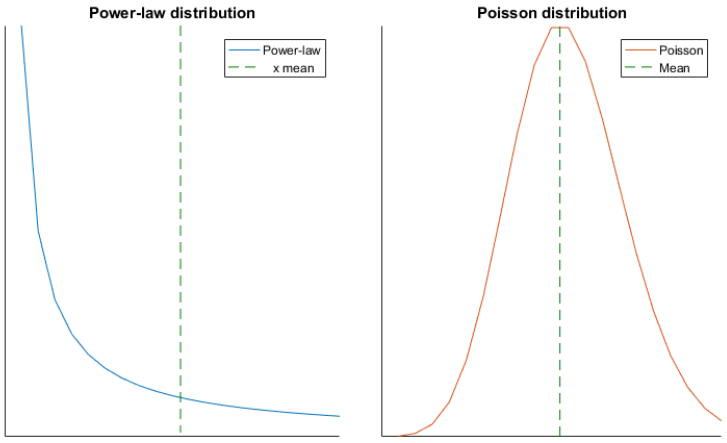


Fig. 3: Visual comparison between a Power-law distribution and a Poisson distribution. While the power-law distribution is not described by the mean value of its data, the mean value of the random dataset describes well the information since it is normally distributed around it.

6.4 Metamath analysis

Analyzing the degrees of metamath network yields interesting results about its structure. Table 3 summarizes statistics for degrees regarding δ^+, δ^- and δ. The values are consistent with equations 1 and 2.

	Mean (μ)	Std. Dev. (σ)	Max Degree	Skewness
Total Degree δ	**36.2708404607**	105.473426224	6093	23.7258374911
Indegree δ^+	**18.1354202303**	23.9225902026	248	2.97762634271
Outdegree δ^-	**18.1354202303**	104.974135948	6091	24.4898820198

Table 3: Statistics for a total of 17538 nodes and 318059 arcs regarding nodes'total degree, indegree and outdegree.

An important observation is the matching μ values for both δ^+ and δ^- which asserts the correctness of the data according to equation 1. These values could not be different since one node's outgoing arcs are the other nodes' incoming arcs in the graph. It is evident that $\delta = \delta^+ + \delta^- = 2\delta^+$ as equation 2 states. Tables 4 and 5 summarize the nodes having the highest values for δ^+ and δ^-. Table 4 also happens to be the top 10 nodes ordered by δ.

Node name	Total degree (δ)	Indegree (δ^+)	Outdegree (δ^-)
syl	6093	2	6091
eqid	3734	2	3732
syl2anc	3499	2	3497
adantr	3285	2	3283
a1i	2716	2	2714
ax-mp	2461	0	2461
adantl	2356	2	2354
syl3anc	2246	2	2244
simpr	1939	2	1937
eqtrd	1826	2	1824

Table 4: Top 10 nodes sorted by outdegree. They are the same top 10 if ordered by total degree.

Node name	Total degree (δ)	Indegree (δ^+)	Outdegree (δ^-)
lgsquadlem1	249	248	1
logtayl	233	230	3
lgsquadlem2	231	230	1
imasdsf1olem	221	220	1
bposlem6	212	211	1
cantnflem1	207	206	1
itg2monolem1	204	203	1
efrlim	204	203	1
pserdvlem2	197	196	1
ostth3	197	196	1

Table 5: Top 10 nodes sorted by indegree.

6.5 Interpretation of individual in-, out-, and total degree.

In order to understand the topology of the network, we analyze its degrees distributions. To clarify the interpretation of the arcs directions, we describe them as following:

The outdegree of nodes is concerned with how much a node helps the rest of network to be built; one node with a high outdegree value indicates that many other theorems rely on it to be proved; perhaps it is some essential mathematical knowledge. It does not play an active role but rather has a passive behaviour on the network construction. Usually, while trying to prove a new theorem, we search for known theorems that can help proving the new one, rather than randomly gathering proven theorems and trying to craft a new one from them. This concept is better illustrated on the next paragraph.

The indegree of nodes is concerned with how many other theorems and axioms are needed to prove one particular theorem; high indegree suggests more complex theorems which rely on many previously proven theorems. The indegree for any node can potentially be reduced since mathematical proofs are not unique and are susceptible of being shortened, therefore using fewer theorems in its proof than actually needed. For such reason, the indegree should not be remarked as one inflexible measure but rather as a more malleable one as the network evolves guided by human knowledge. One important remark here is the possibility of randomly choosing one theorem and reduce its proof length, consequently reducing its indegree; meanwhile, it is not possible to peer at one random theorem or axiom and deliberately reduce its outdegree. It would be necessary to look at all theorems which depend on it and try to shorten their proofs regarding this particular axiom or theorem in order to purposefully and hopefully reduce the randomly chosen outdegree.

Finally, about the total degrees, it is difficult to give one simple description about them. From table 4 it is possible to see that nodes with high total degree are governed solely by their outdegree. Similar conclusions can be drawn from table 5. All source nodes have their total degree totally accounted by their outdegree ($n_s \to \delta_s^- = \delta_s$). Conversely, all sink nodes have their total degree entirely accounted by their indegree ($n_t \to \delta_t^+ = \delta_t$). What, then, could be inferred from a node with, say, one-half of its total degree being composed of its indegree and the other half of its outdegree ($\frac{1}{2}\delta_i = \delta_i^+ = \delta_i^-$)? And how frequently would that happen? Figure 4 shows the proportion of node degrees for all nodes based on their **in-** and **out-** degree. The rightmost values represent all sink nodes; meanwhile, the leftmost values represent all source nodes (axioms). It is clear that this proportion varies greatly.

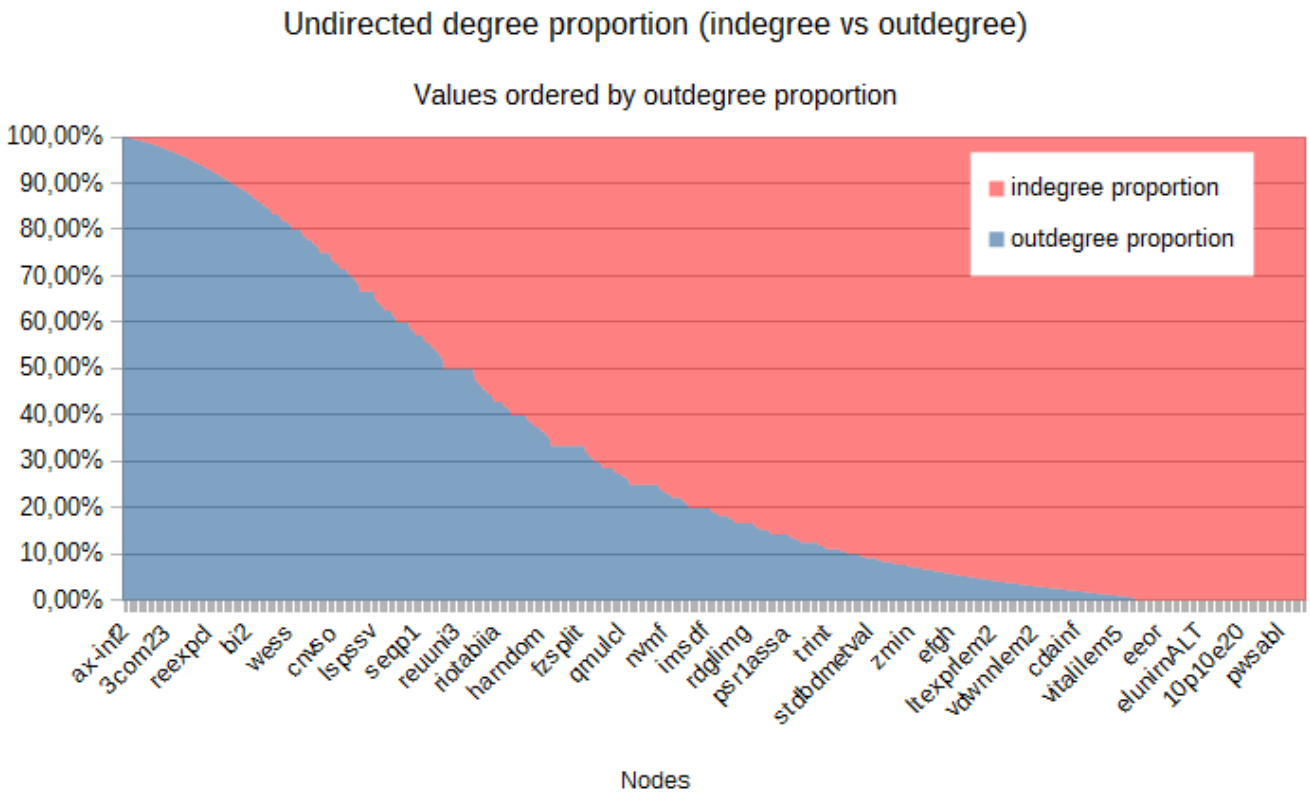


Fig. 4: Proportion of **in-** and **out-** degrees for all nodes. The values are ordered according to the outdegree proportion to create a smooth curve.

6.6 Distribution analyses

Figure 5 shows the results for each degree distribution. One simple empirical way of testing a distribution for power-law behaviour is to use a log-log plot, where the logarithmic scale is used for both axes, and see if the data appears as a straight line [12, p. 1].
The distribution in figure 5 suggests a power-law for the outdegree. The total degree seems to follow a log-normal since its shape resembles a parabola [8]. Not so clear is the indegree distribution which stands in the middle of the two other distributions. Notice that in order to produce the log-log plot, zero-values are not considered, thus the frequency for nil in- and out- degrees are not represented in the figure, but their values are the number of sources (65) and sinks (2529), respectively. The right-hand end of the figure is noisy due to the size of the graph which only contains a little more than 17000 nodes. When it happens, one explanation is that "the power-law distribution dwindles in this region, meaning that each bin only has a few samples in it, if any. So the fractional fluctuations in the bin counts are large and this appears as a noisy curve on the plot." [12].

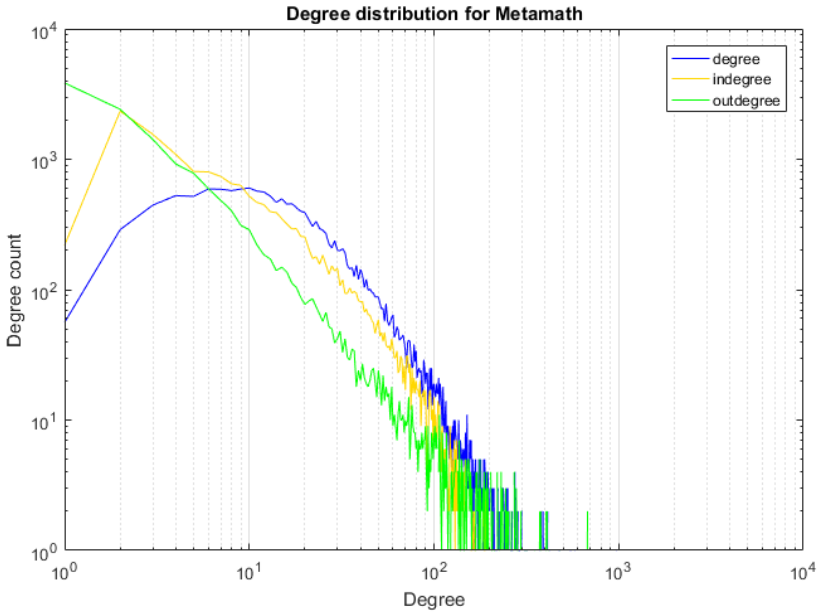


Fig. 5: All three degree distribution in a log-log plot. The outdegree seems like the best candidate for a power-law distribution. The total degree distribution seems to be better fit into a lognormal distribution. The indegree stands in between the power law or lognormal distributions. Integer bins have been used to group data.

Total degree analysis The total degree indeed follows a log-normal distribution. Figure 6 shows the frequency histogram fitted by the log-normal distribution. Figure 7 shows the normal distribution for the log of the variable , which confirms the quality of the log-normal fit.

Indegree analysis The same quality could not be achieved while fitting the indegree data. The histogram in figure 8 loses precision for $\delta^+ < 14$ and it peaks at two points: $\delta^+ = 2$ and $\delta^+ = 3$; these two peaks extrapolate the fitting. Figure 9 shows that the log of the variable δ^+ does not fit the normal distribution as well as in figure 7. As previously explained, since indegree represents elements that are necessary to prove a theorem, and since proofs can be shortened, it seems possible that throughout the network evolution the distribution will change to a more definitive and conclusive shape. The peaks in figure 8 could lead someone to wonder if a power-law would fit some range of the indegree distribution. In order to test it, two power-law fits have been tried, yet without success. In figure 10 we tried to exclude the first point of the distribution and afterwards we tried ignoring more points to fit the tail of the data. Neither approaches yielded significant outcomes, therefore the power-law for indegrees has been rejected.

Outdegree analysis It has been observed that δ^- visually follows with better accuracy the power-law distribution if we ignore sink nodes and unitary outdegree nodes ($\delta^- = 1$). It is important to understand that identifying power-laws can be tricky. Usually, the strategy is to plot the histogram and see if it appears as a straight line on a log-log plot [12]. Figure 11 shows how data looks more like a straight line, albeit it is hard to tell how the line should go through the right side. As previously said, this occurs because of the data scarcity for large values. Striving for accuracy, we apply the proposed method found on [12] which uses the cumulative probability distribution function. Instead of using the histogram of the data, the probability $P(x \geq \delta^-)$ is rather used.

$$P(X) = \int_x^\infty p(x')dx' \tag{6}$$

Even though the representation of data is no longer as simple as its histogram, if the data follows a power-law $p(x) = Cx^{-\alpha}$, then

$$P(X) = \int_x^\infty x'^{-\alpha}dx' = \frac{C}{\alpha - 1}x^{-(\alpha-1)} \tag{7}$$

Therefore, the cumulative distribution also follows a power-law, but with an exponent $\alpha - 1$ [12]. In figure 12 we plot the complementary cumulative probability of δ^- being greater than a certain value. It naturally decreases for large values. We observe that for $\delta^- \in [4, 99]$ a power-law holds true with exponent $\alpha - 1 = 0.7745$. The nodes contained within that range sum up to 6772 nodes, thus accounting for 38.61% of the total network. After $\delta^- \geq 100$, the data curves and deviates from the straight line.

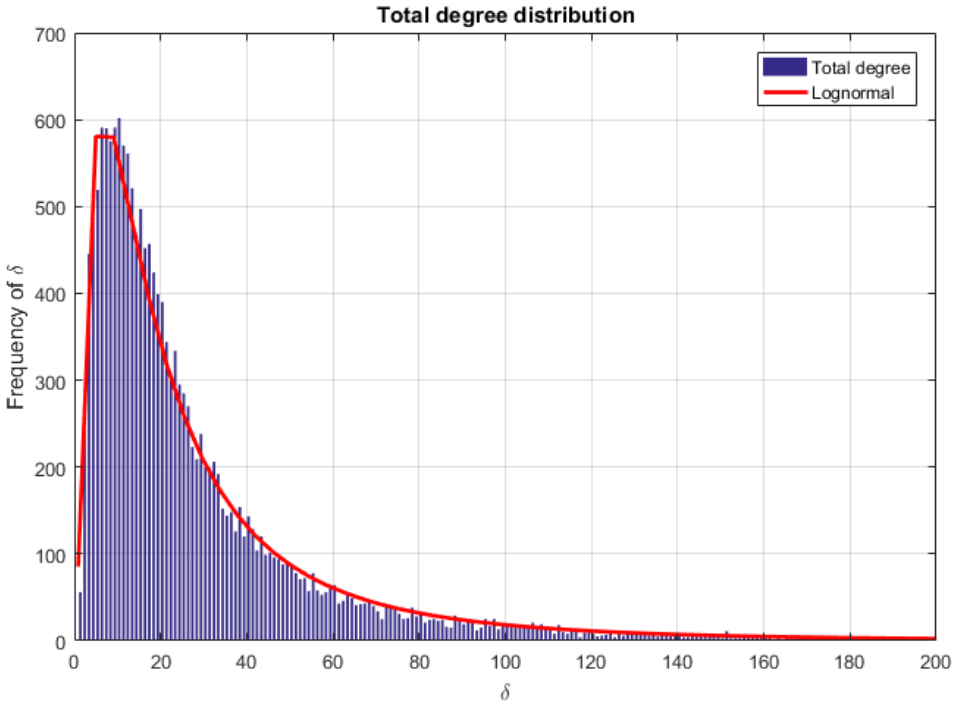


Fig. 6: Log-normal distribution fit for the total degree of the graph nodes

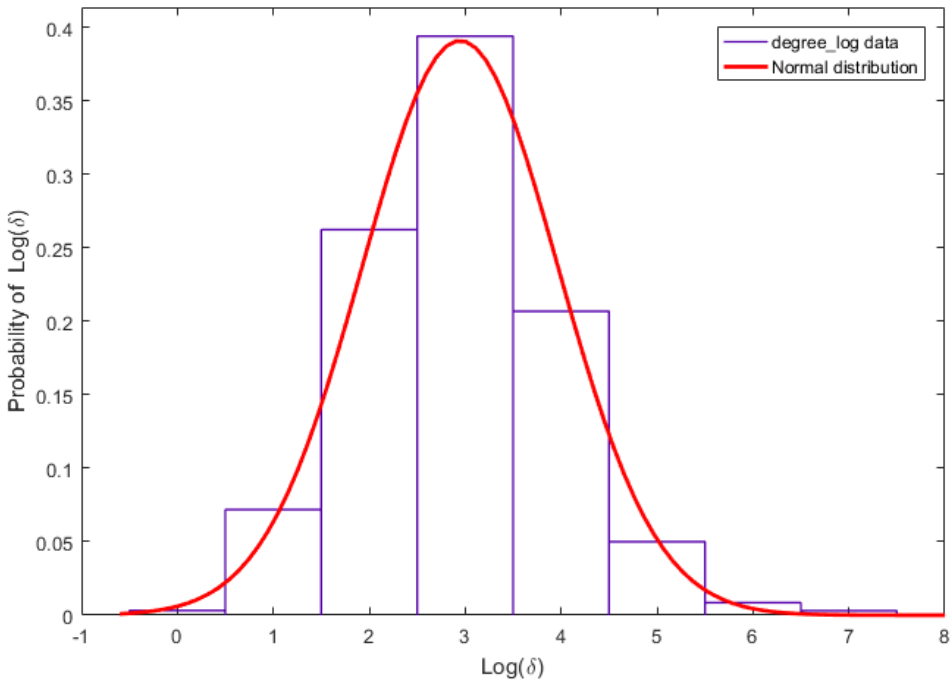


Fig. 7: Log of δ using integer bins fitted by a normal distribution. The log of the variable follows a normal distribution.

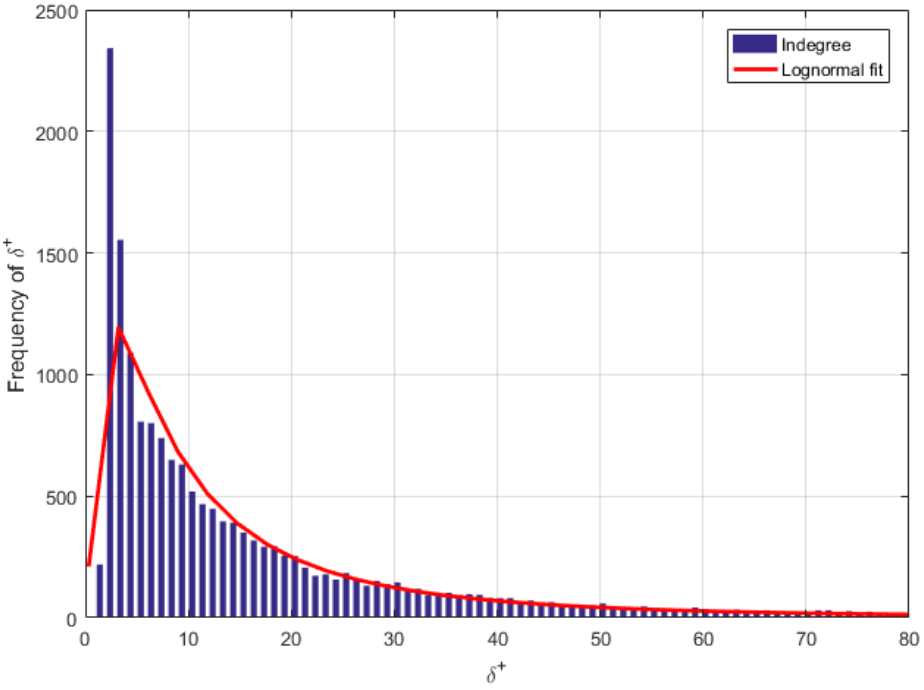


Fig. 8: The indegree distribution does not follow a lognormal distribution properly. Zero values are not considered for the lognormal fitting.

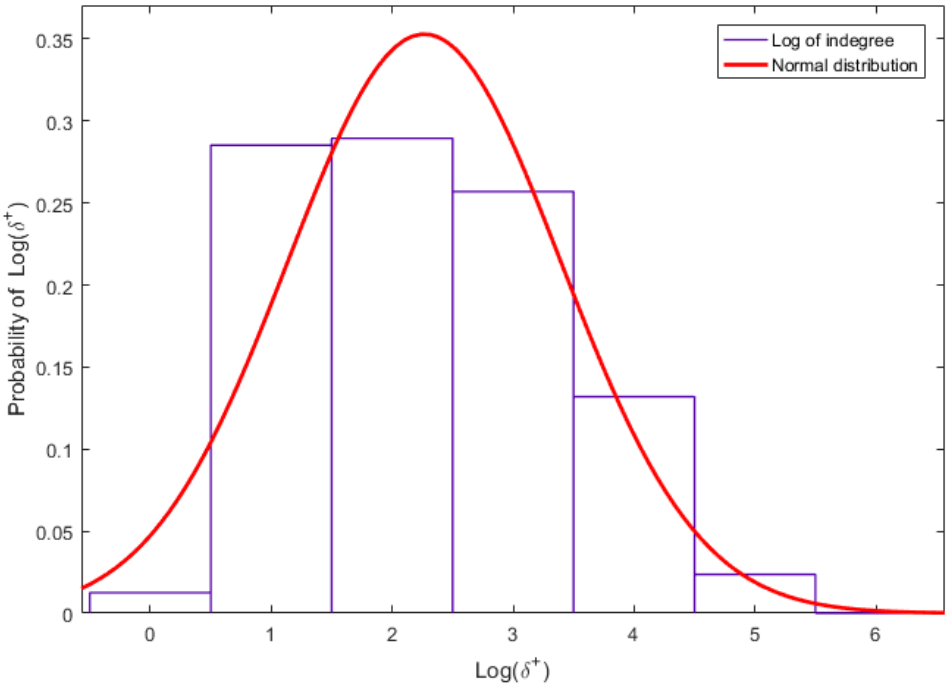


Fig. 9: Log of δ^+ using integer bins fitted by a normal distribution. The fit is unsatisfactory and inconclusive about the distribution.

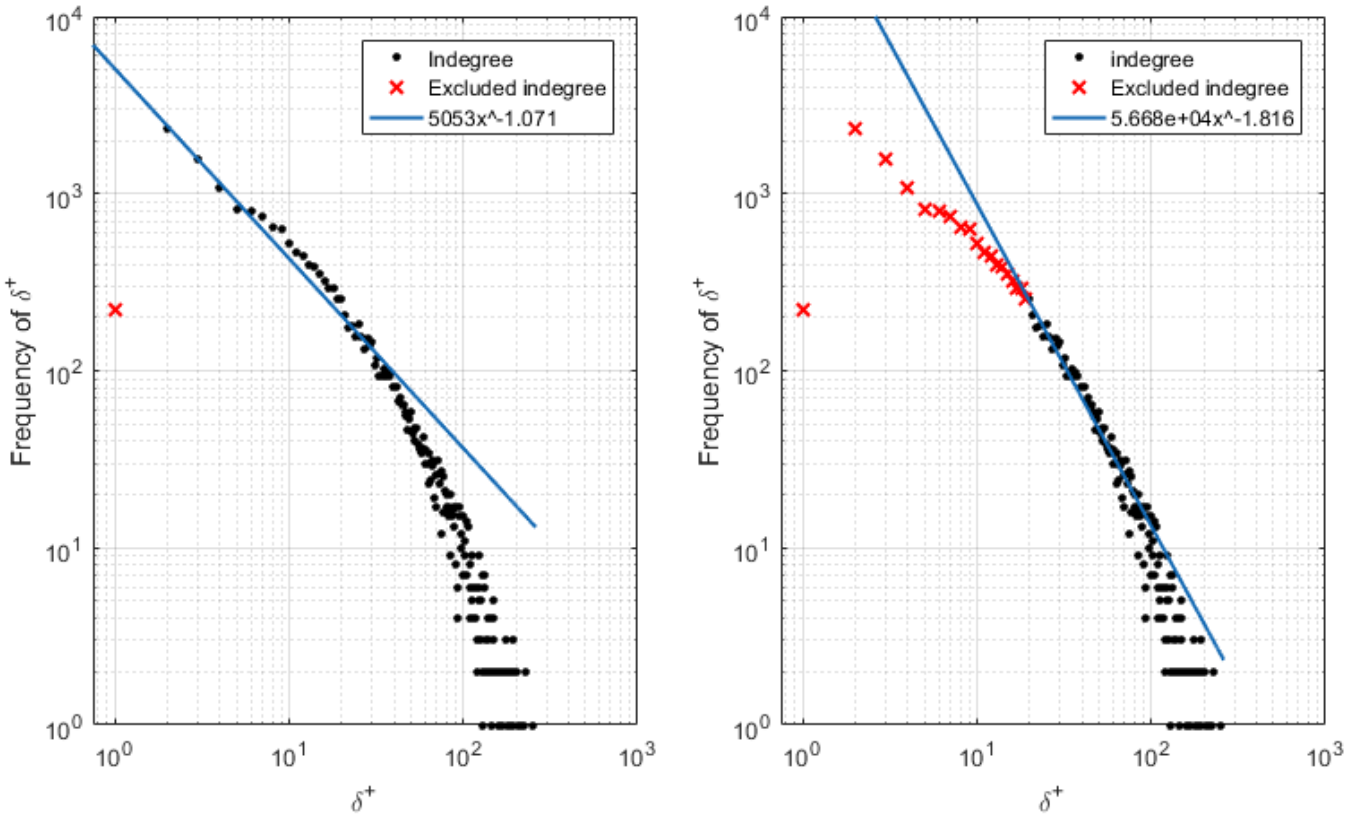


Fig. 10: Trying to fit δ^+ with a power law distribution. The points do not seem to follow any power law for the most general case.

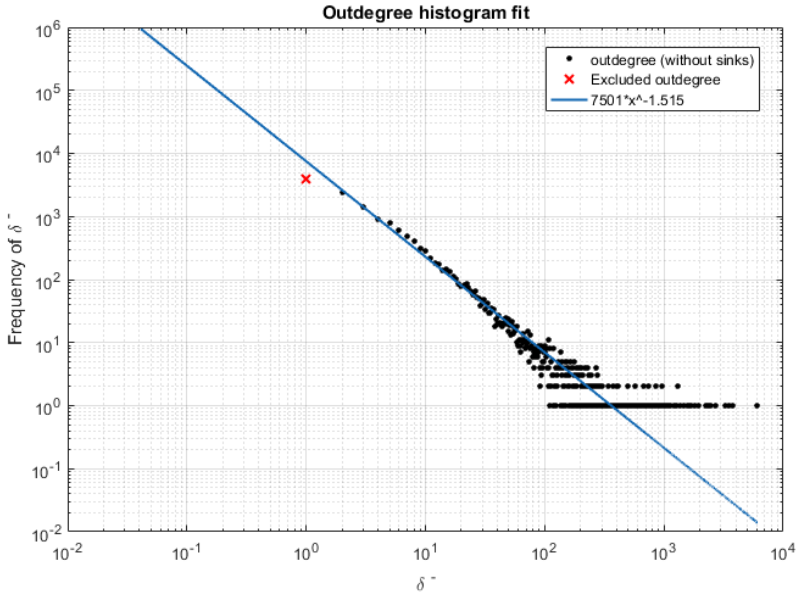


Fig. 11: Outdegree fitted by a power law. The first point of the outdegree distribution has been excluded.

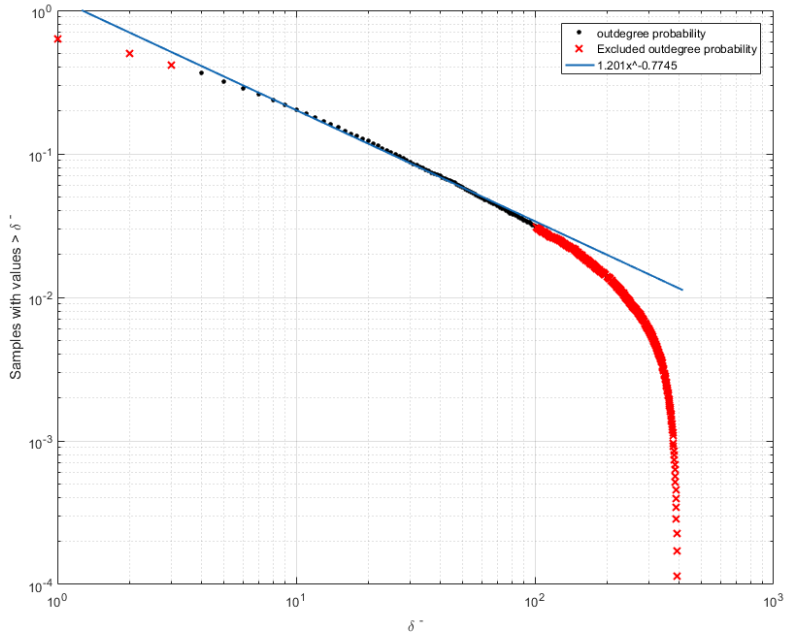


Fig. 12: Portion fitted by power-law with slightly concave curvature due to finite size effect.

7 Source decomposition

Given a Directed Acyclic Graph (DAG), its topological sorting reveals the precedence of connected elements [14]. In our case, it shows the dependency of latter theorems upon previous theorems and axioms. For every arc (u, v), it indicates that u must occur before v can occur [14]. Inasmuch as the graph has an acyclic orientation, we do a source decomposition of the graph. This decomposition is similar to the sink decomposition described in [15, p. 565], yet the operations are performed upon *source* nodes rather than on *sink* nodes.

In the source decomposition, for all source nodes, group them together into a layer L_0. Then, all their direct neighbours are grouped into a layer L_1. The process continues until the last set of possible nodes have been assigned into a layer $L_{\lambda-1}$. A node is in a layer $L_k, 0 \leq k \leq_{\lambda-1}$ if and only if the longest directed path from it to a source has k edges. Two layers L_k and L_l are said to be successive if and only if $|k-l| = 1$; contrariwise, they are said to be nonsuccessive. Every node in L_k has at least one neighbour in $L_{k-1}, 1 \leq k \leq \lambda - 1$. λ is the length of the decomposition.

The same process can be performed by selecting first the sink nodes, likewise done in [15]. Given the structure of the herein described graph, we find the decomposition by sources a more natural approach in order to study the structure of dependency among nodes.

This decomposition shows the largest path between two nodes. One idea behind this decomposition process is to identify potential bottleneck layers within the network. That is, layers with only a few nodes within it. For example, traversing the graph from a starting axiom on layer L_0 to some node in layer L_k, it has to have at least one path which goes through all the intermediary layers $L_j, j = 1, ..., k-1$, because each layer represents one step of dependency among nodes. A layer $L_k, 1 \leq k \leq \lambda - 2$, containing only a few nodes would indicate that all the following layers $k + 1$ depend on it to be proven. Notice that even if a different set of theorems could make for a new demonstration, it would still be constricted by the narrow layer L_k unless it could be entirely proven from theorems appearing only before layer L_k, yet we find it unlikely for most cases.

Figures 13 and 14 show the size of each layer after decomposing the graph starting from source nodes. As it can be observed, a valley is formed around the fifty-seventh layer, which only contains three theorems, and thereafter the layers grow in width again. The theorems found are **sb3**[9], **sb4**[10], and **sb56**[11]. Since it is difficult to see details on figures 13 and 14, table 6 lists the 20 shortest layers found during the source decomposition.

[9] http://us.metamath.org/mpeuni/sb3.html
[10] http://us.metamath.org/mpeuni/sb4.html
[11] http://us.metamath.org/mpeuni/sb56.html

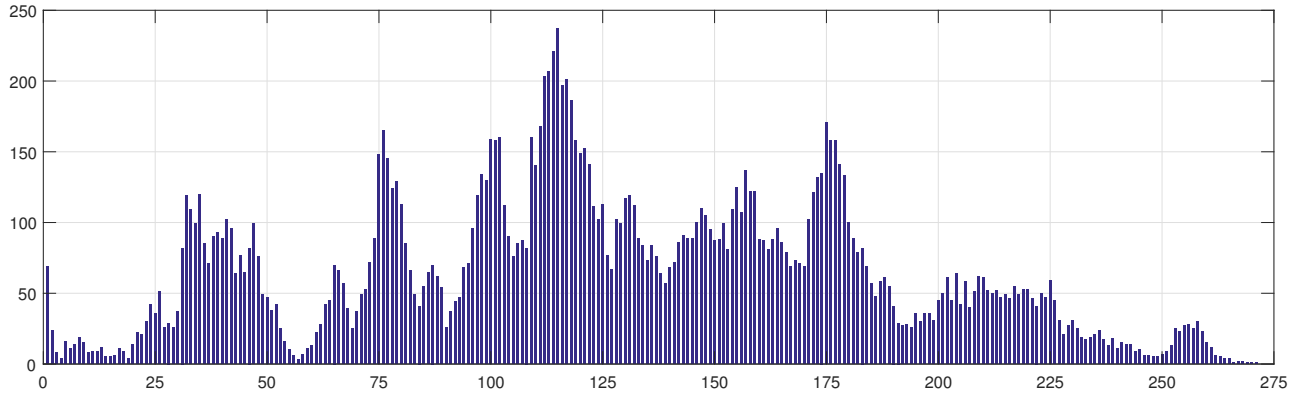


Fig. 13: Barplot of the width of each layer after a decomposition by sources.

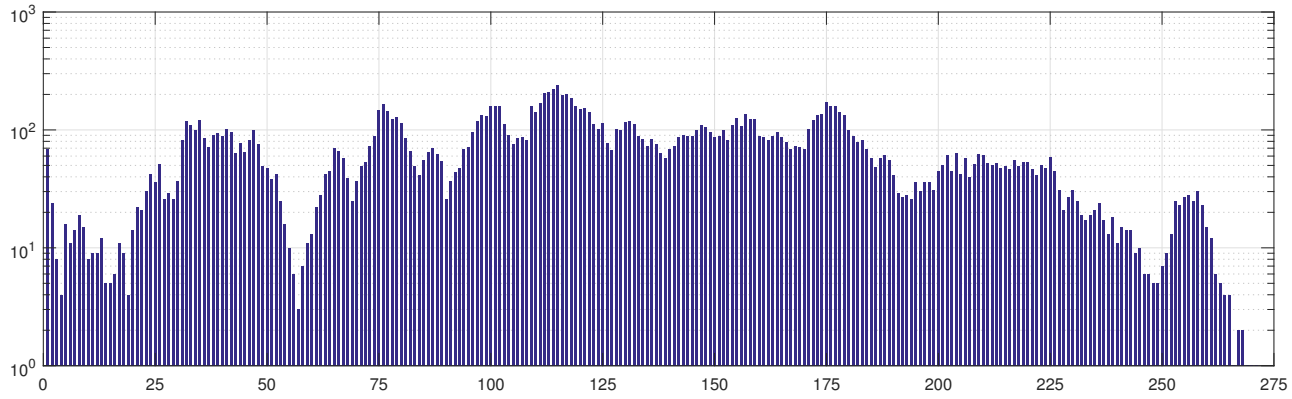


Fig. 14: Barplot of the width of each layer after a decomposition by sources using a log scale on the y-axis for a better visualization.

Layer	Layer size	Layer components
4	4	syl mpi id 2alimi
14	5	com4l com35 con2i tbwsyl re1luk1
15	5	com4t com14 com5l nsyl notnot1
16	6	com4r com24 com52l notnoti con1d con3i
19	4	con1i con4i ja pm2.01d
56	6	equs5 ax11v ceqsex6v ceqsex8v clel2 clel4
57	3	sb3 sb4 sb56
246	6	itgaddlem2 itgmulc2lem1 bddibl efif10 eff10lem basellem4
247	6	itgadd cniccibl iblulm efifo eff10 basellem5
248	5	itgsub itgfsum logrn basellem8 circgrp
249	5	itgmulc2lem2 ellogrn dflog2 eff102 basellem9
263	5	dvlog2 efopn dvatan chebbnd2 ostth2
264	4	logtayl atancn efrlim ostth
265	4	logtaylsum logtayl2 atantayl dfef2
266	1	atantayl2
267	2	atantayl3 leibpi
268	2	leibpisum log2cnv
269	1	log2tlbnd
270	1	log2ub
271	1	birthday

Table 6: Twenty smaller layers of a graph decomposition applied to source nodes ordered by layer number.

Besides the last six layers that mostly contain only one or two elements, which is expected to happen since the network is still growing up every day, the narrowest layer is the fifty-seventh which was already mentioned. Narrow layers that appear at the beginning of the decomposition could indicate that they are more important and relevant because it means that more nodes depend on them to be proven. For any theorem to be proved, it must be reachable from axioms (mathematically saying, the converse must occur, but as in our case arcs go from axioms to theorems, it is easier to described it like so).

We remark upon the last layer, the two-hundred-seventy-first one, which only has one element on it named as "**birthday**"[12]. This theorem refers to the well-known birthday problem which asks: "What is the least number of people required to assure that the probability that two or more of them have the same birthday exceeds $\frac{1}{2}$?" [16]. The fact that it appears in the last layer implies that all the other layers are necessary for it to be proven, thus revealing its complexity despite its intuitive formulation which one would not find difficult to understand. λ is the length of the path that goes from some axiom up to **birthday** theorem. Despite all layers being mandatory, it does not mean that all nodes on them are in the same way necessary to prove **birthday**, but only the nodes which lie on the *paths* that go through those $\lambda - 1$ layers starting from L_0.

[12] http://us.metamath.org/mpegif/birthday.html

8 Proof steps distribution

Another exploratory approach to metamath ecosystem regards how many steps a theorem needs to be proved. Metamath has on its website the steps necessary to prove theorems laid out in a human-friendly way. Theorem **2p2e4**[13] is a good example which shows that $2 + 2 = 4$ needs, in the time being, 10 steps to be proved. Those steps are the relevant ones in a human perspective of practicality. For metamath software, though, those steps are longer because they include all the necessary substitutions of definitions which are necessary to verify that all well-formed formulas are correct. What metamath does is not explicitly displaying such steps to the layperson. "You can tell the program what level of detail of the proof you want to look at. You may want to look at just the logical inference steps that correspond to ordinary formal proof steps, or you may want to see the fine-grained steps that prove that an expression is a term." [3, p. 37].

Although our graph has been built to show how theorems relate to other theorems on their proof, the graph in itself does not show *how* the theorems were used to construct a proof because both the graph relationships are not ordered and also some theorems may be used twice or more during a proof construction[14]. In order to obtain all the number of steps, it was necessary to crawl all theorem pages and manually extract that information from their HTML pages. It is only necessary to find the first column of the last row from the HTML table.

One motivating reason to explore this aspect of metamath is that "The length of a proof can, to a certain extent, be considered an objective measure of its 'beauty,' since shorter proofs are usually considered more elegant. In the set theory database set.mm provided with Metamath, one goal was to make all proofs as short as possible." [3, p. 16]. Mathematicians like to be short and concise.

Figure 15 shows a scatter plot of the proofs steps. The majority of them are concentrated below the 100 steps mark. Theorems with a longer proof steps are not as common, especially after the 300 steps mark. Table 7 lists the 10 longest proofs in metamath. Both power-law and lognormal fits have been tried upon the data, yet without a conclusive result. On figure 16 we try to fit the data into a lognormal distribution, which resembles figure 8 from section 6, where a portion of the data extrapolates the fit. A lognormal distribution is not conclusive for the data. Figure 16 reveals that the most common are theorems with three, four, or five steps to be proved, extrapolating the lognormal distribution fitting.

[13] http://us.metamath.org/mpeuni/2p2e4.html
[14] As one example, theorem **halfpm6th** uses theorem **3eqtr2i** on its proof three times.

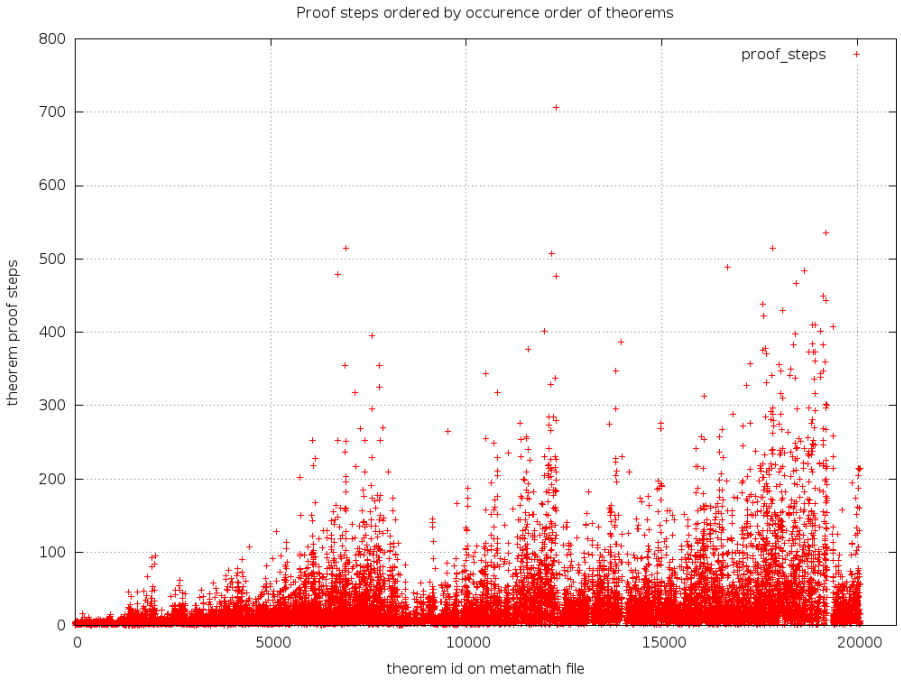


Fig. 15: Scatter plot of theorems and the number of steps each theorem needs to be proved.

Theorem	**Proof steps**
2503lem2	706
lgsquadlem1	535
cantnflem1	515
itg2monolem1	514
vdwlem6	507
imasdsf1olem	489
logtayl	484
marypha1lem	479
4001lem1	476
pserdvlem2	467

Table 7: The 10 largest proofs of metamath

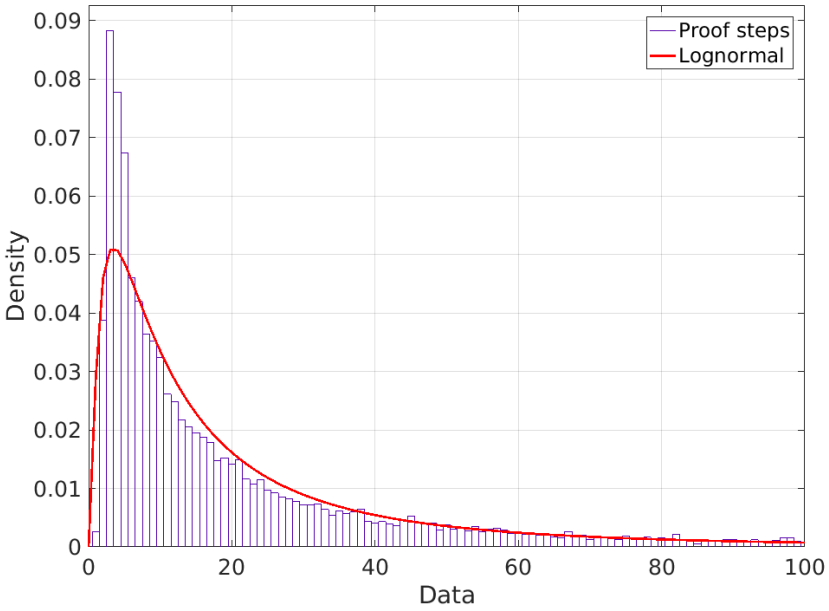


Fig. 16: PDF of proof steps approximated by a lognormal distribution, however it is not a good approximation.

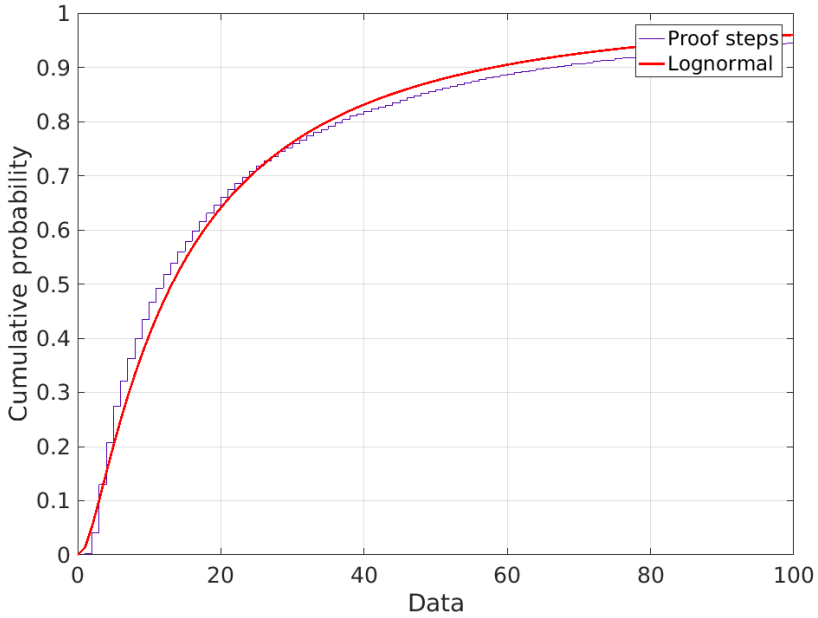


Fig. 17: CDF of proof steps approximated by a lognormal distribution.

9 Reachability of nodes

Another exploratory approach in order to understand the Metamath graph structure is to quantify how many nodes can be reached from a source node. Given a node x, a $DFS(x)$ yields a tree with root in x. All the paths in that tree leading from x to all sink nodes show the dependencies of the nodes in the tree in relation to x. The size of the tree, i.e, the number of nodes in the tree, shows how many nodes, directly or not, depend on x to be proved. We found an interesting linear relationship between the number of sinks and the size of the DFS tree. Two scenarios are explored: a DFS starting from axioms only and a DFS starting from all nodes. Of those two scenarios, in each one we calculate both the number of sinks and the total number of nodes reached by each DFS.

9.1 Axioms to sinks and axioms to everyone

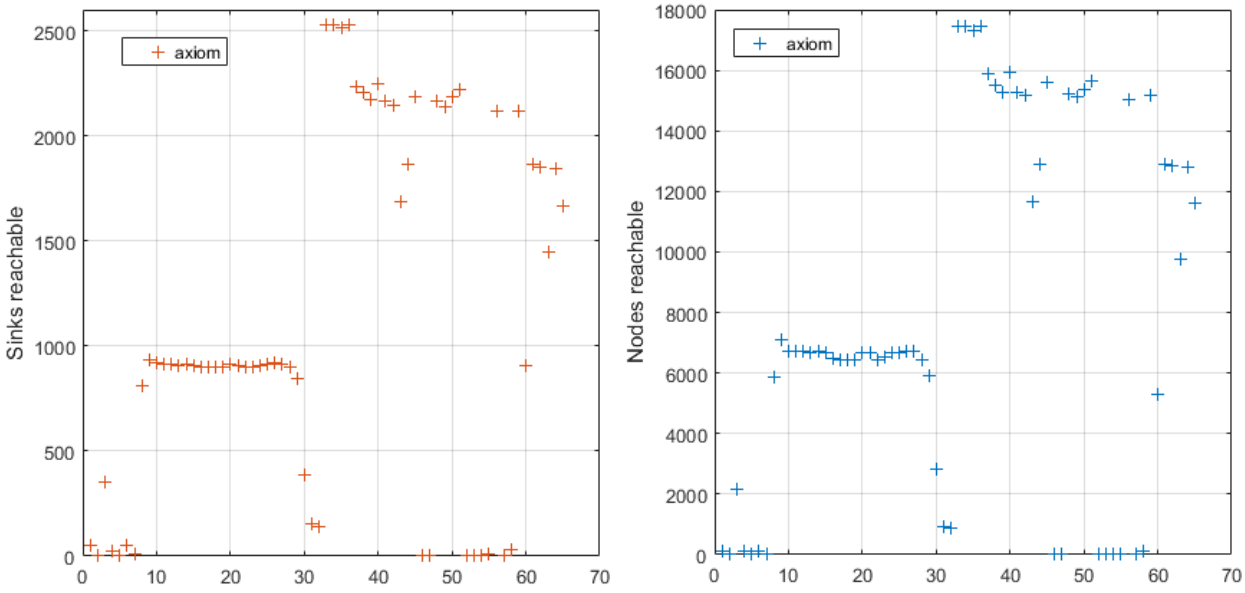


Fig. 18: Scatter plot for the counting of nodes that can be reached from a DFS starting from axioms. The left figure shows the number of sinks reachable, meanwhile the right one shows the size of the DFS trees (quantity of reached nodes in general). Every point on the scatter plot represents one axiom.

From figure 18 it is remarkable the similarity between the two images despite the different scales on the y-axis. It suggests a linear relationship between the data since one figure can be seen as a scaled version of the other by a constant factor. To confirm it, a scatter plot is created where for each axiom, its sink reach count value is used for the x-axis value and its general reach count value (DFS tree size) as the y-axis value. Figure 19 confirms the suspicions, where each point on the plot represents one axiom with its two reach values: reachable sinks and reachable nodes. The points are fit by the equation $y = 7.0112x + 43.847$, meaning that for each number of sinks that one axiom can reach, in average, it can also reach seven times that number of nodes in general.

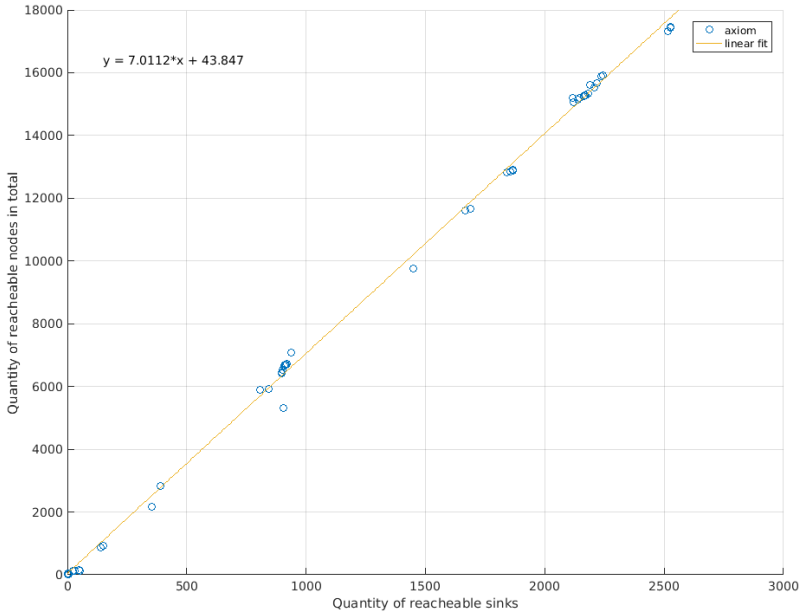


Fig. 19: Linear relationship between the quantity of sinks and general nodes that can be reached by a DFS starting from a given axiom (source node).

9.2 Everyone to sinks and everyone to everyone

The same analysis is performed for the other scenario where the DFS is performed for all nodes. Similar to figure 18, figure 20 suggests that the data has a linear relationship since the plots are similar by some scale factor. In the same manner, we confront these two information to seek a linear relationship between the data.

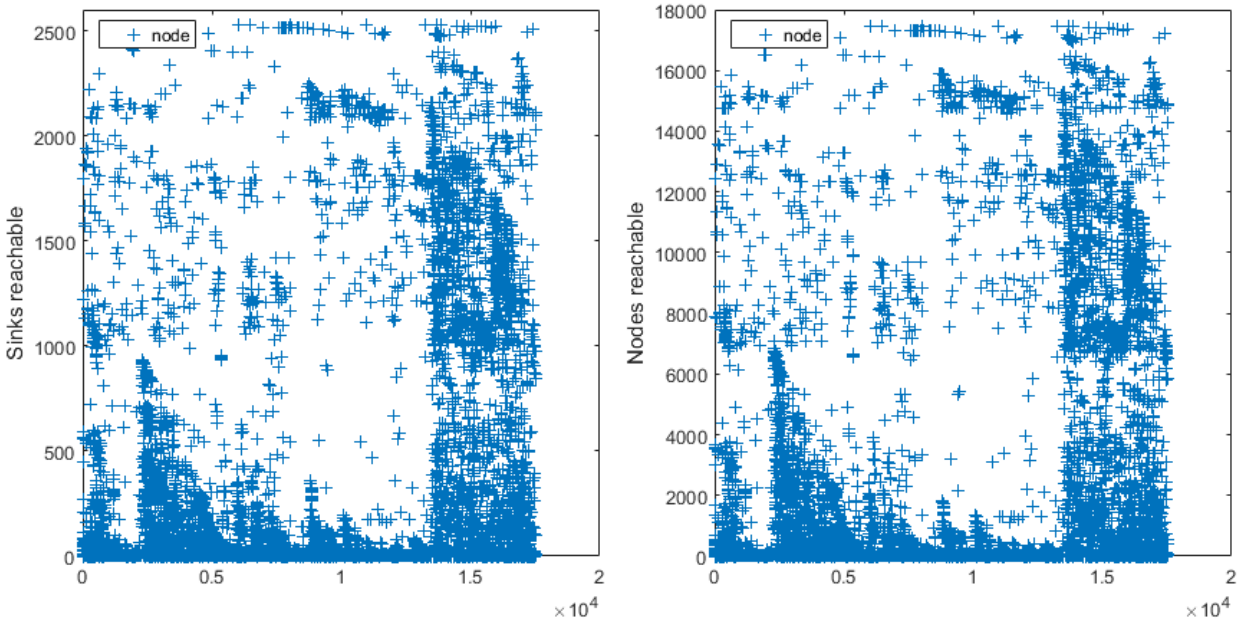


Fig. 20: Scatter plot for the counting of nodes that can be reached from a DFS starting from every node. The left figure shows the number of sinks reachable, meanwhile the right one shows the size of the DFS trees (quantity of reached nodes in general). Every point on the scatter plot represents one node of the graph.

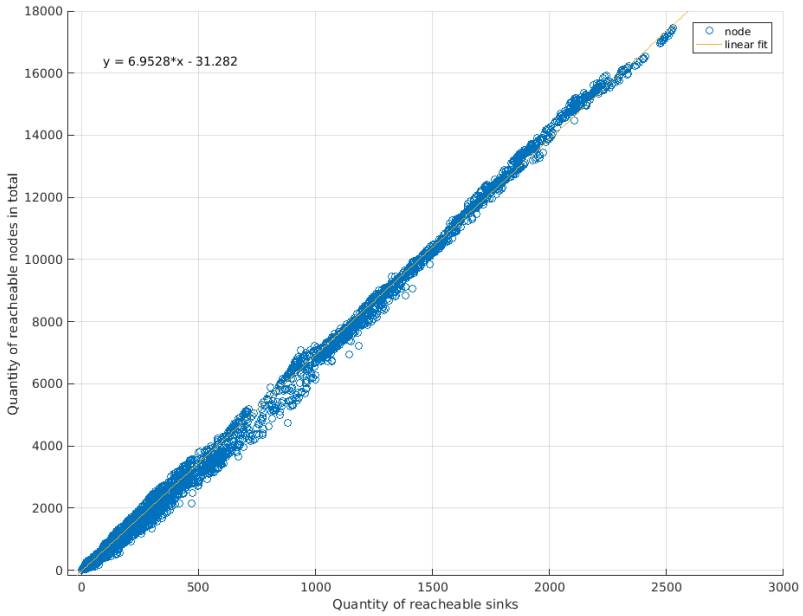


Fig. 21: Linear relationship between the quantity of sinks and general nodes that can be reached starting from a given node (axiom or theorem).

The data on figure 21 is fitted by the linear equation $y = 6.9528x - 31.282$, having the same interpretation of figure 20: For each sink one node can reach, in average, seven times that quantity of nodes in general can be reached as well.

9.3 Interpretation of data

It is difficult of say exactly what that linear relationship can mean. Consider figure 22. Figures 22a and 22b are almost identical, but with one simple node addition, the values for DFS-tree size and sinks do not convey significant information about their structure. Figure 22a has a ratio of $0.75 = \frac{3}{4}$ between DFS-tree size and sinks, meanwhile figure 22b has a ratio of $0.2 = \frac{1}{5}$. We imagine that for each DFS-tree, a new theorem could appear in the network making usage of the sinks, completely changing this proportion, even though the structure *per se* has not changed so much. Another example is between figures 22b and 22c. Although having the same values for DFS-tree size and sinks, their structures are completely different, thus the general relationship between these two values carry little to no information about the general DFS-tree structure. Nonetheless, is it quite surprising that some relationship between these values exists.

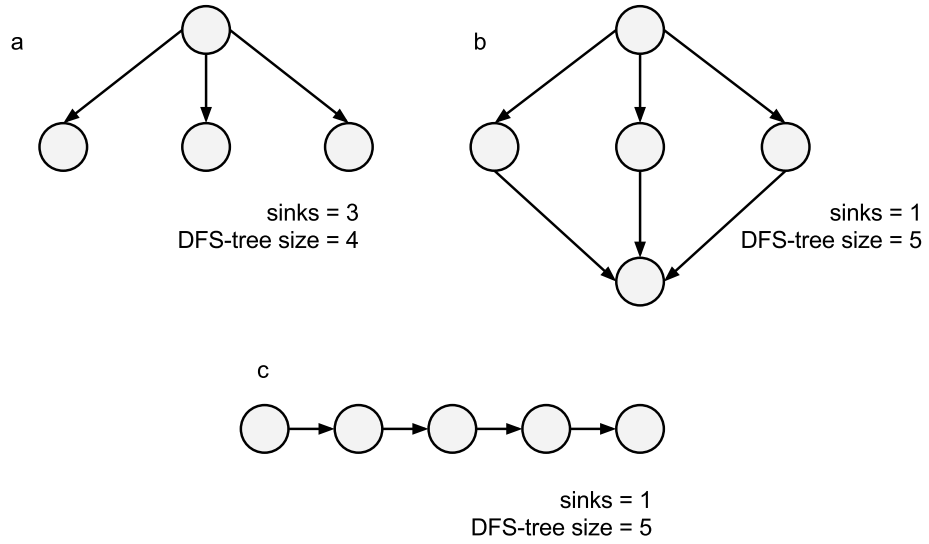


Fig. 22: Hypothetical graph trees. The difficulty to give a proper meaning for the relationship between the DFS-tree size and the number of sinks of the tree arises because totally different trees can have the same values, or small changes in structure can create a significant difference between these two measurements.

10 Max Flow approach

Our final approach towards Metamath is to analyze its axioms trying to realize which ones are the most relevant based on max flow.

A directed graph can be interpreted as a "flow network" where the rate at which information moves through a communication network from a source to a sink can be modeled [14]. Each arc indicates whither it is possible to move throughout the network. The max flow problem computes the greatest rate at which content can be delivered from a source node to a sink node. It is also a classical combinatorial problem [17].
From [14] we have the definitions: A **flow network** $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative **capacity** $c(u, v) \geq 0$. If we have that $(u, v) \in E$, then it is required that $(v, u) \notin E$. If $(u, v) \notin E$, then for convenience we define $c(u, v) = 0$. Also, self-loops are not allowed, thus $c(u, u) = 0$. Two distinguishable nodes are: a ***source*** s and a ***sink*** t. A ***flow*** in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfy both constraints:

(a) **Capacity constraint** $\forall u, v \in V, 0 \leq f(u, v) \leq c(u, v)$
(b) **Flow conservation** $\forall u \in V - s, t, \sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$

Different from the reachability of nodes of section 9, which sees how many sinks and nodes can be reached by a DFS starting from some other node, this study tries to identify important axioms.

Using the herein described graph would lead us to a problem, though: a node could have multiple flows flowing through it. Considering that each arc initially has the same capacity, that is, $c(e) = 1, \forall e \in E$, a node n could contribute twice or thrice to a flow within the network since the flow within n would be restricted only by $min(\delta_n^+, \delta_n^-)$. Our approach wants to address nodes individually. To assure that each node will contribute only once to the flow, the following technique has been used to give constraints to *nodes* rather than *arcs*: every node n is halved into n' and n''. These two halves are connected such that $c(n', n'') = 1$ and $O_{n'} = I_{n''}$. All the original reaching arcs reach the first half ($I_{n'} = I_n$), whereas all the original leaving arcs depart from the second half ($O_{n''} = O_n$). We call the arcs connecting the two halves *internal arcs* whose form is $e_i = (n', n'')$. External arcs have the form $e_e = (u'', v')$. Before running the max flow algorithm, we define the arcs constraints such that $c(e_i) = 1$ and $c(e_e) = 2$; that is, *external* arcs have twice the capacity of *internal* arcs which only contain an unitary capacity. Since all internal arcs have less flow capacity than external ones, the flows will get constricted within nodes rather than on external arcs. This graph modification technique is illustrated in figure 23.

The software HI-PR[15] [17] has been used to calculate the max flow. HI-PR is an efficient implementation of the so called push-relabel method for solving this sort of problem.

[15] http://www.avglab.com/andrew/soft.html

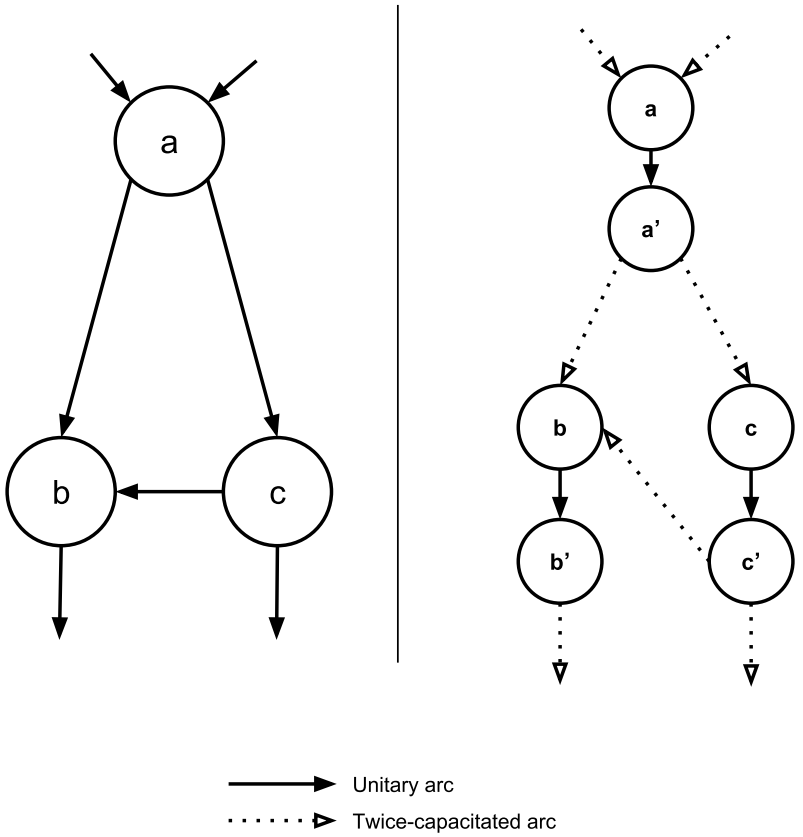


Fig. 23: Graph technique to impose flow constrictions onto nodes.

The reason for that modification is that running the max flow problem on that instance of graph shows how many distinct dependency paths exist between nodes. Let $x \in V$ be any node, axiom or theorem, and let $DFS(x)$ be the descendancy of that node throughout the graph. Given some node $y \in DFS(x)$, either x is a fundamental axiom in order to prove y, or x is a theorem which needs to be proved before y can be proved. All directed paths between x and y ($x \rightsquigarrow y$) are contained within $DFS(x)$. Each of these paths represents a different dependency to prove the validity of y based on the validity of x. All these paths are never disjoint because x and y always participate on them. If both x and y are not taken into account, then it is possible for some of the directed paths between them to never share any node in common, thus being disjoint. As $c(e_i) < c(e_e), \forall e \in E$, we ensure that the flow will always be throttled on some node rather than on some other *external* arc. The number of disjoint paths of $x \rightsquigarrow y$ is given by $f(x, y)$. Each of these paths can be seen as one independent way of explaining why x is essential for y. A unitary flow between nodes means that only one path exists between them.

Considering the 65 source nodes (all the axioms nodes) and the 2529 sink nodes, for each pair source-sink the max flow has been calculated. We have found that the majority of them has only a unitary flow, as figure 24 shows. The data neither follows a lognormal nor does it follow a power-law distribution. Nonetheless, it is interesting to see how many unique paths exist for many of the axiom-sink pairs.

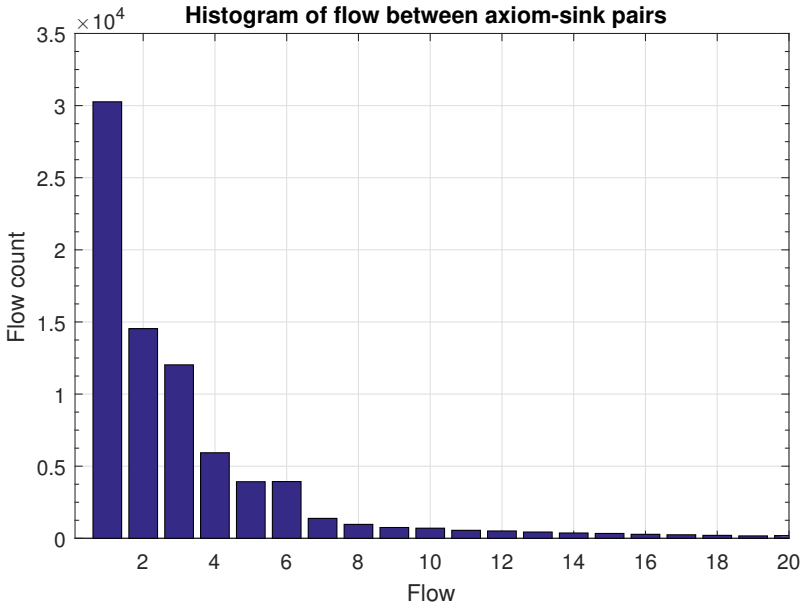


Fig. 24: The max flow has been calculated for all pairs of axiom-sink of Metamath graph. The image shows the histogram of flows.

10.1 Classes of axioms

All max flow values for each axiom-sink pair have been summarized and we call them as *accumulated flow* for a given axiom. Tables 9 and 10 resume the accumulated flows for all axioms. Also, each axiom has been categorized according to information found on metamath website. The categories are briefly explained in table 8 if the reader is further interested to know more about them. Figure 25 gives a visual clue for the tabular data on tables 9 and 10.

Category	Explanation
??	There are two axioms that we were unsure on how to classify them: the axiom **ax-cc** (Countable Choice) and the axiom **ax-dc** (Dependent Choice). We suspect that they are classifiable under the ZFC category, albeit their total flow is insignificant when compared to the other categories and then is does not affect the overall results.
Prop. calc.	Axioms of propositional calculus. They are axioms of logic and a prerequisite for predicate calculus [3, p. 56]. They are constructed with the connectives $\neg$(not), $\wedge$(and), $\vee$ (or), $\rightarrow$ (implies), and $\leftrightarrow$ (iff) by using certain closed rules of formation [18, p. 14], meaning that any new formed expression by appropriate rules do not strengthen the language.
Pred. calc.	Axioms of predicate calculus. Other times called first-order logic, the predicate calculus are a prerequisite for set theory [3, p. 56]. It is a more complex language than the propositional calculus [18, p. 84] and introduces new logical symbols: $=$ (equality) and $\exists$ (existential quantifier) [18, p. 86].
ZFC	Zermelo-Fraenkel (ZF) is the set theory most commonly used amongst other versions of set theory [3, p. 56]. Set theory is concerned with manipulation of objects and their collections [3, p. 56] and "the key idea of set theory is the notion of a set as a single abstract object apart from the elements of which it is composed" [18, p. 456]. Metamath names it as Zermelo-Fraenkel set theory with Choice (ZFC) because of the axiom of choice "ax-ac" which "is usually considered an extension of ZF set theory rather than a proper part of it" [3, p. 63].
Derived	Those axioms classified as "derived" can be derived from the axioms of predicate calculus classified as "Pred. calc.". Yet, they were left in metamath because other subsystems can be derived and might be of interest for particular studies. [a]
Complex	The axioms grouped in this category are based on the theorems postulated from the ZFC set theory. After a theorem of complex number is proved, it is re-introduced as an axiom within metamath. For example, after proving "axresscn"[b], the axiom "ax-resscn"[c] is introduced to be used like any other axiom. Metamath justifies this by stating: "This lets us easily identify which axioms are needed for a particular complex number proof, without the obfuscation of the set theory used to derive them".
TG	TG stands for the Tarski-Grothendieck Axiom. It is only one axiom named **ax-groth**.[d] It could be considered as an extension of the ZFC set, yet Metamath prefers to put it into its own category. More details about it can be found in [e].

[a] http://us.metamath.org/mpegif/mmset.html#subsys
[b] http://us.metamath.org/mpegif/axresscn.html
[c] http://us.metamath.org/mpegif/ax-resscn.html
[d] http://us.metamath.org/mpegif/ax-groth.html
[e] http://us.metamath.org/mpegif/mmset.html#groth

Table 8: Description of the different categories of axioms within Metamath.

Axiom name	Acc. flow	Ax. category	Axiom name	Acc. flow	Ax. category	Axiom name	Acc. flow	Ax. category
ax-cc	25	?	ax-pre-mulgt0	844	complex	ax-11	5603	pred. cal.
ax-dc	3	?	ax-cnex	808	complex	ax-14	3227	pred. cal.
ax-1cn	4331	complex	ax-pre-sup	389	complex	ax-13	1689	pred. cal.
ax-icn	3355	complex	ax-addf	214	complex	ax-9v	16	pred. cal.
ax-1ne0	3119	complex	ax-mulf	173	complex	ax-12	3	pred. cal.
ax-i2m1	2802	complex	ax-4	16644	derived	ax-mp	32293	pro. cal.
ax-resscn	2795	complex	ax-120	8852	derived	ax-1	21977	pro. cal.
ax-rrecex	2705	complex	ax-16	5617	derived	ax-2	9105	pro. cal.
ax-rnegex	2407	complex	ax-10	4002	derived	ax-3	3566	pro. cal.
ax-mulrcl	2268	complex	ax-15	33	derived	ax-groth	8	TG
ax-mulcl	2217	complex	ax-100	14	derived	ax-sep	3714	ZFC
ax-cnre	2202	complex	ax-60	8	derived	ax-ext	2117	ZFC
ax-addcl	2126	complex	ax-110	4	derived	ax-nul	1856	ZFC
ax-mulcom	1791	complex	ax-50	4	derived	ax-pr	1841	ZFC
ax-1rid	1362	complex	ax-90	4	derived	ax-un	1669	ZFC
ax-pre-lttri	919	complex	ax-17	19285	pred. cal.	ax-pow	1449	ZFC
ax-pre-lttrn	916	complex	ax-gen	14259	pred. cal.	ax-rep	906	ZFC
ax-addrcl	910	complex	ax-7	9111	pred. cal.	ax-inf2	352	ZFC
ax-addass	899	complex	ax-8	8956	pred. cal.	ax-ac	54	ZFC
ax-distr	898	complex	ax-6	7216	pred. cal.	ax-reg	50	ZFC
ax-mulass	898	complex	ax-5	6190	pred. cal.	ax-inf	2	ZFC
ax-pre-ltadd	897	complex	ax-9	5826	pred. cal.			

Table 9: Summary for all flows leaving a source node and arriving to every sink node sorted and grouped by axiom category

Axiom name	Acc. flow	Ax. category	Axiom name	Acc. flow	Ax. category	Axiom name	Acc. flow	Ax. category
ax-mp	32293	pro. cal.	ax-resscn	2795	complex	ax-pre-ltadd	897	complex
ax-1	21977	pro. cal.	ax-rrecex	2705	complex	ax-pre-mulgt0	844	complex
ax-17	19285	pred. cal.	ax-rnegex	2407	complex	ax-cnex	808	complex
ax-4	16644	derived	ax-mulrcl	2268	complex	ax-pre-sup	389	complex
ax-gen	14259	pred. cal.	ax-mulcl	2217	complex	ax-inf2	352	ZFC
ax-7	9111	pred. cal.	ax-cnre	2202	complex	ax-addf	214	complex
ax-2	9105	pro. cal.	ax-addcl	2126	complex	ax-mulf	173	complex
ax-8	8956	pred. cal.	ax-ext	2117	ZFC	ax-ac	54	ZFC
ax-120	8852	derived	ax-nul	1856	ZFC	ax-reg	50	ZFC
ax-6	7216	pred. cal.	ax-pr	1841	ZFC	ax-15	33	derived
ax-5	6190	pred. cal.	ax-mulcom	1791	complex	ax-cc	25	?
ax-9	5826	pred. cal.	ax-13	1689	pred. cal.	ax-9v	16	pred. cal.
ax-16	5617	derived	ax-un	1669	ZFC	ax-100	14	derived
ax-11	5603	pred. cal.	ax-pow	1449	ZFC	ax-60	8	derived
ax-1cn	4331	complex	ax-1rid	1362	complex	ax-groth	8	TG
ax-10	4002	derived	ax-pre-lttri	919	complex	ax-110	4	derived
ax-sep	3714	ZFC	ax-pre-lttrn	916	complex	ax-50	4	derived
ax-3	3566	pro. cal.	ax-addrcl	910	complex	ax-90	4	derived
ax-icn	3355	complex	ax-rep	906	ZFC	ax-dc	3	?
ax-14	3227	pred. cal.	ax-addass	899	complex	ax-12	3	pred. cal.
ax-1ne0	3119	complex	ax-distr	898	complex	ax-inf	2	ZFC
ax-i2m1	2802	complex	ax-mulass	898	complex			

Table 10: Summary for all flows leaving a source node and arriving to every sink node sorted by accumulated flow

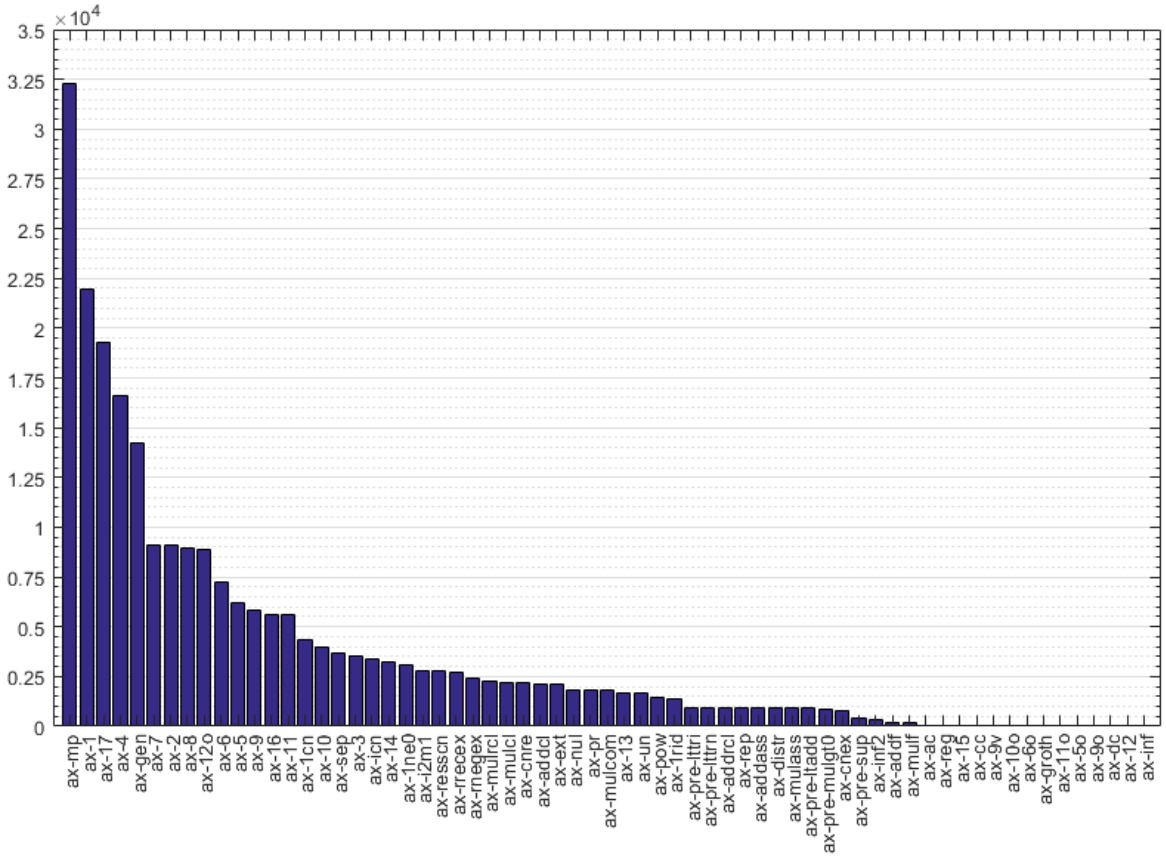


Fig. 25: Summary for accumulated flow for each individual source node.

As the data shows, the axiom with the greatest reachability regarding accumulated flow, or in other words, the one having the most quantity of disjoint paths to sinks, is by far the axiom "ax-mp"[16]. It represents the assertion rule of logic called *Modus Ponens* which permits to create new theorems from ancestor ones. "This rule states that if the wff ϕ is an axiom or a theorem, and the wff $\phi \rightarrow \psi$ is an axiom or a theorem, then the wff ψ is also a theorem." [3]. It shows the importance of propositional calculus for the network.

[16] http://us.metamath.org/mpeuni/ax-mp.html

The second most relevant axiom found is "ax-1"[17] which represents the first axiom of the propositional calculus. Metamath has "ax-mp", "ax-1", "ax-2", and "ax-3" to represent the axioms of propositional calculus[18]. Contrary to our expectations, the other two axioms of propositional calculus, "ax-2" and "ax-3", were not so well ranked by this approach with "ax-2" being the seventh and "ax-3" being the eighteenth ranked node.
In [18] we find that "from the point of view of its ability to express mathematical ideas, the propositional language has very limited power. [...] Nevertheless, the structure of the propositional language has much in common with the much more useful first-order languages". Motivated by that, each accumulated flow has been aggregated by its axiom category to better understand how each set of axioms contribute to the network. The aggregated data is summarized in table 11 and visually represented in figure 26.

10.2 Interpretations and discussion

Based on the results of max flow from axioms to sink nodes in the metamath graph, and after categorizing and aggregating the flows, we see that the axioms for predicate and propositional calculus play, respectively, the major role for all the derived theorems of the network. That seems consistent with the literature about mathematics.

As the set of axioms for complex number has been left on their own category on Metamath, we see how they also have a significant importance for the overall network. Just to remind the reader, they are postulated based of the ZFC set of axioms. They were introduced on Metamath on their own after being proved from the ZFC set so as to have a better separation of concerns. Thus, the "**complex**" category can be seen as an extension of the "**ZFC**" set.

As explained on table 8, the "**derived**" category can be seen as an extension of the "**Predicated calculus**" category. Considering that, we once again group the aggregated values to give a final vision about Metamath groups of axioms. This is illustrated on table 12 and on figure 27. It is possible to see that the **Predicated calculus** dominates the network, followed by the set of **Propositional calculus** and **ZFC** which have almost the same flow quantity.

The axioms **ax-cc**, **ax-dc**, and **ax-groth** from the two remaining axiom sets seem to have a very specific and limited purpose on the network if seen on their own category, since their flow values are negligible. We suspect, though, that these three axioms could be grouped into the **ZFC** category. Our mathematical knowledge limits us to make a judgment about it.

[17] http://us.metamath.org/mpeuni/ax-1.html
[18] http://us.metamath.org/mpeuni/mmset.html#scaxioms

Category	**Aggregated flow**
Predicated calculus	81381
Propositional calculus	66941
Complex numbers	42245
Derived	35182
ZFC	14010
?	28
TG	8

Table 11: Aggregated accumulated flows for each axiom set category

Category	**Aggregated flow**
Predicated calculus + Derived	116563
Propositional calculus	66941
ZFC + Complex numbers	56255
?	28
TG	8

Table 12: Grouped aggregated accumulated flows for each axiom set category

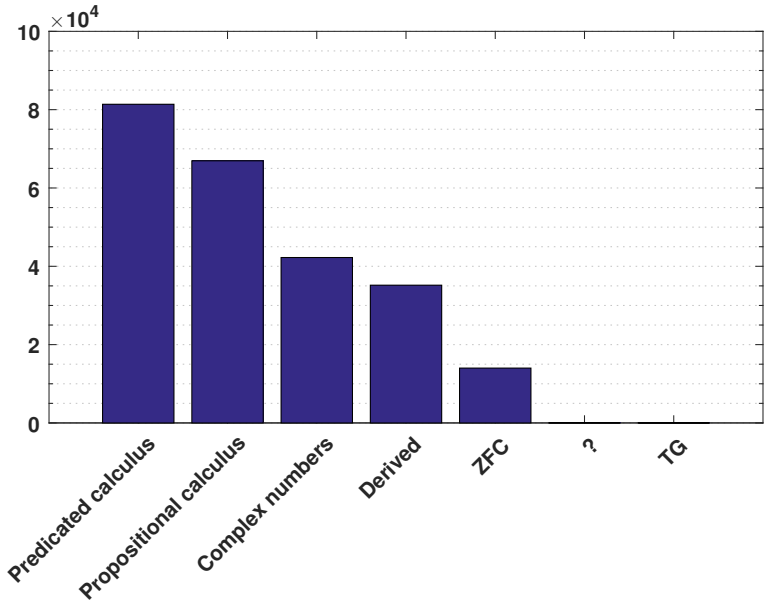


Fig. 26: Aggregated accumulated flows for each axiom set category.

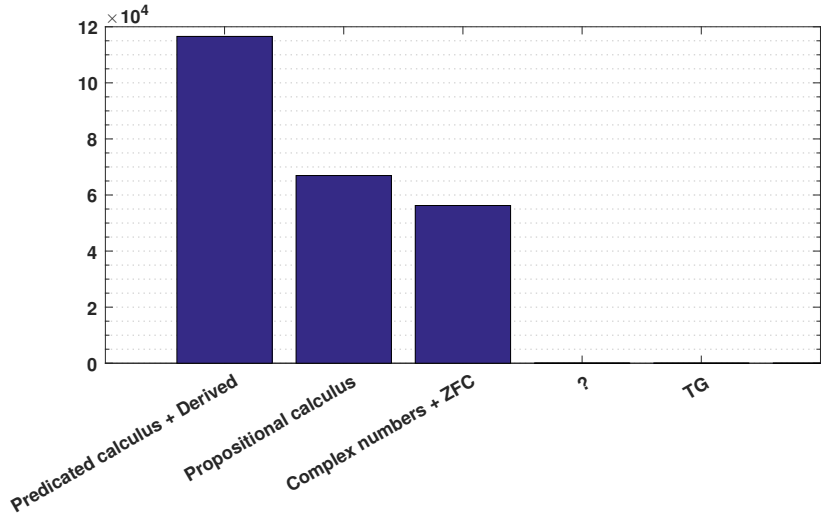


Fig. 27: Grouped aggregated accumulated flows for each axiom set category.

Conclusions

Mathematical proof assisted by software is something possible and feasible. Tools such as Metamath help out mathematicians and offer a standard way of querying and parsing information which benefit analysis such as the ones performed on this work. We have described the basics of Metamath, its file structure, and we have given a glimpse about its intricacies while analyzing it, so the new and not yet acquainted reader may grasp it easily.

As the mathematical knowledge does not grow randomly, but rather is driven by human intelligence, we have shown that some structure underlies the manner how theorems and axioms relate. We were quite surprised by the findings because theorems have a purpose to exist, and since theorems are not created at random, many natural restrictions are imposed for the connections of new theorems on the network. Only because a theorem is highly used it does not mean that necessarily all new theorems are likely to rely upon it. How likely is a highly connected theorem to be used by a new theorem, only because of its popularity? An individual analysis of theorems would be required to answer that question.

We have described how to read and interpret each of the possible degree measures; outdegrees act in a passive way, indegrees act in a more dynamic manner, and also that the total degree of nodes do not tell much about a node.

While studying the degree distribution, we have shown that the network has a tendency to be structured as a scale-free network, yet with some remarks. We have concluded that the total degree of nodes (δ) follows a lognormal distribution with high accuracy; also, that the indegree of nodes (δ^+) tends to follow a lognormal distribution yet without conclusive evidence; and finally, that the outdegree of the network (δ^-) seems to follow a power-law distribution. As it has been shown before, networks may evolve from a lognormal distribution to a power-law distribution through well-known mechanisms [12], thus the scale of a network is not a fixed and immutable structure but rather malleable as it evolves throughout the time. As we have not found similar research about the structure of the mathematical knowledge, we are not sure to affirm which mechanisms of network formation govern Metamath network. Another factor which makes it difficult to give a verdict about the distributions is the size of the dataset which contains only 17538 nodes. Many other degree studies have been performed on voluminous sets of data.

While decomposing the graph by sources, we have shown that Metamath has some high dependency upon three theorems: **sb3**, **sb4**, and **sb56**. Also, we have discovered that the **birthday** theorem was the last theorem found on that decomposition, meaning that it is the most dependent theorem of the network for the time being.

During the analysis of the proof steps of theorems, we have not found any conclusive evidence to affirm that it fits into some distribution, yet it may evolve to some more conclusive shape as the network evolves with new theorems.

Surprisingly enough, we have found a relationship between the size of a DFS-tree and its number of leaves. Starting a DFS from anywhere on the graph leads to an average number of nodes seven times bigger than the number of leaves that

the DFS-tree has. However, these values do not convey significant information or details about any particular graph structure as far as we have seen.

Lastly, we have studied the axioms through a max flow approach and we have also presented a clever trick to add constraints to nodes rather than on arcs. Also, we have described how to calculate disjoint paths between sources and sinks by using appropriate flow capacity on arcs. The results show that the sets of axioms for predicated calculus endow most of the network. The axioms of propositional calculus, although being a stronger language to describe mathematical knowledge, were ranked having nearly half the influence than the set of Predicated calculus, considering the influence based on disjoint paths between sinks and nodes. The ZFC set has been raked almost with the same relevance than the Propositional calculus. For the remaining three axioms, we suspect that their values could be grouped into the **ZFC** set. Since we are studying Metamath from a computer science perspective, we were not concerned by that details.

In this work, we have analyzed an area which is not so commonly analyzed with graphs compared to other natural and human phenomena. We hope to shed some light and insights on mathematical knowledge analysis.

References

1. Cléuzio Fonseca Filho. *História da computação: O Caminho do Pensamento e da Tecnologia [History of computing: The path of thought and technology]*. EDIPUCRS, 2007.
2. Rogério Santos Mol. Introdução à história da matemática [introduction to the history of mathematics]. *Belo Horizonte: CAED-UFMG*, 2013.
3. Norman Megill. Metamath: A computer language for pure mathematics. 1997.
4. I. Robinson, J. Webber, J. Webber, and E. Eifrem. *Graph Databases*. O'Reilly, 2013.
5. Flavio B Gonzaga, Valmir C Barbosa, and Geraldo B Xexéo. The network structure of mathematical knowledge according to the wikipedia, mathworld, and dlmf online libraries. *Network Science*, 2(03):367–386, 2014.
6. Réka Albert, István Albert, and Gary L Nakarado. Structural vulnerability of the north american power grid. *Physical review E*, 69(2):025103, 2004.
7. Edward T Lu and Russell J Hamilton. Avalanches and the distribution of solar flares. *The astrophysical journal*, 380:L89–L92, 1991.
8. Mukund Seshadri, Sridhar Machiraju, Ashwin Sridharan, Jean Bolot, Christos Faloutsos, and Jure Leskove. Mobile call graphs: beyond power-law and lognormal distributions. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 596–604. ACM, 2008.
9. Lada A Adamic and Bernardo A Huberman. Zipf's law and the internet. *Glottometrics*, 3(1):143–150, 2002.
10. Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, volume 29, pages 251–262. ACM, 1999.
11. Albert-László Barabási. The architecture of complexity. *IEEE control systems*, 27(4):33–42, 2007.
12. Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.
13. Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet mathematics*, 1(2):226–251, 2004.
14. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction to algorithms. third, 2009.
15. Valmir C Barbosa and Eli Gafni. Concurrency in heavily loaded neighborhood-constrained systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 11(4):562–584, 1989.
16. Frederick Mosteller. Understanding the birthday problem. In *Selected Papers of Frederick Mosteller*, pages 349–353. Springer, 2006.
17. Boris V Cherkassky and Andrew V Goldberg. On implementing push-relabel method for the maximum flow problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 157–171. Springer, 1995.
18. P Hinman. Fundamentals of mathematical logic. 2005. *AK Peters*.

A Appendices

A.1 JFlex lexical analyzer file

```
%{
    private String string_found = "";
    private StringBuilder string_builder = new StringBuilder();
    private Symbol symbol(int type) {
        return new Symbol(type, yyline, yycolumn);
    }
    private Symbol symbol(int type, Object value) {
        return new Symbol(type, yyline, yycolumn, value);
    }
%}

/*  Macro Declarations */

LineTerminator      = \r|\n|\r\n
InputCharacter      = [^\r\n]
WhiteSpace          = {LineTerminator} | [ \t\f]

/* In order to avoid REGEX problems, all special characters are escaped */
/* We have only removed the '$' character from here to avoid issues with
   metamath reserved tokens */
SpecialChars        =
\‘|\~|\!|\@|\#|\%|\^|\&|\*|\(|\)|\-|\_|\=|\+|\[|\]|\{|\}|\;|\:|\’|\"|\,|\.|\<|\>|\/|\?|\\|\|

PrintableChars      = ([a-zA-Z0-9]|{SpecialChars})

/* A label token consists of any combination of letters, digits,
   and the characters hyphen, underscore, and period */
Label               = [a-zA-Z0-9\-\_\.]+

/* A math symbol token may consist of any combination of the 93 printable
   standard ascii characters other than '$'. */
/* See metamath book p. 93 for more details. */
MathSymbol          = {PrintableChars}+

%state STRING, PROOF, COMPACT_PROOF, COMMENT, INCLUDE

%%
/* ------------------------Lexical Rules Section---------------------- */

/*
   This section contains regular expressions and actions, i.e. Java
   code, that will be executed when the scanner matches the associated
   regular expression. */
/* YYINITIAL is the state at which the lexer begins scanning. */

<YYINITIAL> {

    /* Blocks syntax */
    "${"         { return symbol(sym.SCOPE_START); }
    "$}"         { return symbol(sym.SCOPE_END); }
    "$("         { yybegin(COMMENT); }
    "$["         { yybegin(INCLUDE); }

    /* non-labeled declarations */
    "$c"         { return symbol(sym.CONSTANT_STMT); }
    "$v"         { return symbol(sym.VARIABLE_STMT); }
    "$d"         { return symbol(sym.DISJUNCT_VARIABLE_STMT); }

    /* labeled declarations */
    "$f"         { return symbol(sym.VARIABLE_TYPE_HYPOTHESIS_STMT); }
    "$e"         { return symbol(sym.LOGICAL_HYPOTHESIS_STMT); }
    "$a"         { return symbol(sym.AXIOMATIC_ASSERTION_STMT); }
```

```
    "$p"          { return symbol(sym.PROVABLE_ASSERTION_STMT); }
    "$="          { yybegin(PROOF); return symbol(sym.PROOF_STMT); }
    "$."          { return symbol(sym.STMT_END); }

    /* Since labels are a subset of math symbols (in terms of regex), we need
        to change to a state where we will look ahead and see if there are any
        $p, $a, $e, or $f symbol. That will tell us when we have found a LABEL
        or a MATH_SYMB */
    {MathSymbol} { string_found = yytext(); yybegin(STRING); }

    /* Don't do anything if whitespace is found */
    {WhiteSpace}       { /* just skip what was found, do nothing */ }
}

/* The STRING state here means that either a label or math symbol was found */

<STRING> {

    {WhiteSpace}  { /* just skip what was found, do nothing */ }

    \$(e|f|a|p)   { yypushback(2); yybegin(YYINITIAL); return symbol(sym.LABEL, string_found); }

    .             { yypushback(1); yybegin(YYINITIAL); return symbol(sym.MATH_SYMB, string_found); }
}

/* The PROOF section refers to all code after a $= token */

<PROOF> {

    {WhiteSpace}  { /* just skip what was found, do nothing */ }

    /* Round brackets aren't present when the proof isn't in compact form */
    "("           { return symbol(sym.LPARENT); }
    ")"           { yybegin(COMPACT_PROOF); string_builder.setLength(0); return symbol(sym.RPARENT); }

    {Label}       { return symbol(sym.LABEL,yytext()); }

    "$."          { yybegin(YYINITIAL); return symbol(sym.STMT_END); }

    /* Sub-state */
    <COMPACT_PROOF> {
        {WhiteSpace}  { /* just skip what was found, do nothing */ }
        {Label}       { string_builder.append(yytext()); }
        "$."          { yybegin(PROOF); yypushback(2); return symbol(sym.COMPACT_PROOF, string_builder.toString());
    }
}

<COMMENT> {
    {WhiteSpace}  { /* just skip what was found, do nothing */ }
    "$)"          { yybegin(YYINITIAL); }
    .             { /* Do nothing */ }
}

<INCLUDE> {
    {WhiteSpace}  { /* just skip what was found, do nothing */ }
    "$]"          { yybegin(YYINITIAL); return symbol(sym.INCLUDE_END); }
    .             { /* Do nothing */ }
}

/* No token was found for the input so through an error.  Print out an
   Illegal character message with the illegal character that was found. */
[^]                    { throw new Error("Illegal character <"+yytext()+">"); }
```

A.2 CUP syntactical analyzer file

```
1
2  /* Terminals (tokens returned by the scanner). */
3
4  terminal            LPARENT, RPARENT;
5  terminal            SCOPE_START, SCOPE_END;
6  terminal            INCLUDE_START, INCLUDE_END;
7  terminal            CONSTANT_STMT, VARIABLE_STMT;
8  terminal            DISJUNCT_VARIABLE_STMT, VARIABLE_TYPE_HYPOTHESIS_STMT;
9  terminal            LOGICAL_HYPOTHESIS_STMT, AXIOMATIC_ASSERTION_STMT;
10 terminal            PROVABLE_ASSERTION_STMT, PROOF_STMT, COMPACT_PROOF,STMT_END;
11 terminal String     LABEL, MATH_SYMB;
12
13 /* Non terminals */
14
15 non terminal    initial_symbol, empty;
16 non terminal    math_symb_list, label_list;
17 non terminal    const_declaration, var_declaration, disjoint_var_declaration;
18 non terminal    var_type_hypothesis_declaration, logical_hypothesis_declaration;
19 non terminal    axiomatic_declaration, theorem_declaration, proof_declaration,              proof_labels;
20 non terminal    include_section, scope_section;
21
22 start with initial_symbol;
23
24 /* The grammar */
25
26 initial_symbol      ::=
27                       const_declaration                initial_symbol
28                     | var_declaration                  initial_symbol
29                     | disjoint_var_declaration         initial_symbol
30                     | var_type_hypothesis_declaration  initial_symbol
31                     | logical_hypothesis_declaration   initial_symbol
32                     | axiomatic_declaration            initial_symbol
33                     | theorem_declaration              initial_symbol
34                     | include_section                  initial_symbol
35                     | scope_section                    initial_symbol
36                     | empty
37                     ;
38
39 empty               ::=
40                     ;
41
42 math_symb_list      ::=
43                       math_symb_list MATH_SYMB:m
44                     |
45                       MATH_SYMB:m
46                     ;
47
48 const_declaration   ::=
49                       CONSTANT_STMT
50                       math_symb_list STMT_END
51                     ;
52
53 var_declaration     ::=
54                       VARIABLE_STMT
55                       math_symb_list STMT_END
56                     ;
57
58 disjoint_var_declaration ::=
59                       DISJUNCT_VARIABLE_STMT
60                       MATH_SYMB:m
61                       math_symb_list STMT_END
62                     ;
63
64
65
```

```
66
67
68  /* See Metamath book p. 105 for references */
69  var_type_hypothesis_declaration ::= /* $f */
70                        LABEL:l
71                        VARIABLE_TYPE_HYPOTHESIS_STMT
72                        MATH_SYMB:constant MATH_SYMB:variable
73                        STMT_END
74                      ;
75
76  logical_hypothesis_declaration ::= /* $e */
77                        LABEL:l
78                        LOGICAL_HYPOTHESIS_STMT
79                        MATH_SYMB:constant
80                        math_symb_list
81                        STMT_END
82                      ;
83
84  axiomatic_declaration ::= /* $a */
85                        LABEL:l
86                        AXIOMATIC_ASSERTION_STMT /* $a */
87                        MATH_SYMB:m
88                        math_symb_list
89                        STMT_END
90                      ;
91
92  theorem_declaration ::=
93                        LABEL:l
94                        PROVABLE_ASSERTION_STMT /* $p */
95                        MATH_SYMB:m
96                        math_symb_list
97                        proof_declaration
98                        STMT_END
99                      ;
100 proof_declaration   ::=
101                       PROOF_STMT /* $= */
102                       label_list
103                     | PROOF_STMT /* $= */
104                       LPARENT proof_labels RPARENT COMPACT_PROOF:p
105                     ;
106 proof_labels        ::=
107                       label_list
108                     | empty
109                     ;
110
111 label_list          ::=
112                       label_list LABEL:l
113                     | LABEL:l
114                     ;
115
116
117 include_section     ::=
118                       INCLUDE_START
119                       INCLUDE_END
120                     ;
121
122 scope_section       ::=
123                       SCOPE_START
124                         initial_symbol
125                       SCOPE_END
126                     ;
127
```