Mining the Archive of Formal Proofs

Jasmin Christian Blanchette[1,2], Maximilian Haslbeck[3], Daniel Matichuk[4,5], and Tobias Nipkow[3](✉)

[1] Inria Nancy and LORIA, Villers-lès-Nancy, France
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[3] Fakultät für Informatik, Technische Universität München, Munich, Germany
nipkow@in.tum.de
[4] NICTA, Sydney, Australia
[5] University of New South Wales, Sydney, Australia

Abstract. The Archive of Formal Proofs is a vast collection of computer-checked proofs developed using the proof assistant Isabelle. We perform an in-depth analysis of the archive, looking at various properties of the proof developments, including size, dependencies, and proof style. This gives some insights into the nature of formal proofs.

1 Introduction

The *Archive of Formal Proofs* (*AFP*, http://afp.sf.org) is an online library of proof developments for the proof assistant Isabelle [21] contributed by its users. The AFP is organized like a scientific journal. Each contribution is called an *article* and is a collection of Isabelle *theories*, i.e., files with definitions, lemmas, and proofs in Isabelle's input language Isar [20,29]. A few articles are ML programs that realize specialized definition or proof facilities. The AFP was started in 2004. This paper refers to the AFP snapshot of 16 April 2015, which contains a total of 64,497 lemmas. The term *lemmas* subsumes theorems and corollaries throughout the paper.

The purpose of this paper is to analyze the following properties of the AFP: general size statistics, dependency graph, proof style and proof size, and performance of fully automatic proof. We attempt to answer a number of questions:

- To what extent are AFP articles reused as the basis of other articles?
- How large are articles?
- What percentage of text is taken up by definitions, lemmas, and proofs?
- How many contributors are behind the AFP, and how large are their contributions?
- Does the dependency graph share characteristics with citation graphs in the scientific literature?
- How did the AFP evolve over time?
- Can we estimate the size of a proof from the statement to be proved?
- What is the typical structure of lemma statements? Are they mostly equalities, Horn clauses, or more complex formulas?

DOI: 10.1007/978-3-319-20615-8_1

– How successful is the automatic proof tool Sledgehammer [4,22] at discharging various goals from the AFP, as opposed to the smaller benchmark sets used in earlier evaluations?

This appears to be the first in-depth analysis of a large collection of computer-checked formal proofs, with the partial exception of Josef Urban's work on the Mizar Problems for Theorem Proving [26], a library of problems for first-order automatic theorem proving generated from the Mizar Mathematical Library [1].

The AFP is heavily biased towards computer science: of the 215 articles, 146 are indexed under computer science and only 82 articles under either mathematics or logic. (Entries may occur in multiple categories.) In contrast, the Mizar Mathematical Library is heavily biased towards mathematics.

Although Isabelle is a generic proof assistant supporting several object logics, all AFP articles use higher-order logic (HOL) as their object logic. Isabelle's version of HOL corresponds to Church's simple type theory [8] extended with polymorphism and Haskell-style type classes. HOL allows nested function types and quantification over functions. Predicates are simply functions to the Boolean type. Named functions are called *constants* in HOL terminology, even if they take arguments. Thus, in the formula $x + 0 = x$, both 0 and + are constants, whereas x is a variable. Otherwise, HOL conventions are a mixture of mathematics and functional programming.

2 Sizes

In its 11 years, the AFP has grown to one million lines of "code" (LOC)—1,018,800 LOC to be precise—where "code" refers to definitions and proofs (including comments but not empty lines). The growth of the AFP over time is shown in Figs. 1 and 2. The growth in Fig. 1 looks roughly linear. If one examines the growth rate, it fluctuates but has an upward trend. The development of the total number of authors that have ever contributed to the AFP is shown in Fig. 3 and it is similar to the size graph. The number of authors active each year, shown in short lighter (in colour: pink) bars in Fig. 3, has only been growing slowly.

The distribution of sizes of the articles is shown in Fig. 4. Half the articles have up to 2,000 LOC, beyond that the number of articles decays sharply, but with a long tail, not all of which is visible in the figure: 9 AFP articles are larger than 20,000 LOC; the largest AFP article (77,100 LOC) is `JinjaThreads`, a formalization of a dialect of Java by Andreas Lochbihler [16]. At first sight it is not clear what distribution Fig. 4 follows. Initially, it shows an exponential decay, but a better fit is a power law: a log-log plot shows something close to a straight line. Many similar phenomena, e.g., file sizes on the Internet [9], also follow a power law.

2.1 Definitions vs. Lemmas vs. Proofs

The three largest categories of text in the AFP are proofs, lemma statements, and definitions (in that order):

Proofs:	593,828 LOC (58 %)
Lemma statements:	192,576 LOC (19 %)
Definitions:	85,808 LOC (8 %)

Above we have only counted proofs associated with lemmas.

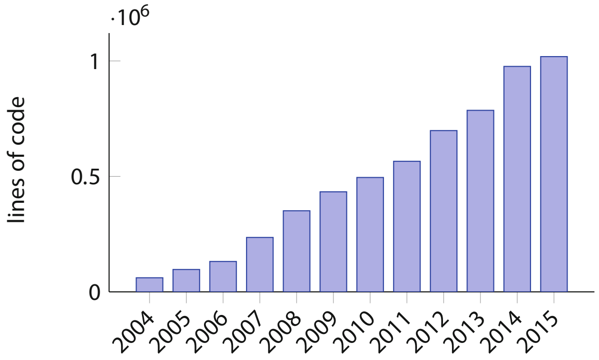

Fig. 1. Size of AFP over time (cumulative)

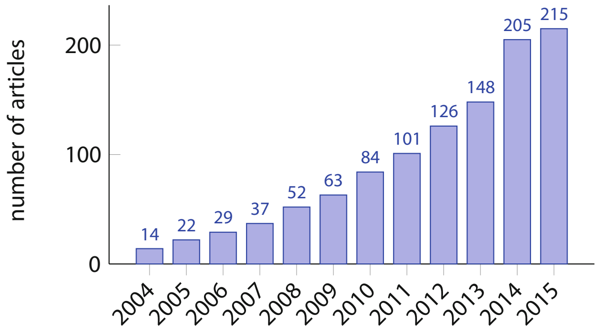

Fig. 2. Number of AFP articles over time (cumulative)

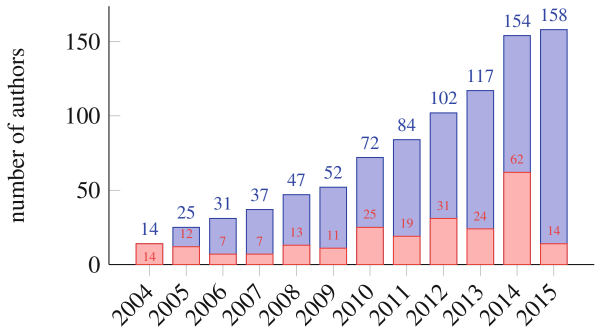

Fig. 3. Number of AFP authors over time (cumulative)

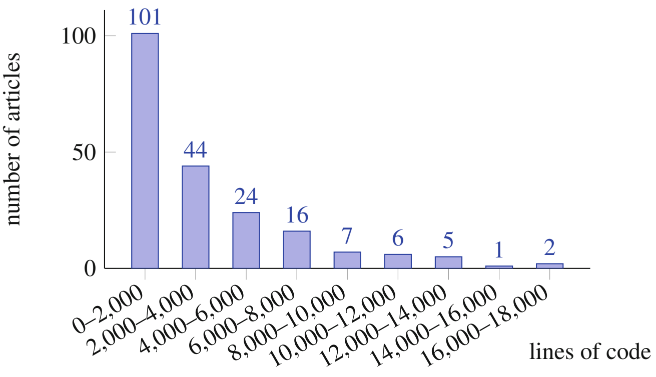

Fig. 4. Sizes of articles

In his study of the textual sizes of the libraries distributed with four proof assistants, Wiedijk [30] measured the following percentages for the above three categories (excluding empty lines): 62/14/1.4 for HOL Light, 50/21/8 for Isabelle, 60/12/10 for Coq, and 84/9/3 for Mizar. The discrepancy between the AFP and Wiedijk's numbers for Isabelle should not surprise because of the differences between the source texts (e.g., applications vs. foundations, degree of polish, age (2015 vs. 2007)) and because of slightly different counting schemes (e.g., proofs associated with lemmas vs. all proofs). As another example that these numbers can fluctuate take the verified C compiler (in Coq) by Leroy [15]: only 44 % of the space (excluding empty lines and comments) is taken up by proofs, 21 % by lemma statements and "supporting definitions", and 24 % by the definition of the compiler and the semantics.

If instead of the size we compare the number of lemmas and definitions, the ratio for the AFP is 64,497/17,909 $\approx$ 3.6. The ratio for the proof of the odd-order theorem in Coq is very similar: 13,000/4,000 $\approx$ 3.25 [11]. This echoes an old adage:

One good definition is worth three theorems.
— Alfred Adler, "Mathematics and Creativity," *The New Yorker* (1972)

2.2 Proof Depth

Isabelle proofs are block-structured and can be nested, i.e., proofs can have subproofs, sub-subproofs, and so on, like the blocks in a programming language. The maximum depth of a proof is a potential indication of its complexity. At the same time, it is also a potential indication that a monolithic proof should be refactored into smaller lemmas.

Figure 5 shows the number of lemmas at each proof depth in the AFP. A proof depth of 1 means that no structured proof commands were used, otherwise known as a *proof script*. The logarithmic y-axis reveals a nearly perfect exponential distribution. The vast majority of proof goals never exceeds a depth of 1. But

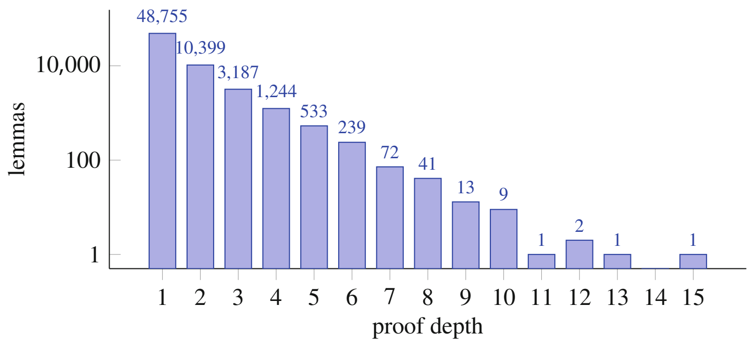

Fig. 5. Number of lemmas at each proof depth

still more than 1 % have a depth of 5 or more, and there is even one proof of depth 15.

It would be interesting to compare this with other block-structured proof languages, such as Mizar [18] and the TLA$^+$ Proof System [7]. Proofs written in the latter system tend to have a richer hierarchic structure. It is not clear to us whether this is due to the application area (the verification of concurrent and distributed algorithms), to specific features of the proof language, or simply to the personal preferences of its users.

3 The Imports Graph

Figure 6 shows the AFP *imports graph*. The nodes of the graph are the AFP articles. We say that an AFP article E_2 *imports* an article E_1 if some theory

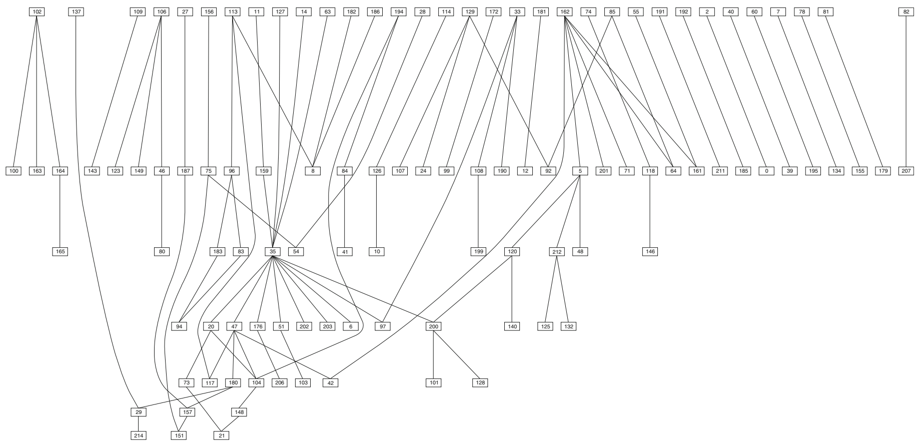

Fig. 6. The AFP imports graph

of E_2 imports some theory of E_1. In this case there is a directed edge from E_1 to E_2. We say that E_2 *depends on* E_1 if there is a non-empty path from E_2 to E_1. The graph is a dag; in Fig. 6 edges always go downwards. We do not show transitive edges and isolated nodes.

One of the questions we want to answer in this section is whether the imports graph share characteristics with citation graphs in the scientific literature.

The key characteristics of the graph are as follows:

Nodes:	215	Depth:	6
Isolated nodes:	106	Max. out-degree:	9
Edges:	97	Max. in-degree:	4

The top three articles with respect to their out-degree, i.e., the three most popular articles, are shown in Fig. 7. The top article, `Collections`, is a framework for efficient implementations of common collection types like sets and maps [14]. All three articles are of a generic nature and designed for reuse.

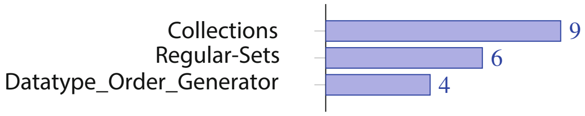

Fig. 7. The most popular AFP articles

3.1 Weakly Connected Components

Now we consider the weakly connected components (WCC) in the graph, i.e., the maximal subgraphs such that from any node there is a path to any other node where edges may be followed either forwards or backwards. Each WCC is a group of loosely related articles. When mining the citation graphs of different subfields of computer science, it was observed [2] that there is always one large WCC that covers 80–90 % of the nodes, and the second largest WCC is smaller by three orders of magnitude: almost everything is connected to almost everything else. A similar phenomenon can be observed in the AFP, although the numbers are smaller. The largest WCC has 70 nodes (1/3 of the graph), whereas the second largest one has 5 nodes.

3.2 The Most Productive Contributors

We should also like to acknowledge the most productive contributors to the AFP. Figure 8 shows the top 5 authors in terms of lines of code they contributed. Each author of an n-author article is assumed to have contributed $1/n$. It turns out that the top 5 authors have contributed a third of the AFP. The contributions are all in programming languages, data structures, and model checking, not in mathematics. The computer science bias is even more overwhelming than the classification of all articles into computer science versus mathematics and logic mentioned in the introduction would have lead us to expect.

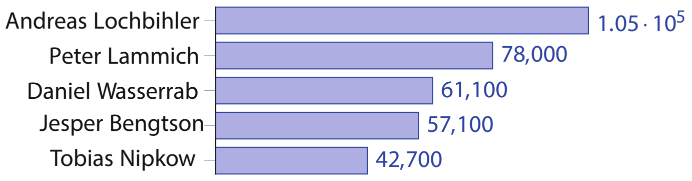

Fig. 8. The most productive authors

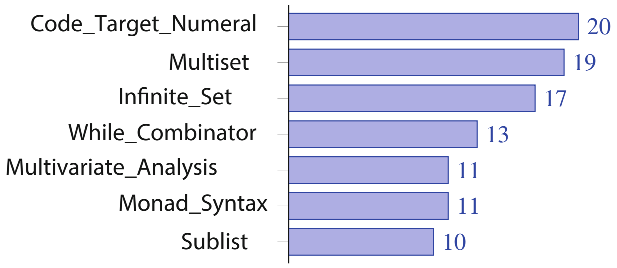

Fig. 9. The most popular library theories

3.3 Library Theories

Isabelle/HOL provides the theories `Main` and `Complex_Main` that contain many standard theories such as sets, relations, lists, natural numbers, integers, and real numbers. Many applications build on either of those. In addition, there is a large number of library theories that are included with Isabelle/HOL and can be imported selectively. The difference to the AFP is that these library theories are originated by the Isabelle developers and are specifically designed for reuse. Figure 9 shows the most popular library theories and how often they are used. The top theories are a mixture of specific theories (e.g., `Multiset`) and substantial mathematical developments (e.g., `Multivariate_Analysis` [12]).

Comparing Fig. 9 with Fig. 7, we find that the top library theories are imported twice as often as the top AFP articles, although both are designed for reuse. There are a number of explanations: library theories are developed by the Isabelle developers who concentrate on the most fundamental theories; adding a theory to the library is a more lightweight process than adding a new article to the AFP; the library was started at least four years before the AFP.

4 Lemma Statement Size vs. Proof Size

In addition to complete proof developments, the size of individual lemma proofs can also be considered. A naive measurement could consider the number of lines between the `lemma` and `qed` keywords to be the size of a proof. However, the intent of measuring the proof size is generally to estimate the effort required to produce that proof. Such a naive measurement will fail to capture the cumulative nature of proofs; for example, a simple corollary will seem to have a trivial proof despite depending on some large result.

4.1 Previous Work

Matichuk *et al.* [17] chose to consider the size of the proof of a lemma as the total number of lines required to prove it. This includes (recursively) the sizes of all lemmas the proof depends on. Their study sought to establish a *leading measure* of proof effort by building upon previous work [25], which demonstrated a linear relationship between proof effort (measured in person-weeks) and proof size. Matichuk *et al.* investigated the relationship between proof size and statement size of lemmas. The motivation is that the specification size for a proof (i.e. the statement size of its top-level theorem) is much easier to calculate early in a proof development, and thus it would be valuable to be able to use this to estimate the eventual size of the entire proof. Statement size was measured as the total number of unique constants used in the lemma statement, including (recursively) all constants used in their definitions.

This proved to be susceptible to lemma *over-specification*, where constants were mentioned but never interpreted, i.e., the lemma could instead have been abstracted over those constants. In these cases, the size of the lemma statement was much larger than its proof would indicate. This prompted an *idealized* measure, which discounts constants whose definitions are never unfolded in the entire proof.

The study examined six software verification proof developments: four proofs from the L4.verified project as well as `JinjaThreads` and `SATSolver Verification` from the AFP. They compared the raw statement and idealized statement sizes to the proof sizes for all lemmas in each proof and found a consistent quadratic relationship, which was strengthened by the idealized measure. However, the exact nature of the relationship was different between each proof: a model built against one proof does not necessarily fit others.

4.2 Analysis Against the AFP

We performed the same analysis against three of the largest AFP articles, shown in Table 1 and Fig. 10. Here R^2 is the usual coefficient of determination for statistical models, where an R^2 of 1 indicates that the model fits the data perfectly. We see that `Group-Ring-Module` partially fits a quadratic model, which can be explained by the hierarchical nature of the proofs. Although `JinjaThreads` and `Group-Ring-Module` both fit a quadratic model, the quadratic coefficient on their regression lines differ by an order of magnitude. `Psi_Calculi`, does not fit the same relationship. There is a column of data points at a statement size of about 100 (or about 60 idealized), indicating that most lemmas in the development actually have the same statement size. This can be explained by the fact that `Psi_Calculi` is a language formalization. Each lemma mentions the inductive set which defines the language semantics, and the size of that constant dominates the statement size of the lemma. This indicates that this measure of lemma statement size is too coarse for `Psi_Calculi`: no model built against this data will be able to discriminate between long and short proofs.

Table 1. R^2 and coefficients a, b, c for quadratic regression with equation ax^2+bx+c statement size versus proof size

AFP article	Measure	R^2	a	b	c
`JinjaThreads`	Raw	0.346	0.04	10.04	287.22
	Idealized	0.712	0.12	16.48	283.49
`Group-Ring-Module`	Raw	0.487	1.29	29.20	154.81
	Idealized	0.622	2.26	20.56	58.25
`Psi_Calculi`	Raw	0.349	0.85	69.21	609.56
	Idealized	0.431	4.87	198.49	798.34

The results are similarly diverse when performing this analysis against the entire AFP. Using the idealized measure, approximately half of the articles have an R^2 of less than 0.5 (or have too few data points to build a model), 50 have an R^2 between 0.5, and 0.7 and 50 have an R^2 greater than 0.7. The best fitting articles are primarily those related to software verification, e.g. `JinjaThreads`, `SATSolverVerification`, `DiskPaxos`. Among these, the quadratic coefficients span two orders of magnitude, from 0.07 to 21.29. The variation in the consistency of this relationship, and the differences between the models, demonstrates proof size cannot be estimated based solely on this coarse measure of lemma statement size. It is, however, an indication that it could be used as part of a more sophisticated measure, which considers the particular domain of the proof as well as additional measures of lemma statement complexity, such as those discussed in the following section.

5 Lemma Statement Complexity

We are interested in the complexity of the 64,497 lemma statements, using more traditional metrics than in the previous section, such as clause and literal counts. We used Isabelle's clausifier to rewrite formulas into conjunctive normal form (CNF), i.e. as a conjunction of clauses, each of which is a disjunction of literals. Literals of the form $\neg\, a$ are negative; otherwise, they are positive.

Out of the AFP's 64,497 lemmas, the clausifier times out or fails on 171 of them. These are completely excluded from the statistics below. A manual inspection reveals that these are typically highly complex formulas, such as custom induction schemas.

As measures of formula complexity, Fig. 11 gives the number of clauses per lemma, and Fig. 12 gives the number of literals per lemma.

A few lemmas give rise to zero clauses: these are typically simple tautologies identified as such by the clausifier. Most formulas give rise to exactly one clause. These are further classified as follows:

- 11,444 formulas correspond to unit equality clauses, i.e., simple equations of the form $t = u$ for two terms t, u.
- 22,475 formulas correspond to conditional equality clauses, i.e., clauses that contain at least one positive literal of the form $t = u$.

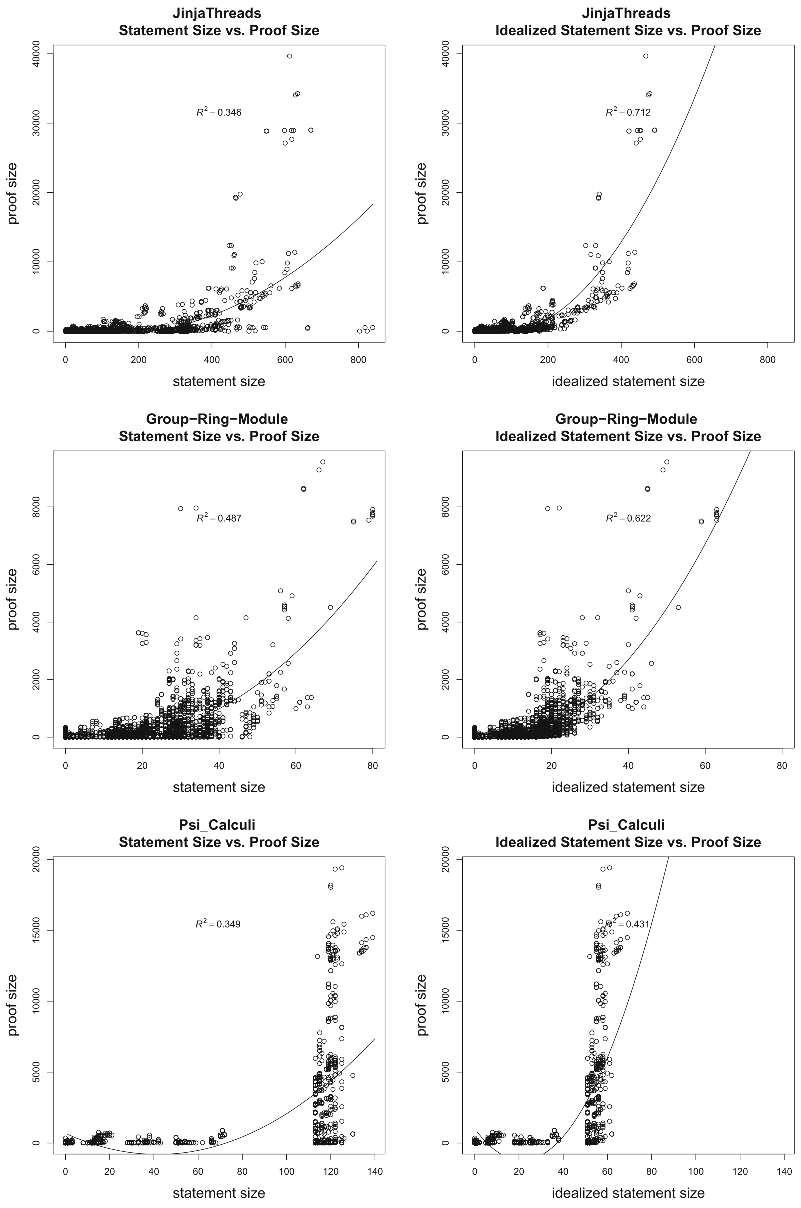

Fig. 10. Relation between statement size and proof size

– 46,186 formulas correspond to Horn clauses, i.e., clauses that contain at most one positive literal. These can be seen as implications $a_1 \wedge \cdots \wedge a_n \Longrightarrow a$ or $a_1 \wedge \cdots \wedge a_n \Longrightarrow$ `False` and are relatively easy to reason automatically about.

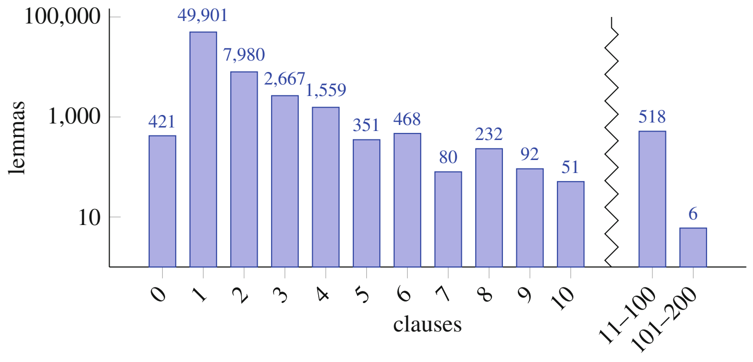

Fig. 11. Number of lemmas with a corresponding number of clauses

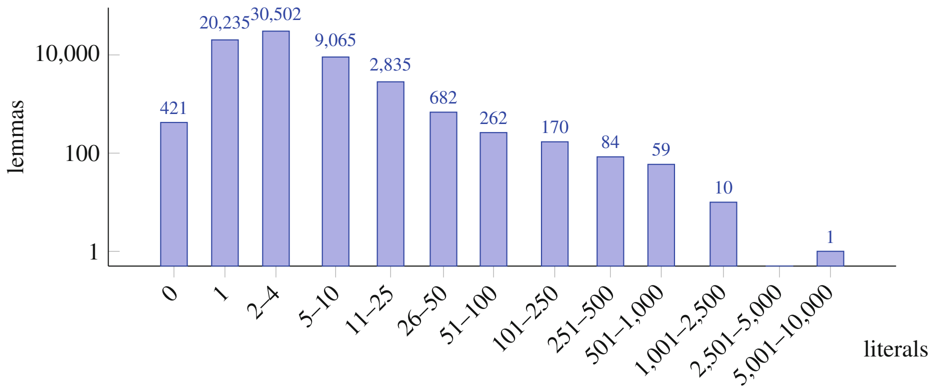

Fig. 12. Number of lemmas with a corresponding number of literals

6 Proof Automation with Sledgehammer

Sledgehammer is a proof tool for Isabelle that exploits powerful first-order automatic theorem provers, notably E [24], SPASS [28], Vampire [23], and Z3 [10]. Given a proof goal, it heuristically selects a few hundred potentially relevant lemmas, invokes the external provers, and upon success produces a proof snippet that can be inserted in the user's formalization to discharge the goal. Similar tools are available for other proof assistants, notably Miz$\mathbb{AR}$ [27] for Mizar and HOLyHammer [13] for HOL Light.

Sledgehammer was introduced in the 2007 edition of Isabelle and started to be used seriously in 2009. In their "Judgement Day" study from 2010, Böhme and Nipkow [6] evaluated Sledgehammer on 1240 subgoals emerging from seven theory representing various applications of Isabelle to computer science and mathematics. They reported a success rate of 46 % for three provers (E, SPASS, and Vampire) run in parallel for 30 s, meaning that nearly half of the goals in these seven theories could be discharged automatically, with no user guidance.

The tool has been further improved since then. Moreover, the automatic provers that form its back-ends have themselves undergone major development. A recent evaluation using a preliminary version of Isabelle2015 finds a success rate of 75 % for six provers run for 30 s on newer hardware [5], for the Judgement Day benchmarks. The success rate rise can be tracked in the various papers on Sledgehammer.

One question that has lingered since 2010 is whether Judgement Day is really representative of Isabelle formalizations. This is especially an issue since Sledgehammer has been extensively tuned against the benchmark set, on the assumption that it is representative. To get a clearer idea of Sledgehammer's usefulness, we now ran an evaluation on 128 randomly selected theories from the AFP. Up to 100 goals were selected for each theory, for a total of 6,934 goals. Our evaluation data is publicly available.[1]

The evaluation harness invokes Sledgehammer on each goal. The hardware setup consists of Linux servers equipped with Intel Core2 Duo CPUs running at 2.40 GHz. Each prover was given 30 s to solve each goal, but the 30 s slot was split into several slices, each corresponding to different problems and options to the prover. Lemmas were selected using the static MePo filter [19], as opposed to the machine learning based MaSh [5], whose development has fully stabilized only after the Isabelle2014 release. The results are summarized in Fig. 13.

As we remarked elsewhere [3], "It is important to bear in mind that the evaluation is not a competition between the provers. Different provers are invoked with different problems and options, and although we have tried to optimize the setup for each, we might have missed an important configuration option. Each number must be seen as a lower bound on the potential of the prover."

In case of success, the search is followed by reconstruction in Isabelle. For most goals, the reconstructed proof is a one-line call to an Isabelle proof method, such as *simp* (term rewriting), *metis* (a built-in resolution prover), or *blast* (a tableau prover). This call can then be inserted in the Isabelle formalization to discharge the goal. Reconstruction is a success if at least one of the attempted proof methods succeeds within 2 s. The percentage of goals with successful one-line proofs is given in the "One-line" column of Fig. 13. A few goals require a more detailed proof, expressed in the Isar format. The goal is considered solved if the Isar proof is successfully generated and replayed. This is reflected in the "+ Isar" column of Fig. 13.

	One-line	+ Isar	+ Oracle
E	49.7	51.4	52.5
SPASS	49.4	50.5	52.0
Vampire	49.5	51.0	51.8
Z3	49.6	50.0	53.7

Fig. 13. Success rate of Sledgehammer invocations per automatic prover (%)

[1] http://www21.in.tum.de/~blanchet/afp_mining_data.tgz.

When both reconstruction approaches fail, the user can still trust the external prover (and Sledgehammer's translation from HOL to the prover's formalism) as an oracle. In practice, most Isabelle users would prefer to work on a manual proof instead. These reconstruction failures are recorded in the "+ Oracle" column of Fig. 13.

The provers are neck and neck. Trusted as oracles, they prove 60.7 % of the goals when used in combination. This is significantly lower than the most recent evaluations based on Judgement Day. We offer the following possible explanations:

- Our evaluation uses the official release (Isabelle2014), instead of a preliminary version of Isabelle2015. It misses out on MaSh [5], on the improved Isar proof generation module [3], and on modern versions of provers. Recently, the SMT solver CVC4 has been integrated with Isabelle and is now, by a clear margin, the most successful prover [5].
- Sledgehammer's development since 2010 has been guided by experimental results on the Judgement Day suite, under the assumption that it is representative of Isabelle. Hence, it is not surprising that Sledgehammer should perform particularly well on these benchmarks.
- There is a lot of variation between theories. For theories with at least ten goals, our evaluation found success rates varying between 10 % and 100 %. We cannot exclude that the seven Judgement Day theories are particularly easy for Sledgehammer. Indeed, one third of Judgement Day consists of a large mathematical theory (`Fundamental_Theorem_Algebra`) whose goals are particularly easy.

7 Conclusion

We can summarize our findings by answering the questions raised in the introduction:

- There is too little reuse to our taste: the top 3 articles are reused 9, 6, and 4 times.
- There is some similarity to citation graphs in the computer science literature: the largest weakly connected component (WCC) in the AFP imports graph is 10 times larger than the next smaller WCC.
- The growth of the AFP appears to be only slightly better than linear although we hope it is only early days.
- The sizes of articles seem to follow a power law with a long tail. Two thirds are less than 4,000 lines long but 9 are longer than 20,000 lines.
- Over the whole AFP, 58 % of the text is taken up by proofs, 19 % by lemma statements, and 8 % by definitions.
- Lemma statement size is quadratically related to lemma proof size in approximately half of the AFP articles (with $R^2 > 0.5$). The exact nature of the relationship is not consistent, however. A more sophisticated measure for statement size/complexity is required to build a predictive model.

- The syntactic nesting depth of proofs follows an exponential decay. Almost 99 % of all proofs have a depth of 4 or less, but there is one proof of depth 15.
- One third of all lemma statements are equations, two thirds are Horn clauses.
- Sledgehammer can automate the proof of about 60 % of the goals that arise, which is respectable but less than on the Judgement Day benchmark suite.

Acknowledgement. Our colleague Johannes Hölzl suggested the proof depth diagram (Fig. 5), which we found insightful. Matichuk is partially supported by NICTA. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. Nipkow is supported by DFG grant NI 491/16-1.

References

1. The Mizar mathematical library. http://mizar.org
2. An, Y., Janssen, J., Milios, E.: Characterizing and mining citation graphs of the computer science literature. Knowl. Inf. Syst. **6**, 664–678 (2004)
3. Blanchette, J.C., Böhme, S., Fleury, M., Smolka, S.J., Steckermeier, A.: Semi-intelligible Isar proofs from machine-generated proofs. Accepted in J. Autom. Reason. http://www21.in.tum.de/~blanchet/isar2.pdf
4. Blanchette, J.C., Böhme, S., Paulson, L.C.: Extending Sledgehammer with SMT solvers. J. Autom. Reason. **51**(1), 109–128 (2013)
5. Blanchette, J.C., Greenaway, D., Kaliszyk, C., Kühlwein, D., Urban, J.: A learning-based relevance filter for Isabelle/HOL (2015) (Submitted). http://www21.in.tum.de/~blanchet/mash2.pdf
6. Böhme, S., Nipkow, T.: Sledgehammer: judgement day. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 107–121. Springer, Heidelberg (2010)
7. Chaudhuri, K., Doligez, D., Lamport, L., Merz, S.: The TLA$^+$ proof system: building a heterogeneous verification platform. In: Cavalcanti, A., Deharbe, D., Gaudel, M.-C., Woodcock, J. (eds.) ICTAC 2010. LNCS, vol. 6255, p. 44. Springer, Heidelberg (2010)
8. Church, A.: A formulation of the simple theory of types. J. Symb. Logic **5**(2), 56–68 (1940)
9. Crovella, M.E., Bestavros, A.: Self-similarity in world wide web traffic: evidence and possible causes. IEEE/ACM Trans. Network. **5**(6), 835–846 (1997)
10. de Moura, L., Bjørner, N.S.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
11. Gonthier, G., et al.: A machine-checked proof of the odd order theorem. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) ITP 2013. LNCS, vol. 7998, pp. 163–179. Springer, Heidelberg (2013)
12. Hölzl, J., Immler, F., Huffman, B.: Type classes and filters for mathematical analysis in Isabelle/HOL. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) ITP 2013. LNCS, vol. 7998, pp. 279–294. Springer, Heidelberg (2013)
13. Kaliszyk, C., Urban, J.: HOL(y)Hammer: online ATP service for HOL light. Math. Comput. Sci. **9**(1), 5–22 (2015)
14. Lammich, P., Lochbihler, A.: The Isabelle collections framework. In: Kaufmann, M., Paulson, L.C. (eds.) ITP 2010. LNCS, vol. 6172, pp. 339–354. Springer, Heidelberg (2010)

15. Leroy, X.: A formally verified compiler back-end. J. Autom. Reason. **43**, 363–446 (2009)
16. Lochbihler, A.: Java and the Java memory model — a unified, machine-checked formalisation. In: Seidl, H. (ed.) ESOP 2012. LNCS, vol. 7211, pp. 497–517. Springer, Heidelberg (2012)
17. Matichuk, D., Murray, T., Andronick, J., Jeffery, R., Klein, G., Staples, M.: Empirical study towards a leading indicator for cost of formal software verification. In: Canfora, G., Elbaum, S. (eds.) International Conference on Software Engineering (ICSE 2015). ACM (2015)
18. Matuszewski, R., Rudnicki, P.: Mizar: the first 30 years. Mech. Math. Appl. **4**(1), 3–24 (2005)
19. Meng, J., Paulson, L.C.: Lightweight relevance filtering for machine-generated resolution problems. J. Appl. Logic **7**(1), 41–57 (2009)
20. Nipkow, T., Klein, G.: Concrete Semantics with Isabelle/HOL. Springer, Heidelberg (2014). http://concrete-semantics.org
21. Nipkow, T., Paulson, L.C., Wenzel, M. (eds.): Isabelle/HOL. LNCS, vol. 2283. Springer, Heidelberg (2002)
22. Paulson, L.C., Blanchette, J.C.: Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In: Sutcliffe, G., Schulz, S., Ternovska, E. (eds.) International Workshop on the Implementation of Logics (IWIL 2010). EPiC Series, vol. 2, pp. 1–11. EasyChair (2012)
23. Riazanov, A., Voronkov, A.: The design and implementation of Vampire. AI Commun. **15**(2–3), 91–110 (2002)
24. Schulz, S.: System description: E 1.8. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR-19 2013. LNCS, vol. 8312, pp. 735–743. Springer, Heidelberg (2013)
25. Staples, M., Jeffery, R., Andronick, J., Murray, T., Klein, G., Kolanski, R.: Productivity for proof engineering. In: Morisio, M., Dybå, T., Torchiano, M. (eds.) Empirical Software Engineering and Measurement (ESEM 2014), pp. 15:1–15:4. ACM, New York (2014)
26. Urban, J.: MPTP 0.2: design, implementation, and initial experiments. J. Autom. Reason. **37**(1–2), 21–43 (2006)
27. Urban, J., Rudnicki, P., Sutcliffe, G.: ATP and presentation service for Mizar formalizations. J. Autom. Reason. **50**(2), 229–241 (2013)
28. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Schmidt, R.A. (ed.) CADE-22. LNCS, vol. 5663, pp. 140–145. Springer, Heidelberg (2009)
29. Wenzel, M.: Isabelle/Isar—a versatile environment for human-readable formal proof documents. Ph.D. thesis, Institut für Informatik, Technische Universität München (2002). http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2002/wenzel.html
30. Wiedijk, F.: Statistics on digital libraries of mathematics. Stud. Logic, Gramm. Rhetor. **18**(31), 137–151 (2009)