Structure in Theorem Proving: Analyzing and Improving the Isabelle Archive of Formal Proofs

Fabian Huch

Technische Universität München, Boltzmannstraße 3, 85748 Garching, Germany

Abstract

The Isabelle Archive of Formal Proofs has grown to a significant size in the past years. It makes up for an impressive body of research, which enables a number of statistical approaches to various aspects in theorem proving, and has not yet been utilized exhaustively. However, the growing size also poses some challenges to address: Material becomes increasingly harder to find, reusability and ease of understanding become more important. This thesis abstract summarizes my research plans on those topics and briefly touches on preliminary results, which indicate that the node in-degree of the dependency graph of the archive follows a scale-free distribution.

Keywords

theorem proving, software engineering, complex networks, proof mining

1. Introduction

Isabelle is an interactive theorem prover [1] with a large collection of formalized material, the *Archive of Formal Proofs* (AFP). At the time of writing, the AFP consists of nearly three million lines of code, and more than 167 000 lemmas have been proven in its close to 600 different entries [2].

The entities defined by those theories and their relationships can be described as a dependency graph (sometimes also referred to as *General Dependency Network* [3]). For software systems, dependency graphs have been found to exhibit certain structural properties. For instance, their in-degree (k_{in}) distribution is often *scale-free*, i.e. it follows some power law $\Pr(k_{in}) \propto k_{in}^{-\gamma}$ [4], in contrast to random graphs where each possible edge is present with a fixed probability (Erdős–Rény model). Graphs with such topological properties are called *complex networks*, which have been studied in the context of real-world graphs from many different areas [5]. In contrast to research focused on object-oriented characteristics (e.g., classes and inheritance), structural analysis on dependency graphs of software systems can be transferred to the field of formal theories more directly.

Another important application of this dependency network is proof automation. Data mining methods to extract patterns from graphs have been extensively studied, for instance *frequent itemset mining* to find subsets of nodes that frequently occur together [6, 7]. The AFP provides

CICM 2021: 14th Conference on Intelligent Computer Mathematics, July 26–31, 2021, Timisoara, Romania
huch@in.tum.de (F. Huch)
https://www21.in.tum.de/home/~huch (F. Huch)
0000-0002-9418-1580 (F. Huch)

enough data to enable pattern mining for proofs as well as evaluate the effectiveness of derived automation.

2. Related Work

Regarding the AFP, empirical analysis has already been done by Blanchette et al. in [8]. They measured the AFP size and authorship distribution and how those evolved over time, computed the session graph, proof depths, and compared proof size to definition size and statement complexity (by number of clauses). Moreover, they benchmarked how many theorems *sledgehammer* could solve on a representative test suite. This research provides a good baseline of empirical data; I plan to follow up on it by analyzing the underlying GDN of AFP theories, which can possibly yield some deeper results.

Very recently, usability of the AFP has also gotten some attention. Two different search engines for theory contents were introduced: The *FindFacts* search by Huch and Krauss in [9], and the concept-oriented *SErAPIS* engine by Stathopoulos et al. in [10]. Additionally, MacKenzie et al. evaluated usability of the AFP webpage in [11] and built the prototype for a re-design.

Complex networks for software systems were first introduced by Myers in [4]. In [12], Zimmermann and Nagappan measured the correlation between software defects and a comprehensive list of network metrics; Šubelj and Bajec later surveyed quality indicators, and performed package prediction by clustering on the network [13]. While there is no concern about defects in theorem proving due to the nature of the field, my aim is to transfer some of those findings to formal theories in order to detect structural problems that impair reusability and readability.

For proof automation, the idea of learning from existing material has been around for some time. In Isabelle, Duncan derived tactics from proofs script sequences using Markov models and genetic programming with the goal of full automatization in [14], which was successful for less complex lemmas. More recently, Nawaz et al. used high utility itemset mining on a syntactic level to discover patterns in proofs scripts of *PVS* [15]. In contrast, I aim to extract patterns from the more fine-grained theorem dependency graph. This also makes it possible to utilize structured Isar proofs in the pattern extraction.

3. Research Topics

In the following, my research plans are separated into concrete questions and topics that involve engineering tasks.

3.1. Research Questions

RQ1: Do theory dependency networks follow the topological patterns typically found in complex networks?

RQ2: Which metrics are relevant quality indicators for formal theories? Can they expose structural problems?

RQ3: Is it possible to detect elements that need to be refactored? Can theory- and session-structure be predicted to generate refactoring recommendations?

RQ4: How can visualization be helpful to better understand the structure of formalizations?

RQ5: Which patterns can we learn from theorem networks? Can they be used to improve automation?

3.2. Further Topics

There are several aspects I want to improve on in the AFP. By introducing digital object identifiers, entries can be much better referenced. Identifiers could potentially even be assigned to individual concepts of an entry, though creating such a mapping poses lots of challenges. Moreover, introducing more fields for entries, such as subject classification and links to (print-)publications, can help organizing the AFP in the future. Finally, the AFP website needs to be improved for better usability – there is already a prototype (as discussed in section 2), which still needs to be integrated.

Currently, the AFP does not allow proofs by `sorry` (which are admitted with the help of an oracle [16]). However, it is often desirable to have what is sometimes referred to as *Formal Abstract* [17]: a formalization of results from literature without rigorous proofs. To that end, I want do add a *proof by reference* mechanism, which weakly checks references and allows such proofs to be added to the AFP in a separate, less trusted, session group.

In Isabelle, I plan to integrate the FindFacts search into the prover IDE (and add support for type-classes and locales), as well as tooling for clone detection and advanced IDE functionality such as structural search and replace.

4. Preliminary Results

The dependency graph of Isabelle and the AFP (release 2021) consists of approximately $2.1 \cdot 10^6$ nodes and $2.5 \cdot 10^8$ directed edges (when considering all types of entities and relationships). Figure 1 shows the in-degree distribution of that graph. For $k_{in} > 2$, it is clearly linear on the log-log scale; this indicates a power-law distribution, which is typical for scale-free networks [5]. The individual AFP components show similar characteristics; this illustrates why complex network science is relevant to the topic.

References

[1] L. C. Paulson, Isabelle: The next seven hundred theorem provers, in: E. Lusk, R. Overbeek (Eds.), 9th International Conference on Automated Deduction, Springer Berlin Heidelberg, Berlin, Heidelberg, 1988, pp. 772–773.

[2] Archive of Formal Proofs, Statistics, 2021. URL: https://www.isa-afp.org/statistics.html.

[3] M. Savic, M. Ivanovic, Graph clustering evaluation metrics as software metrics, in: Z. Budimac, T. G. Grbac (Eds.), Proceedings of the 3rd Workshop on Software Quality Analysis, Monitoring, Improvement and Applications (SQAMIA 2014), Lovran, Croatia,

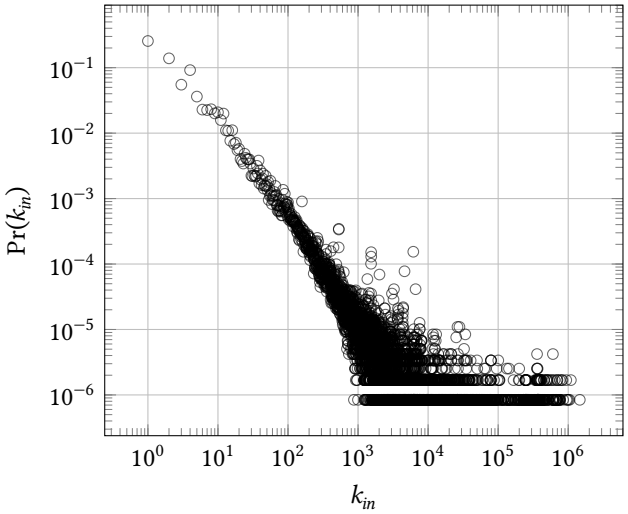

Figure 1: In-Degree distribution the dependency graph from Isabelle and AFP theories (release 2021), on a log-log scale.

September 19-22, 2014, volume 1266 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2014, pp. 81–89.

[4] C. R. Myers, Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs, Phys. Rev. E 68 (2003) 046116. doi:10.1103/PhysRevE.68.046116.

[5] R. V. Solé, R. Ferrer-Cancho, J. M. Montoya, S. Valverde, Selection, tinkering, and emergence in complex networks, Complexity 8 (2002) 20–33. doi:10.1002/cplx.10055.

[6] C. C. Aggarwal, H. Wang (Eds.), Managing and Mining Graph Data, volume 40 of *Advances in Database Systems*, Springer US, 2010. doi:10.1007/978-1-4419-6045-0.

[7] D. J. Cook, L. B. Holder, Mining Graph Data, John Wiley & Sons, 2006. doi:10.1002/9780470073049.

[8] J. C. Blanchette, M. Haslbeck, D. Matichuk, T. Nipkow, Mining the Archive of Formal Proofs, in: M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, V. Sorge (Eds.), Intelligent Computer Mathematics, Springer International Publishing, Cham, 2015, pp. 3–17.

[9] F. Huch, A. Krauss, Findfacts: A scalable theorem search, in: Isabelle Workshop, 2020.

[10] Y. Stathopoulos, A. Koutsoukou-Argyraki, L. Paulson, Serapis: A concept-oriented search engine for the isabelle libraries based on natural language, in: Isabelle Workshop, 2020.

[11] C. MacKenzie, J. Fleuriot, J. Vaughan, An evaluation of the archive of formal proofs, arXiv preprint arXiv:2104.01052 (2021).

[12] T. Zimmermann, N. Nagappan, Predicting defects using network analysis on dependency graphs, in: Proceedings of the 30th International Conference on Software Engineering, ICSE '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 531–540. doi:10.1145/1368088.1368161.

[13] L. Šubelj, M. Bajec, Software systems through complex networks science: Review, analysis and applications, in: Proceedings of the First International Workshop on Software Mining,

SoftwareMining '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 9–16. doi:10.1145/2384416.2384418.

[14] H. Duncan, The Use of Data-Mining for the Automatic Formation of Tactics, Ph.D. thesis, University of Edinburgh, 2007.

[15] M. S. Nawaz, P. Fournier-Viger, J. Zhang, Proof Learning in PVS with Utility Pattern Mining, IEEE Access 8 (2020) 119806–119818. doi:10.1109/ACCESS.2020.3004199.

[16] M. Wenzel, et al., The Isabelle/Isar reference manual, Technische Universität München, 2021.

[17] T. Hales, Big conjectures, in: Computer-aided mathematical proof, 2017.