ERC Advanced Grant 2016

Annex 1 to the Grant Agreement (Description of the Action) Part B

Proposal Acronym: ALEXANDRIA
Proposal number: GA 742178
Proposal Title: Large-Scale Formal Proof for the Working Mathematician
Principal Investigator: Lawrence Paulson
Host Institution: University of Cambridge

Contents

1.a. Summary

Mathematical proofs have always been prone to error. Today, proofs can be hundreds of pages long and combine results from many specialisms, making them almost impossible to check. One solution is to deploy modern verification technology. Interactive theorem provers have demonstrated their potential as vehicles for formalising mathematics through achievements such as the verification of the Kepler Conjecture. Proofs done using such tools reach a high standard of correctness.

However, existing theorem provers are unsuitable for mathematics. Their formal proofs are unreadable. They struggle to do simple tasks, such as evaluating limits. They lack much basic mathematics, and the material they do have is difficult to locate and apply.

ALEXANDRIA will create a proof development environment attractive to working mathematicians, utilising the best technology available across computer science. Its focus will be the management and use of large-scale mathematical knowledge, both theorems and algorithms. The project will employ mathematicians to investigate the formalisation of mathematics in practice. Our already substantial formalised libraries will serve as the starting point. They will be extended and annotated to support sophisticated searches. Techniques will be borrowed from machine learning, information retrieval and natural language processing. Algorithms will be treated similarly: ALEXANDRIA will help users find and invoke the proof methods and algorithms appropriate for the task.

ALEXANDRIA will provide (1) comprehensive formal mathematical libraries; (2) search within libraries, and the mining of libraries for proof patterns; (3) automated support for the construction of large formal proofs; (4) sound and practical computer algebra tools.

ALEXANDRIA will be based on legible structured proofs. Formal proofs should be not mere code, but a machine-checkable form of communication between mathematicians.

1.b. Curriculum vitae

Lawrence Paulson is Professor of Computational Logic at the University of Cambridge, where he has worked since 1982. In the early 1980s, Paulson worked alongside Michael J. C. Gordon and Gérard Huet. These collaborations made a great impact on the development of the ML family of languages — Paulson wrote the highly successful textbook *ML for the Working Programmer* — and on several interactive theorem provers. Chief among these is HOL, which in several versions remains in wide use today. In 1986, Paulson created Isabelle, now one of the world's most popular interactive theorem provers. (Isabelle is now developed jointly with Tobias Nipkow's group at TU Munich, with contributions from many international colleagues.) Paulson has applied Isabelle to the formalisation of deep theorems in set theory and logic, and on a more practical note, to the verification of security protocols.

Paulson's contributions to Isabelle and HOL make him one of the world's most significant figures in interactive theorem proving. He has also contributed to fully automatic theorem proving and to a variety of applications, such as computer security. According to Google Scholar, Paulson has approx. 14,400 citations and an h-index of 49. Counting only papers published since 2011, his h-index is 26. Twenty-two papers have 100 citations or more.

PERSONAL INFORMATION

Family name, First name:	PAULSON, Lawrence Charles
Researcher unique identifiers:	ORCiD: 0000-0003-0288-4279
	Scopus Author ID: 7005293178
Nationality:	UK and USA
Date of birth:	20 September 1955
URL for web site:	`http://www.cl.cam.ac.uk/~lp15/`

EDUCATION

1977–81	PhD (Computer Science). Stanford University, Stanford, California. Topic: compiler generation from denotational semantics. Advisor: John Hennessy
1973–77	BS (Mathematics). California Institute of Technology, Pasadena, California

CURRENT POSITIONS

2002–	*Professor of Computational Logic*, University of Cambridge, UK
1987–	*Fellow* and *Director of Studies*, Clare College, Cambridge

PREVIOUS POSITIONS

1998–2002	*Reader in Computational Logic*, University of Cambridge
1993–98	*University Lecturer*, University of Cambridge
1983–93	*Assistant Director of Research*, University of Cambridge
1982–83	*Research Assistant*, University of Edinburgh, UK

FELLOWSHIPS AND AWARDS

2008	*ACM Fellow*. "For contributions to theorem provers and verification techniques"
2006	*Distinguished Affiliated Professor* of the Department of Informatics at the Technical University of Munich
2003	*Pilkington Teaching Prize*, University of Cambridge

SUPERVISION OF GRADUATE STUDENTS & POSTDOCTORAL FELLOWS

1983–	Supervised 19 Cambridge students to their PhD, of whom two (Andrew D. Gordon and Jacques Fleuriot) won national competitions for their dissertations
1983–	Supervised 15 postdoctoral researchers and 13 medium-term research visitors

TEACHING ACTIVITIES

1983–	University of Cambridge: Many lecture courses, including Logic and Proof, Interactive Formal Verification, Foundations of Computer Science, Programming in LISP, Software Engineering
1987–	Clare College: Small-group teaching in a variety of subjects

INSTITUTIONAL RESPONSIBILITIES

1983–	University of Cambridge: Chairman or member of committees for graduate and undergraduate teaching, promotions, student liaison, etc.; member of Faculty Board
1987–	Clare College: Various committee memberships including Governing Body

COMMISSIONS OF TRUST

2014–	*Reviewer*, ERC Starting Grants
2003–	*Member*, EPSRC Peer Review College (reviewing research proposals for the UK)
2008–	*Reviewer*, Qatar National Research Fund
1997–	*Editorial Board*, *J. Automated Reasoning*
2010–	*Trustee*, *Conference on Automated Deduction* (CADE)
2007–	*Trustee*, conference on *Interactive Theorem Proving* (ITP)
2013	*External research assessor*, University of Hertfordshire, UK
2010	*Programme committee co-chair* (with Matt Kaufmann), conference on Interactive Theorem Proving, Edinburgh, UK. 114 attendees.
2007–10	*Member*, Royal Society International Grants Panel
1998–2007	*Founding editor* of London Mathematical Society *J. Computation and Mathematics*

MEMBERSHIPS OF SCIENTIFIC SOCIETIES

1996–	ACM (Fellow since 2008)
1998–	London Mathematical Society

MAJOR COLLABORATIONS

2006–	With Christoph Benzmüller, Free University of Berlin, Germany, on the LEO-II theorem prover and its applications.
1989–	With Tobias Nipkow, Technical University of Munich, Germany. This collaboration continues to develop the Isabelle system.
1982–85	With Gérard Huet, Inria Rocquencourt, France. This collaboration contributed to the theorem provers HOL88 and Coq, and to the programming language ML.

CAREER BREAKS

2009–11	His first wife's diagnosis of terminal cancer (she died in June 2010) caused a near-cessation of research for 18 months, with a normal rate of publication not resuming until 2013.

Appendix: All grants and funding of the PI (Funding ID)

No other funding currently.

1.c. Early achievements track-record

Paulson has investigated a wide variety of topics during his career, from building theorem provers to constructive type theory to formalising set theory. In the past 10 years, his research has concentrated in three main areas:

- *Cryptographic protocol verification.* Paulson's paper "The inductive approach to verifying cryptographic protocols", published in 1998 and cited more than 1100 times, introduced the first sound and practical method for proving security protocols correct using ordinary logic. The approach is based on a formalisation of the operational semantics that's realistically detailed but also gives a readable specification. Paulson continued this work with the verifications of the gigantic SET protocol suite and the Zhou-Gollmann non-repudiation protocol. He has also treated protocols involving a variable number of participants and secret sharing.[1] (Papers 1 and 2 below.)
- *Integrating automatic theorem provers with Isabelle.* Paulson led a team to build the first useful integration between automatic and interactive theorem provers. The project required solving several distinct problems and demanded extensive empirical tuning to achieve acceptable results. Previous work with this aim did not have a lasting impact, but Isabelle's *Sledgehammer* can genuinely be called revolutionary. An Isabelle user can now invoke powerful external provers at any time, with lemma selection and translation done automatically. One sign of its dramatic impact on the research community is the emergence of workshops with titles such as "Hammers for Type Theories".[2] (Papers 3–6 below.)
- *Automatic theorem proving for the real numbers.* Paulson led a team to develop a wholly original type of resolution theorem prover, *MetiTarski.* Combining a standard resolution prover (Metis) with a decision procedure for nonlinear real arithmetic, MetiTarski automatically proves theorems involving transcendental functions such as log, exp, sin and cos. Difficult problems involving multiple occurrences of special functions can be solved in seconds, and MetiTarski delivers explicit, machine-readable proofs.[3] MetiTarski has hundreds of registered users and is now used as a component of a number of other verification tools, including KeYmaera, PVS and Why3. Thus it is used in software verification and to verify hybrid systems. (Papers 7 and 8 below.)

All three of these projects demonstrate ambitious, complex challenges being overcome through leadership, originality and persistence. The last two projects also delivered substantial pieces of software.

[1]Funded by the EPSRC, grant reference GR/R01156/01, period 10/2000–10/2003, but work continued afterwards.
[2]EPSRC grant GR/S57198/01 from 2004–2007.
[3]EPSRC grants EP/C013409/1 from 2005–2008 and EP/I011005/1 from 2010–2014.

The 10 papers shown below mostly fall in one of these three areas. Paper 9 emerged from one of the two Ph.D. dissertations on machine learning. Paper 10 reports the first ever machine formalisation of Gödel's second incompleteness theorem; this solo project reached a goal that many had thought to be unattainable. In all ten papers, Paulson contributed substantially to both the text and to the research itself.

From 2006 to 2009, Paulson held two other EPSRC grants (EP/D070511/1 and EP/G002290/1) delivering work done mainly by postdoctoral researchers. Although significant — among other things, they delivered LEO-II, an automatic theorem prover for higher-order logic — those papers are not listed below. Paulson has published a further nine refereed articles since 2014, mostly jointly with colleagues or students. In recent work, he has translated substantial portions of the HOL Light library for multivariate analysis in support of new work on decision procedures for nonlinear arithmetic.

1. Giampaolo Bella, Fabio Massacci and L. C. Paulson. Verifying the SET purchase protocols. *J. Automated Reasoning* **36** (2006), 5–37. [*51 citations*]
2. Giampaolo Bella and L. C. Paulson. Accountability protocols: formalized and verified. *ACM Trans. Information and System Security* **9** 2 (2006), 138–161. [*43 citations*]
3. Jia Meng, Claire Quigley and L. C. Paulson. Automation for interactive proof: first prototype. *Information and Computation* **204** 10 (2006), 1575–1596. [*60 citations*]
4. L. C. Paulson and Kong Woei Susanto. Source-level proof reconstruction for interactive theorem proving. *In*: Klaus Schneider and Jens Brandt (editors), *Theorem Proving in Higher Order Logics* (Springer LNCS 4732, 2007), 232–245. [*61 citations*]
5. Jia Meng and L. C. Paulson. Translating higher-order clauses to first-order clauses. *J. Automated Reasoning* **40** 1 (2008), 35–60. [*110 citations*]
6. Jia Meng and L. C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *J. Applied Logic* **7** 1 (2009), 41–57. [*89 citations*]
7. Behzad Akbarpour and Lawrence C. Paulson. MetiTarski: An automatic theorem prover for real-valued special functions. *J. Automated Reasoning* **44** 3 (2010), 175–205. [*87 citations*]
8. James P. Bridge and Lawrence C. Paulson. Case splitting in an automatic theorem prover for real-valued special functions. *J. Automated Reasoning* **50** 1 (2013), 99–117.
9. James P. Bridge, Sean B. Holden and L. C. Paulson. Machine learning for first-order theorem proving: learning to select a good heuristic. *J. Automated Reasoning* **53** 2 (2014), 141–172.
10. L. C. Paulson. A mechanised proof of Gödel's incompleteness theorems using Nominal Isabelle. *J. Automated Reasoning* **55** 1 (2015), 1–37.

Personal circumstances made travel difficult for many years, but Paulson has been increasingly available for invited lectures starting in 2012.

1. LCF + logical frameworks = Isabelle (25 years later). *Milner Symposium.* Edinburgh, April 2012.
2. Overcoming intractable complexity in an automatic theorem prover for real-valued functions. *Computability and Complexity in Analysis.* Cambridge, UK, June 2012.
3. MetiTarski: past and future. *Interactive Theorem Proving — ITP 2012*, Princeton, New Jersey (Springer LNCS 7406, 2012), 1–10.
4. Isabelle/HOL tutorial: verifying functional programs and inductively defined sets. *Relational and Algebraic Methods in Computer Science.* Cambridge, UK, September 2012.
5. MetiTarski's menagerie of cooperating systems. Plenary Lecture for the *FroCoS* and *TABLEAUX* conferences. Nancy, France, September 2013.

6. Theorem proving and the real numbers: overview and challenges. Plenary Lecture for *NASA Formal Methods*. Houston, Texas, April 2014.
7. Automated theorem proving for special functions: the next phase. *Symbolic Numeric Computation*. Shanghai, China, July 2014.
8. The Future of Formalised Mathematics. Keynote Lecture for EACA 2016 (XV Encuentro de *Álgebra Computacional y Aplicaciones* [Computer Álgebra and Applications]). Logroño, Spain, June 2016.

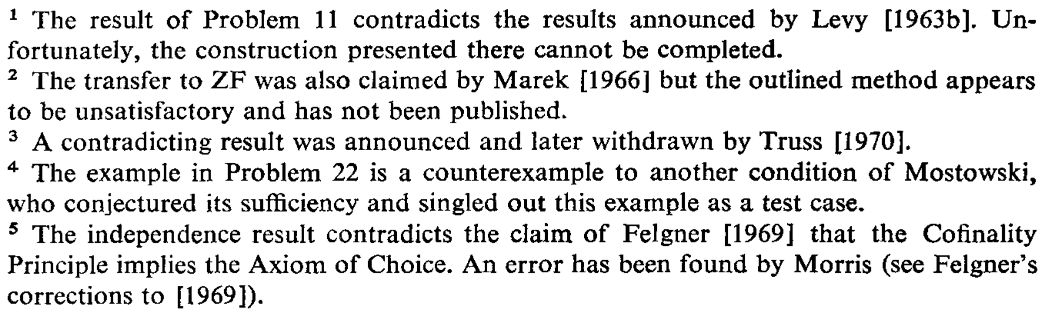
Paulson is a successful mentor of postdocs and students at all levels, many of whom now occupy senior positions. Andrew Gordon is now a Principal Researcher at Microsoft. David Aspinall, Giampaolo Bella, Jacques Fleuriot, Mateja Jamnik, Sara Kalvala and Florian Kammueller have academic positions at European universities, including Cambridge and Edinburgh. Other former colleagues are working in industry: Behzad Akbarpour, for NVIDIA; Jia Meng, for Google; Kong Susanto, for Cadence. Many others could be listed.

Paulson has served for many years on the programme committees of numerous international conferences and workshops, and on various editorial boards. He serves on the steering committees of the two main conferences in his field: ITP (Interactive Theorem Proving, formerly TPHOLs) and CADE (Conference on Automated Deduction). He was programme committee co-chair of ITP in 2010. He was elected an ACM Fellow in 2008 "for contributions to theorem provers and verification techniques".

2.a. State-of-the-art and Objectives

The Crisis in Modern Mathematics

Mathematicians are fallible. This point can be grasped by looking at the footnotes on a single page (118) of Jech's *The Axiom of Choice* [49]:


^{1} The result of Problem 11 contradicts the results announced by Levy [1963b]. Unfortunately, the construction presented there cannot be completed.
^{2} The transfer to ZF was also claimed by Marek [1966] but the outlined method appears to be unsatisfactory and has not been published.
^{3} A contradicting result was announced and later withdrawn by Truss [1970].
^{4} The example in Problem 22 is a counterexample to another condition of Mostowski, who conjectured its sufficiency and singled out this example as a test case.
^{5} The independence result contradicts the claim of Felgner [1969] that the Cofinality Principle implies the Axiom of Choice. An error has been found by Morris (see Felgner's corrections to [1969]).

Even the greatest are not immune: "When the Germans were planning to publish Hilbert's collected papers and to present him with a set on the occasion of one of his later birthdays, they realized that they could not publish the papers in their original versions because they were full of errors, some of them quite serious. Thereupon they hired a young unemployed mathematician, Olga Taussky-Todd, to go over Hilbert's papers and correct all mistakes. Olga laboured for three years." [63, p. 201].

Errors have always been present in mathematical works. Today, the crisis is out in the open: "And who would ensure that I did not forget something and did not make a mistake, if even the mistakes in much more simple arguments take years to uncover?" Thus Voevodsky [65], a Fields medallist, called for computer-based proof verification, setting off a frenzy of activity. While his approach (homotopy type theory) involves a new foundation for mathematics, ALEXANDRIA will look at the problem from the opposite end: **large-scale mathematical libraries**.

Specifically, ALEXANDRIA is concerned with questions such as these:

- How do we organise libraries of formal mathematics so that they communicate ideas?
- What sort of *proof idioms* — legible proof fragments — can be extracted from libraries?
- What support do mathematicians most need when undertaking large formal proofs?
- What about mathematical knowledge encoded in the form of algorithms?

Let's examine the background of all these issues.

The Interconnectedness of Mathematical Knowledge

Fermat's Last Theorem is concerned with positive integers, but its proof involves exotic objects (elliptic curves and modular forms) unimaginable to Fermat. This situation is typical: the Prime Number Theorem is most easily proved using complex analysis; topological notions find their way into set theory and computation theory; combinatorics finds its way into set theory, algebra and the theory of program termination. Today's mathematician needs to be able to apply knowledge from any field of mathematics, even one in which they are not expert. This magnifies the risk of making a mistake.

New foundations can only be part of the solution. Throughout the history of mathematics, standards of rigour have continually increased. From Newton to Cauchy, from Dedekind to Hilbert, works once regarded as genius are regularly re-evaluated as naive, the original proofs all discarded. No mathematics student studies the great works of Euclid or Newton any more. And so, work on foundations will never end. We need to build the great mathematical edifice, to live in it and work in it, knowing that it will have to be moved to new foundations, not once but repeatedly. ALEXANDRIA is about the edifice as a whole, and ultimately about moving it too.

ALEXANDRIA is about managing large bodies of formal mathematical knowledge. Instead of trying to find some ideal universal language, it recognises that, while new formalisms emerge all the time, the underlying mathematical ideas and dependencies remain the same. Meanwhile, we need support to use all this formal knowledge: sophisticated search mechanisms, mining to identify common reasoning patterns that can be suggested to the user working on a similar proof, support for verified computations, and other measures useful to research mathematics. ALEXANDRIA will elicit its priorities by **working with mathematicians themselves**.

Interactive Theorem Provers

Interactive theorem provers, such as Coq, HOL and Isabelle, are software tools that carry out formal mathematical and logical reasoning. They allow users to develop elaborate specification hierarchies. Types and functions can be defined using concepts drawn from functional programming but also classical mathematics and set theory, or alternatively, constructive type theory.

The original motivation for this technology was to verify computer systems used in railway signalling, air traffic control, medical or other safety-critical applications. Interactive theorem provers can be used to formalise the specification and implementation of a critical system. Correctness properties can then be proved rigorously using the rules of formal logic.

Until 10 years ago, formal verification was prohibitively laborious. But immense progress has been made. Sledgehammer, in particular, brings the full power of multiple external reasoning tools and machine learning to the problem at hand. Users now enjoy substantial automation. Impressive recent achievements include the seL4 operating system kernel [50] (verified using Isabelle) and CompCert C compiler [13] (verified using Coq).

How Mathematicians Use Computers

Mathematicians have long used computers as exploratory tools, but they have often regarded computer calculations with distrust. A noteworthy example is the Robbins conjecture, which dates from the 1930s and asks whether every Robbins algebra is Boolean. In 1996, McCune settled the question affirmatively using EQP, a specialised computer program for equational reasoning. His machine proof was later written out in the traditional style and published in the *Journal of Algebra* [20]. The irony is that the published proof was much more likely to contain errors than the original machine proof.

Probably the best-known example of proof by computer is Appel and Haken's [2] solution of the Four Colour Theorem in 1976. The use of a computer program to check thousands of configurations compromised the proof to many observers, as the correctness of the code could not be guaranteed. Other mathematicians simplified the proof and reduced the number of configurations to be checked, but could not eliminate the need for a computer to check them.

A similar situation arose with Hales's 1998 proof [32] of the Kepler Conjecture, a 400-year-old problem concerning the optimal packing of spheres. Part of Hilbert's 18th Problem, it was of great importance to mathematicians. And yet, no conventional proof was on offer:

> As this project has progressed, the computer has replaced conventional mathematical arguments more and more, until now nearly every aspect of the proof relies on computer verifications. Many assertions in these papers are results of computer calculations. [32, p. 11]

Once again, the use of a computer to check a large number of cases left referees unsure about the proof's validity, and the proof itself was also intricate. Fortunately, by this time, interactive theorem provers were becoming available.

Mathematicians can trust interactive theorem provers because they are designed with sound reasoning as the supreme priority. Many adopt the *LCF architecture* [55, p. 293], delegating all reasoning to a small kernel of trusted code, minimising the possibility of error even at the expense of performance. They have been used to verify their own correctness [37] down to the level of assembly language code [51]. ALEXANDRIA will stress an additional basis for trust: it will focus on proofs that are human-readable as well as machine-readable, and on the communication of mathematical ideas as well as formal symbol strings. A sceptical mathematician will be able to follow an argument simply by **reading the formal proof text**.

Mathematics in Interactive Theorem Proving

Safety-critical systems frequently operate in real-world environments that can only be modelled using advanced mathematics, such as information theory or probability. One mathematical concept typically depends on a chain of others: for example, Markov chains involve probability theory, which is defined in terms of Lebsegue integration and measure theory, depending on further concepts such as Borel spaces and topology [45, 57]. Moreover, a formal proof may require algebraic calculations such as multiplication and factoring of polynomials, differentiation or integration of a formula, or isolating the roots of a polynomial.

Researchers generally reject the soft option of simply asserting well-known mathematical theorems as axioms. Although such theorems are almost certainly correct in some form, mathematics textbooks frequently make unstated assumptions. The precise statement of the theorem may be unclear. Researchers prefer to formalise and verify the mathematics necessary for their verification task. Others have tried from the outset to apply their tools to research mathematics.

An early milestone is Gonthier's formalisation of the Four Colour Theorem [29], which addressed the doubts about Appel and Haken's proof. Gonthier's proof still checked many configurations by computer, but now they were being checked by a formally verified program running within Coq. A similar situation arose with Hales's proof of the Kepler Conjecture. Hales organised a global effort, the Flyspeck project [34], to formalise his proof. The argument was formalised using HOL Light, while the calculations were confirmed using Isabelle [60]. Through these efforts, the proof was not merely confirmed but simplified [35]. ALEXANDRIA will provide **an environment for undertaking large proofs and verified computations**.

Advances in the Formalisation of Mathematics

A string of recent achievements demonstrate that much of modern mathematics falls within the scope of existing verification tools [25]. The most striking of these is the formalisation of the Odd Order Theorem [30], a deep result about finite groups. Feit and Thompson's original proof (published in 1963) was 255 pages long, justifying a formal treatment. The Coq proof required the formalisation of a wide variety of mathematical topics. Other examples include the Prime Number Theorem [40] (formalised independently using Isabelle and HOL Light) and the Central Limit Theorem [3] (in Isabelle). Also noteworthy is Paulson's recent formalisation of Gödel's Second Incompleteness Theorem, which yielded new insights [62] into a highly technical proof that is seldom presented rigorously.

Isabelle's Archive of Formal Proofs[4] contains an immense amount of material from every branch of mathematics and theoretical computer science. The AFP comprises nearly 300 entries contributed by Isabelle users, with about 80,000 named theorems and 1.4 million lines of proofs. This includes much core mathematics: linear algebra, vector spaces and matrix theory, multivariate analysis, probability, complex analysis to Cauchy's integral theorem, and topological spaces. A new proof development is sent to the Isabelle AFP every five days. This material — contributed by the worldwide Isabelle community — **is the raw material for ALEXANDRIA**.

Some mathematicians rely on *computer algebra systems* such as Maple and Mathematica to manipulate large formulas. There are similarities between such software and interactive theorem provers, but a derivation done using a computer algebra system cannot be called a proof. Computer algebra systems are focused on high performance when solving large problems. They sometimes make unwarranted assumptions to simplify their calculations. However, a few of the most important algorithms for computer algebra have been implemented as proof methods in interactive theorem provers, where their soundness is guaranteed. These algorithms include Gröbner basis methods [19], semi-definite programming (sum of squares decompositions) [39] and interval arithmetic [21]. And using computational reflection, bespoke algorithms can be verified and used to make specialised computations. **Algorithms are mathematical knowledge** and will be managed by ALEXANDRIA.

The Requirements of Mathematicians

Mathematicians are starting to look at such developments, but they soon encounter enormous differences between formalised mathematics and traditional mathematical practice. For starters, the many ambiguities in mathematical notation (contrast the various meanings of $y^{-1}x$, $f^{-1}g$, $f^{-1}[X]$, $\sin^{-1}x$, $\sin^2 x$, $\partial^2 f/\partial x$) must be resolved, but a formal mathematical syntax is often brutal. The resulting formulas can be all but unreadable, with proofs that look like (and often

[4]`http://www.isa-afp.org`

are) computer source code. This goes against a key principle of mathematics: a proof is not merely a guarantee of correctness, but a **form of communication** from one mathematician to another.

One strand of research focuses on logical foundations and formalisms. Relevant issues here include the treatment of undefined values such as $x/0$, the distinction between set theory and type theory, and the choice between classical and constructive logic. The recent work on homotopy type theory belongs to this strand. ALEXANDRIA, however, will focus on quite separate issues: on how to create and exploit large-scale libraries of mathematical knowledge, libraries that yield their secrets to humans as well as to machines. The scientific questions raised by large-scale mathematical libraries are independent of the specific formalism used to write assertions.

Prof. Tim Gowers, a Fields medallist at Cambridge, and Mohan Ganesalingam have started a project to create an automatic theorem prover for mathematics [27, 28]. Among their key requirements is that problems and solutions must be communicated using natural language. ALEXANDRIA takes the view that only a formal language can deliver the rigour we need; it will be as legible as possible, and natural language will play a number of supporting roles.

Structured Proofs and Proof Idioms

The main idea of LCF [31] was to code everything in a metalanguage (hence the name ML). Proofs were ML code, and users could add automation using ML, but not tamper with the inference system. LCF also introduced *proof tactics*, for proving theorems by working backwards from the conclusion. However, realistic mathematical arguments also require working forward from known facts, which is difficult to express using tactics. Because of these entrenched traditions, formal proofs in most proof assistants substantially consist of unreadable lists of commands.

Mizar is one proof assistant that emerged outside these traditions (in 1970s Poland). Designed to support mathematics, it provides a rich assertion language based on set theory [4]. Mizar proofs consist of declarations rather than commands: they are formal versions of mathematical arguments with a nested block structure. Each block proves a conclusion from given assumptions, all written out as explicit formulas. Compared with the usual list of proof tactics, a Mizar proof is longer but infinitely clearer. Although Mizar is no longer widely used, structured proofs live on in Isar [66], a simplified version of Mizar's proof language for Isabelle.

Figure 1 displays an Isar proof (shortened to save space) that the square root of a prime number p is irrational. Any mathematician should be able to follow this proof, given a brief introduction to the syntax. The main steps of the proof can plainly be seen: assuming that $\sqrt{p}$ is rational, writing it in lowest terms as m/n (this section has been highlighted manually), deducing that $m^2 = p \times n^2$, and then that p divides both m and n, contradiction. Automation (in the form of `auto`, `simp`, etc.) takes care of obvious reasoning, with only difficult calculations made explicit.

Legible proofs have several major advantages. They communicate proofs to human beings as well as to machines. (Isabelle proofs can be typeset automatically, as in the figure, and interleaved with markup and text.) And this reinforces trust in the software, since users can examine their own proofs. A sceptical mathematician will not be reassured by the claim that the proof assistant has verified its own correctness unless that very correctness proof is legible; the bare fact of its existence will carry little weight.

The highlighted text is an example of a *proof idiom*. It is simply a small chunk of formal text. Informally, it expresses a rational number as a pair of natural numbers having no common factor. Formally, it takes some preconditions (here *sqrt p* $\in \mathbb{Q}$) and yields two quantities, here *m* and *n*, satisfying the properties shown. In this case, the proof idiom is an application of a single theorem, *Rats_abs_nat_div_natE*. In general, a few theorems may be used together, with a varying number

```
theorem sqrt_prime_irrational:
  assumes "prime p"  shows "sqrt p ∉ ℚ"
proof
  assume "sqrt p ∈ ℚ"
  then obtain m n :: nat where
      n: "n ≠ 0" and sqrt_rat: "|sqrt p| = m / n"
    and "coprime m n" by (rule Rats_abs_nat_div_natE)
  have eq: "m² = p * n²" <omitted>
  have "p dvd m ∧ p dvd n"
  proof
    from eq have "p dvd m²" ..
    with ‹prime p› show "p dvd m" by (rule prime_dvd_power_nat)
    then obtain k where "m = p * k" ..
    with eq have "p * n² = p² * k²" by (auto simp add: power2_eq_square ac_simps)
    with p have "n² = p * k²" by (simp add: power2_eq_square)
    then have "p dvd n²" ..
    with ‹prime p› show "p dvd n" by (rule prime_dvd_power_nat)
  qed
  then have "p dvd gcd m n" by simp
  with ‹coprime m n› have "p = 1" by simp
  with p show False by simp
qed
```

Figure 1: An Isabelle Structured Proof

of premises and parameters.

Proof idioms are ideal for expressing reasoning that is too difficult to be done automatically. Isabelle can automatically apply the theorem named above, splitting a rational number into its numerator and denominator. But in this proof, m and n are needed to express an elaborate argument. Other such arguments include epsilon-delta calculations in analysis and the manipulation of summations. Many of these calculations will be unique to the theorem at hand, but the steps used in the preparation are reused in many proofs.

We can imagine a new type of automation. The user who assumes or deduces "sqrt p $\in \mathbb{Q}$" could be offered both a relevant lemma and an associated proof fragment: a proof idiom. The example above is only three lines, but reusable proof blocks can be much longer.

A promising way to identify proof idioms, and the theorems on which they are based, is to process the two million lines of proofs in the Isabelle libraries. Machine learning has already been shown to be effective at identifying relevant lemmas, in the context of Sledgehammer [10]. Prior work in the context of ACL2 [42] and Coq [43] investigates another sort of "proof pattern", using machine learning to identify similar-looking theorems and to locate common sequences of tactics in short, command-oriented proofs. An ALEXANDRIA proof idiom is fundamentally different: not a list of commands but a **self-contained structured subproof**, centred on the use of a lemma or lemmas and with explicit premises and conclusions.

Here is another example, from metric spaces. If x is an element of an open set S, then there exists some $\epsilon > 0$ such that $B \subseteq S$, where B is the open ball of radius ϵ around x (that is, $\{y \mid d(x,y) < \epsilon\}$). In a formal proof, this step would involve invoking a certain theorem but would include surrounding text specifying the set S and naming the variable ϵ. If instead the closed ball $\{y \mid d(x,y) \leq \epsilon\}$ is required, then a different theorem must be used, and the system itself should identify the main possibilities. If the proof goes on to use the axiom of choice to define a map $f : S \to \mathcal{P}S$ yielding a ball for each $x \in S$, that would be another idiom, and now $x \in S$ holds in a local scope. Such idioms will be found in the libraries automatically.

Heuristics for detecting proof idioms in libraries and suggesting them to users will be a top priority for ALEXANDRIA. **Formal proofs will be built section by section, not line by line.**

Maintenance and Migration of Libraries

A further strength of structured proofs is their synergy with automation. Isabelle provides several powerful automatic tools, above all Sledgehammer, which delivers the problem at hand together with all appropriate lemmas to external theorem provers such as Vampire and Z3 [9]. The combination of structured proofs and powerful automation makes it possible to write proofs using bisection: if the conclusion can't be proved immediately, think of an intermediate property implied by the assumptions, and see if the conclusion can be proved from that; if not, think of a further intermediate property and so forth. By the same bisection strategy, prove the intermediate properties from the given assumptions. Human creativity then consists of identifying these properties; beginners can prove theorems without learning many prover commands.

The synergy of structured proofs and automation facilitates maintenance. The Isabelle AFP is 12 years old, but all 1.4 million lines of proofs are maintained and continue to work as Isabelle is updated. Any failure in a structured proof is localised, and in most cases it is easily repaired using Sledgehammer or other automation. Indeed, we can envisage a procedure that would attempt to repair such faults automatically.

We can imagine how structured Isabelle proofs could one day be migrated to a more advanced system. Automatically translate the formulas to the (presumably more expressive) formalism of the new prover, translate the proof structures similarly, and finally use the (presumably much more powerful) automation of the new prover to fill in the gaps. This vision might be contrasted with the prevailing approach to interoperability, exemplified by Hurd's OpenTheory [48], which relies on having some lowest-common-denominator formalism. They transfer statements and claims of correctness, but no comprehensible proofs. It's useful, just like emulating old hardware is useful, but porting the old code to a more advanced programming language is better — especially if it can be done automatically. **ALEXANDRIA's library will have enduring utility and significance.**

Scientific Questions and Approach

Given all these observations, we are in a position to finalise the scientific objectives:

- How do we organise libraries of formal mathematical knowledge so that they communicate ideas to mathematicians, who prefer natural language?
- What sort of proof idioms can be extracted from a large corpus of formal material?
- What support do mathematicians most need when undertaking large formal proofs?
- **Algorithms are also knowledge**: how can mathematicians be advised of which algorithms are available and appropriate for a particular problem?

We can already imagine some answers to the questions above. Annotating formal proofs with links to the mathematics literature, and creating glossaries relating formal expressions to the corresponding concepts, would allow at least keyword-based search. In the future, this will be extended to use advanced information retrieval and natural language techniques. Similar techniques will be applied to computer algebra algorithms such as Gröbner basis methods. Machine learning is a key technology: analysis of these proof libraries will reveal a wealth of information about which theorems are invoked in particular situations, which is the basis for identifying proof idioms.

Although there are many computer-based formalisms for mathematics (every computer algebra system provides one), few are connected with a sound deductive system. ALEXANDRIA will be based on Isabelle/HOL, which supports readable structured proofs and has nearly two million lines of formalised mathematics in its libraries. Thanks to Sledgehammer, Isabelle arguably has the strongest automation of any proof assistant. Isabelle's user interface displays a live document that is continuously updated during editing, so that the status of every part of the proof is immediately clear to the eye. ALEXANDRIA will link this into ecosystem of libraries and decision procedures to support the formalisation of mathematics in the large. ALEXANDRIA will make suggestions in response to queries, and in some cases, unprompted.

The scientific questions above give us four research themes:

1. **Libraries of formal mathematics.** The goal is to create a comprehensive and well organised library of fundamental mathematics, annotated with references to the mathematical literature and linked to mathematical concepts to support sophisticated high-level searches.
2. **Intelligent search in libraries and identification of proof idioms.** Existing work on identifying relevant facts and common patterns will be transformed into a basis for identifying self-contained, legible proof idioms. Various approaches to high-level search will be investigated, using machine learning.
3. **Automated support for proof construction.** Proof idioms, relevant facts and other observations extracted from the libraries must be presented to the user in a way that is helpful and not distracting.
4. **Computer algebra and verified computations.** Identifying the right computer algebra method for the given problem is even more difficult than identifying relevant lemmas, since code is opaque, and there are too few past examples to drive machine learning.

But a key part of the methodology is to involve mathematicians as much as possible. The requirements they identify will set the priorities and refine the research programme.

2.b. Methodology

Research Methods

The elicitation of requirements from mathematicians will be accomplished by involving them at all stages of the work, from an initial pilot study to a final evaluation study, formalising research-level mathematics.

Much of the research methodology for ALEXANDRIA is well understood. There is extensive experience of formalising mathematics using proof assistants, and in particular in higher-order logic [3, 38, 40]. There is a good starting point for each of the four objectives. We already have the AFP, a huge library of mathematics; we have a basic library search mechanism and there is ongoing work on mining information from the AFP [11]; the Isabelle user interface already provides much support for proof construction [67]; a number of computer algebra procedures are already implemented [19]. There is also much prior work on using machine learning to analyse large bodies of formalised mathematics.

Outline Work Plan

The ALEXANDRIA team will include **two professional mathematicians**, and their first task will be to execute a year-long **pilot study** on the formalisation of some of their own mathematics research. This will identify issues connected with the scientific questions: the completeness of the existing libraries, their clarity, facilities for search and some wholly unanticipated problems. In recent years, mathematics graduates have done internships under the PI to formalise mathematical topics using Isabelle; their experiences have shaped this proposal, paving the way for a much deeper investigation. Executing the pilot study will serve as a stress test on Isabelle and its libraries and will refine the priorities of the project. As work proceeds, the project will organise regular workshops and tutorials in order to draw in other mathematicians and hear their voices.

The third researcher, an **Isabelle architect**, will come from the Isabelle community. This person will first undertake an **initial consolidation** task (with the PI), focusing on the needs that are already known. We are aware of the need for offline search (the ability to search libraries that aren't actually loaded into Isabelle). We already have a great many computer algebra decision procedures, which need to be consolidated and integrated with standard tools (a few of Isabelle's most impressive procedures [17] are almost never invoked, simply because users don't know that they exist). We are already aware of the potential for mining information from big libraries such as the AFP. As requirements emerge from the mathematicians, we can consider their feasibility and select the most promising ones to be implemented. The initial consolidation task will set the stage for the more ambitious work that will commence following the pilot study.

Equipped with the information from the pilot study, work on the main objectives can commence. An **assistant** or student with expertise in machine learning will join the project at this point. **Intelligent search and mining** for proof idioms can begin with the libraries as they are. However, the first objective concerns **formal libraries that convey mathematical ideas**; achieving this will be the task of the first mathematician, who will use a variety of techniques ranging from refactoring to annotations, with the ultimate objective of supporting natural language searches. (Natural language work will be done in collaboration with students and departmental colleagues.) The objective **automated support for proof construction** is substantially concerned with extracting proof idioms from previous proofs and therefore follows on from the task of mining libraries.

Support for **computer algebra technology** is substantially independent from the other tasks, as many of the requirements are known already. Nevertheless, this task will also be informed by the results of the initial pilot study. The second mathematician will be recruited with sufficient expertise in computer algebra techniques and real algebraic geometry to undertake this work.

Finally, an **extended case study** will serve to demonstrate and evaluate the project. It could be a continuation of the initial pilot study, or something quite different. Ideally it will be a collaborative effort involving a number of outside mathematicians. We expect that ALEXANDRIA will attract widespread interest. Ongoing cooperation with Nipkow's group at the Technical University of Munich will benefit the project substantially, in terms of engineering support, formalising necessary mathematics and evaluating the work.

Now let's look in more depth at the methodologies underlying the four main objectives.

Objective 1: Libraries of Formal Mathematics

We shall create a comprehensive and well organised library of fundamental mathematics. Such a library should be more than a list of formalised theorems and their proofs. It should contain references to the mathematical literature and conceptual annotations, in order to support sophisticated search techniques as well as conveying mathematical ideas.

The starting point for this library will be the enormous amount of mathematics that has been formalised. Much of this material is highly legible, although there is overlap and duplication. For the purposes of the project, full coverage is unnecessary and ALEXANDRIA will focus on applied mathematics, allowing the libraries to be useful for verification projects in domains such as aerospace.

The library will be organised according to the Cambridge undergraduate mathematics curriculum. This a systematic body of mathematical knowledge, strictly controlled by the Faculty of Mathematics. The course materials [64] include a detailed syllabus for each course, identifying prerequisite courses and recommended textbooks. Focusing on continuous mathematics, the following strands emerge:

- Vectors and Matrices; Linear Algebra; Vector Calculus
- Complex Methods (including Fourier and Laplace transforms)
- Metric and Topological Spaces; Analysis II; Topics in Analysis
- Variational Principles; Differential Equations; Mathematical Methods; PDEs
- Numerical Analysis I and II
- Optimisation (Linear programming, network flows, simplex and Ford-Fulkerson algorithms)
- Probability; Markov Chains; Stochastic Financial Models
- Groups, Rings and Modules; Algebraic Geometry

Structuring the library along these well-established lines will make it more intelligible to mathematicians. It is striking how much of this material has already been formalised in Isabelle, or in some version of HOL (from which it could be imported with a modest effort) or in Coq or Mizar. Isabelle provides most of the HOL Light multivariate analysis library [36] and much linear algebra, including vector spaces and matrices [23, 52]. An adequate library is already available as a starting point; it can be refined and extended as necessary as the project proceeds.

This task will be undertaken by the two mathematicians.

Objective 2: Intelligent Search in Libraries and Proof Idioms

The millions of lines of proofs at our disposal present opportunities as well as challenges. On the one hand, this material has outgrown Isabelle's existing search tools. But we can mine this vast body of proofs to identify useful lemmas. The use of machine learning to identify relevant lemmas [10] and proof patterns [43] is already well established. The next step is to extend this approach to proof idioms, which have been discussed above. An important special case is to identify lemmas that are particularly relevant given the available facts.

Explicit search will also be necessary, and search terms should include abstract mathematical concepts as well as formal symbols. The search problem is complex. The Isabelle theory of metric and topological spaces [46] illustrates some of the issues. The abstract notion of a *filter* unifies several motions related to limits, continuity and differentiation, and provides a uniform framework for such apparently disparate concepts as the limit of a continuous function and the sum of a series. Refinements such as left and right limits are also handled. The drawback of this highly abstract approach is that an intuitive concepts such as "sum of a series" is now encoded as a filter involving a certain pattern.

Complicating matters further, many mathematical concepts combine several ideas: an *open map* is a function that maps open sets to open sets, but no single symbol abbreviates this precise idea. ALEXANDRIA will develop strategies for annotating formal material with intuitive nomenclature so that users can identify the theorems they need. Information retrieval techniques will be used to support a variety of queries, such as to locate applications of a given theorem or to identify material on the basis of key concepts or by author. Indexes will be generated for off-line searching. Natural language can play a number of roles to support interaction with the software, in both searches and proof summarisation. We shall lay the groundwork to facilitate collaboration with researchers qualified in these technologies.

This task will be undertaken by the Isabelle architect, the assistant and the PI, and probably by graduate students and departmental colleagues.

Objective 3: Automated Support for Proof Construction

In the context of interactive theorem proving, automation has always referred to powerful decision procedures and simplifiers for proving specific statements. With Sledgehammer, Isabelle can invoke a battery of external provers, frequently solving problems that might take hours of user interaction. But other forms of automation assist productivity. For example, Isabelle informs the user if the statement to be proved is already known, or trivial, or clearly false.

The next stage is to assist users in writing extended proofs. Proof idioms will be a priority, and there is prior work [43] suggesting how advice derived from a library can be offered to users. But we can also envisage integrating proof idioms with Sledgehammer: given a set of available facts and a desired conclusion, Sledgehammer could identify and return a relevant proof idiom that fits the purpose. The beauty of this approach is that the user would not have to learn anything.

Another possibility is for the user to name a previous proof that the current proof is to be based on, and any analogous initial steps could be copied automatically. Ultimately, adding previously-used material to a formal proof could be driven by natural language, through a command such as "now there exists an open ball around x within the set S". ALEXANDRIA will open the door to follow-on projects using other technologies.

A further issue is that people do not always write clear proofs. As with programming, many users will be satisfied with the first proof they are able to finish, however messy and unreadable it may be. Automated refactoring of proofs will be necessary to achieve our goal of readable formal libraries. There is already valuable initial work by Blanchette et al. [8], which generates structured proofs from the low-level proofs delivered by external resolution theorem provers. More work is necessary to process tactic-oriented proofs, to achieve true readability and to incorporate proof idioms extracted from the existing libraries.

This task will be undertaken by the Isabelle architect, the assistant and the PI. Work involving natural language may be done by a graduate student supervised by departmental colleagues. Many projects connected with this task could be undertaken by Master's students here at Cambridge.

Objective 4: Computer Algebra and Verified Computations

Proofs in analysis sometimes require calculations — differentiation, polynomial identities, factoring — that are best done with the help of a computer algebra system. The objective of integrating computer algebra techniques with interactive theorem proving has been investigated for decades. Typically, theorem provers do not trust the outputs of computer algebra systems, leaving two approaches to ensuring correctness. The first is to check the provided answer (for example, an integral can be verified simply by taking the derivative) [41]; the second is to implement useful computer algebra algorithms within the proof assistant itself [56]. In either case, the calculation is confirmed by the prover's logical core. But this work doesn't address the issue of usability.

Isabelle has provided computer algebra technology for years, but these tools are scattered in various places and hardly used. These include Gröbner basis algorithms for proving algebraic identities [19], sum-of-squares methods for proving facts about polynomials [18], interval methods using floating-point arithmetic for proving inequalities involving transcendental functions [44], and Sturm sequence methods for counting the real roots of polynomials [24]. Moreover, there exist various ad hoc tricks for performing symbolic differentiation and algebraic simplification. Integrating these tools and techniques will yield a small number of outwardly simple proof methods for solving mathematical problems that arise within Isabelle proofs.

Some of these tools work by logic alone, while others involve *computational reflection* [6, 14]: the fast execution of verified algorithms via translation into ML. Reflection introduces a small degree of risk, because the translation process that transforms verified algorithms to executable code is itself unverified, but this risk is on the same scale as that of errors in the ML compiler itself. Computational reflection has been known for more than 30 years [15] as a way to deliver tremendous improvements in performance with little risk. Recent work [51] demonstrates that this translation process can itself be verified, with the prospect of reducing this risk almost to zero.

The recent formalisation of the Sturm-Tarski theorem [53] and real algebraic numbers [54] at Cambridge will allow the development of advanced algorithms [22], including polynomial root isolation and quantifier elimination for polynomial inequalities. Algorithms based on Bernstein polynomials, which can be used to bound the values of multivariate polynomials, should be easy to formalise thanks to prior work [7, 58]. In the guise of Bézier curves, they have applications to computer graphics in addition to their proven applications elsewhere [59]. Further decision procedures and algorithms will be implemented as necessary to support the project's other goals, based in turn on formalising necessary theory from Basu et al. [5] or elsewhere.

There are numerous instances of mathematical theory supported by bespoke computations. The Four Colour Theorem and the Kepler Conjecture are two well-known examples; Hales [33] describes many others. ALEXANDRIA will investigate what sort of general support can be provided for computations that form parts of proofs and therefore must be verified.

A comprehensive library of computer algebra procedures creates the same problems of search that we already have with libraries of formal mathematics, and to make matters worse, little information can be obtained automatically from the code itself. But machine learning cannot play a role until we accumulate a body of formal proofs that invoke these procedures.

This task will be undertaken by the second mathematician and the Isabelle architect.

Diagrammatic Work Plan

The Gantt chart (Fig. 2) presents the distribution of the tasks among the 60 months of the project. A darker colour indicates greater intensity of work. The Pilot Study and Initial Consolidation will prepare the ground for further work. A workshop at this point (summer 2018) will involve other mathematicians in the project goals. The tasks of Intelligent Search and Automated User Support go together, as the former feeds into the latter. More general work on the Libraries and on Computer Algebra will be less intensive, unless the Pilot Study alters the priorities. Evaluation takes place in the last year, but some other work will continue.

Summary

ALEXANDRIA will bring together two mathematicians and an ATP architect, along with an assistant or student, with the aim of identifying and implementing the most important ideas and technologies needed to support formalised mathematics **in the large**. The mathematicians will

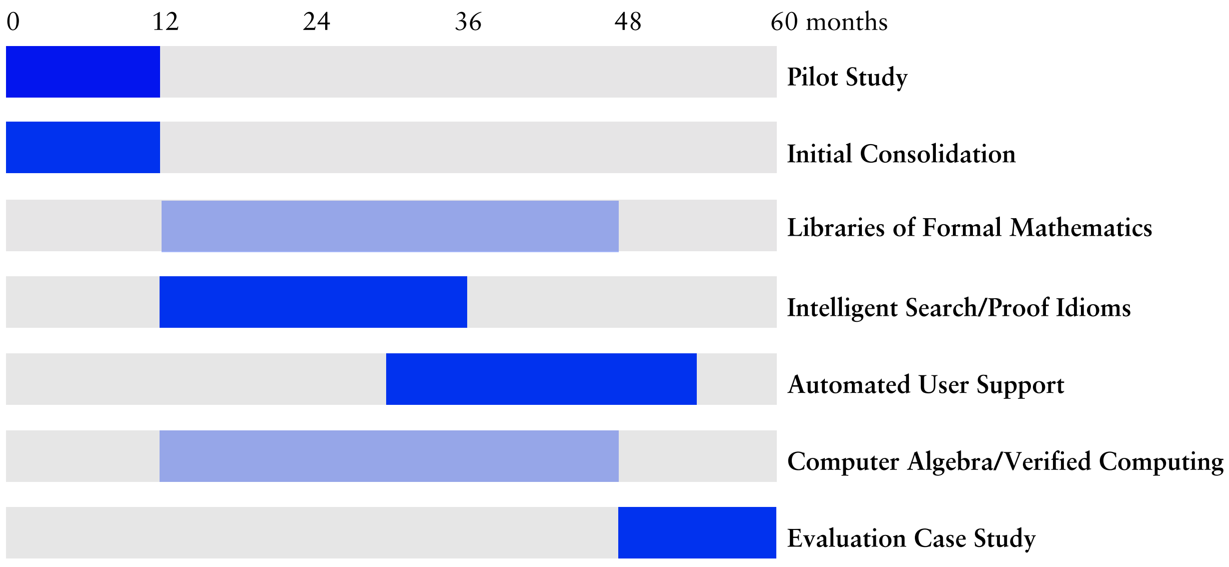


Figure 2: Task Chart for ALEXANDRIA

refactor the existing body of formalised mathematics and formalise new research-level material, while documenting difficulties and discussing possible solutions. The architect's job will be to examine the feasibility of these suggestions and implement them. The work plan includes a pilot study, tasks connected with the research objectives, and a final evaluation study, extending over five years. One of the main scientific objectives is simply to elicit requirements from mathematicians themselves.

ALEXANDRIA is particularly concerned with the management of large libraries of formalised mathematics. Formal mathematics must be legible and must communicate the underlying mathematical ideas to human readers. A key priority is to investigate the identification and reuse of proof idioms — self-contained, declarative proof fragments — so that proof can be built piece by legible piece rather than line by cryptic line.

Project outcomes will be incorporated into Isabelle or the AFP where they will be freely available. Workshops will be scheduled at suitable intervals in order to broaden the interactions between mathematicians and the theorem proving community. Papers describing project results will be published in leading journals, conferences and workshops, including mathematics journals.

Rewards and Impacts

Doubts in mathematical results are now pervasive. Many people know that Wiles's proof of Fermat's Last Theorem contained a serious error that could not be corrected for a year. Dramatic announcements — a solution to the ABC Conjecture, a fast new algorithm for graph isomorphism — are routinely greeted with scepticism. We work towards the day when new results are published as machine-checked formal documents and there is no longer any question of mistakes in proofs.

ALEXANDRIA will create a system that can cope with enormous bodies of formalised mathematics and assist mathematicians in finding what they need to write large, verified proofs.

ALEXANDRIA will deliver a well-organised and searchable body of formalised mathematics, covering most of the undergraduate curriculum and containing appropriate literature references.

These proofs will be legible to humans as well as to machines. It will consist of declarative structured proofs. It will be a valuable resource for both research mathematics and university teaching.

ALEXANDRIA will also have a transformative effect on the verification community. To verify a real-world system frequently involves creating a large proof involving a significant amount of mathematics. For example, the dynamics of cyber-physical systems typically involve analogue phenomena expressed using differential equations, which often have no closed-form solutions. Such verification problems are just another form of large-scale mathematics.

ALEXANDRIA will create a usable proof development environment for mathematicians, building on the Isabelle theorem prover. Working alongside mathematicians to elicit requirements, and enlisting the aid of computer scientists from a range of specialities, ALEXANDRIA will deploy a multiplicity of technologies to support working mathematicians.

2.c. Resources (including action costs)

Scientific Environment

The University of Cambridge is one of the foremost universities in the world. The Computer Laboratory is the birthplace of both Isabelle and HOL, and pioneered many applications of formal verification, including hardware, floating point numerical algorithms and cryptographic protocols. The Computer Laboratory, along with the Department of Engineering, has expertise in other technologies relevant to ALEXANDRIA, including machine learning and natural language processing. Cambridge also has one of the world's leading departments of Mathematics.

Paulson created Isabelle in 1986 and made major contributions to HOL. He created Sledgehammer, the first successful link between interactive theorem provers and automatic theorem provers. He created MetiTarski [1], the first integration between a classical automatic theorem prover and an advanced decision procedure for polynomials. Paulson is also one of the three editors of the Archive of Formal Proofs. Thus he is conversant with one of the main mathematical libraries and several of the main proof assistants. He has a strong track record in formalising advanced mathematics, notably the works of Gödel [61, 62]. He has supervised two Ph.D. dissertations [16, 47] on machine learning. Paulson has a track record of delivering ambitious projects.

Personnel

ALEXANDRIA will hire *three postdoctoral researchers* for five years to assist the PI in carrying out the project tasks. These researchers will need strong qualifications (two in mathematics, one in automated theorem proving), and an appropriately high level of salary is proposed. The project vision, along with prospect of a five-year position at Cambridge, should attract high-quality applicants. An *assistant* or student will join the project at the start of year 2 and work for 42 months, chiefly on Intelligent Search and Automated User Support.

Budget

The *travel* budget assumes a total of 10 trips per year among the five team members, mostly to conferences in order to communicate with colleagues and disseminate project results. Most travel will be within the EU, but major events do occur in the USA and elsewhere. We estimate €104,140 for travel and subsistence and a further €22,860 for conference fees.

The *computing* budget is necessary due to the project's heavy use of Isabelle, which is demanding of processor power and memory. The PI and each of the three postdoctoral researchers will require a powerful workstation (costing €4445) and a notebook computer (€1524). The notebooks are required to demonstrate our technology at conferences, and allow working away from the office. The expected lifetime of a computer is three years, so at the start of the fourth year, new computers will be necessary for each of the four project participants. Computers are budgeted under consumables.

Although most software used in the project is free, the project will purchase two (permanent) licences for the Maple computer algebra system at €1270 each. A further €1270 will cover other potential software requirements.

A *Poly/ML support contract* costing €1270 per year will ensure that any problems with Poly/ML will be dealt with quickly; this is crucial, as Isabelle requires Poly/ML to run.

PI's Commitment to the Project

The PI will devote 50% of his working time to the project over the period of the grant. Paulson will not accept any senior administrative responsibilities, such as chairing a department or organisation. His department will reduce his undergraduate-level teaching load by half.

References

[1] B. Akbarpour and L. Paulson. MetiTarski: An automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 44(3):175–205, Mar. 2010.

[2] K. Appel and W. Haken. Every planar map is four colorable. *Bull. Amer. Math. Soc.*, 82(5):711–712, 09 1976.

[3] J. Avigad, J. Hölzl, and L. Serafin. A formally verified proof of the central limit theorem, 2016. preprint.

[4] G. Bancerek and P. Rudnicki. A compendium of continuous lattices in Mizar. *Journal of Automated Reasoning*, 29(3-4):189–224, 2002.

[5] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Springer, 2nd edition, 2006.

[6] S. Berghofer and T. Nipkow. Executing higher order logic. In P. Callaghan, Z. Luo, J. McKinna, and R. Pollack, editors, *Types for Proofs and Programs (TYPES 2000)*, volume 2277 of *LNCS*, pages 24–40. Springer, published 2002.

[7] Y. Bertot, F. Guilhot, and A. Mahboubi. A formal study of Bernstein coefficients and polynomials. *Mathematical Structures in Computer Science*, 21(04):731–761, 2011.

[8] J. C. Blanchette, S. Böhme, M. Fleury, S. J. Smolka, and A. Steckermeier. Semi-intelligible isar proofs from machine-generated proofs. *J. Autom. Reasoning*, 56(2):155–200, 2016.

[9] J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT solvers. *Journal of Automated Reasoning*, 51(1):109–128, 2013.

[10] J. C. Blanchette, D. Greenaway, C. Kaliszyk, D. Kühlwein, and J. Urban. A learning-based fact selector for isabelle/hol. *Journal of Automated Reasoning*, pages 1–26, 2016.

[11] J. C. Blanchette, M. P. L. Haslbeck, D. Matichuk, and T. Nipkow. Mining the Archive of Formal Proofs. In M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings*, volume LNCS 9150, pages 3–17. Springer, 2015.

[12] S. Blazy, C. Paulin-Mohring, and D. Pichardie, editors. *Interactive Theorem Proving — 4th International Conference*, LNCS 7998. Springer, 2013.

[13] S. Boldo, J.-H. Jourdan, X. Leroy, and G. Melquiond. A formally-verified C compiler supporting floating-point arithmetic. In *ARITH, 21st IEEE International Symposium on Computer Arithmetic*, pages 107–115. IEEE Computer Society Press, 2013.

[14] S. Boutin. Using reflection to build efficient and certified decision procedures. In M. Abadi and T. Ito, editors, *Theoretical Aspects of Computer Software*, LNCS 1281, pages 515–529. Springer, 1997.

[15] R. S. Boyer and J. S. Moore. Metafunctions: Proving them correct and using them efficiently as new proof procedures. In *The Correctness Problem in Computer Science*, pages 103–184. Academic Press, 1981.

[16] J. P. Bridge. *Machine Learning and Automated Theorem Proving*. PhD thesis, University of Cambridge, 2010. Online at `URLhttp://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-792.pdf`.

[17] A. Chaieb. Verifying mixed real-integer quantifier elimination. In Furbach and Shankar [26], pages 528–540.

[18] A. Chaieb. *Automated Methods for Formal Proofs in Simple Arithmetics and Algebra*. PhD thesis, Technical University of Munich, 2008.

[19] A. Chaieb and M. Wenzel. Context aware calculation and deduction: Ring equalities via Gröbner bases in Isabelle. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, LNCS 4573, pages 27–39. Springer, 2007.

[20] B. I. Dahn. Robbins algebras are boolean: A revision of McCune's computer-generated solution of Robbins problem. *Journal of Algebra*, 208(2):526–532, 1998.

[21] M. Daumas, C. Muñoz, and D. Lester. Verified real number calculations: A library for interval arithmetic. *IEEE Transactions on Computers*, 58(2):226–237, 2009.

[22] L. de Moura and G. O. Passmore. Computation in real closed infinitesimal and transcendental extensions of the rationals. In M. P. Bonacina, editor, *Automated Deduction —- CADE-24*, LNCS 7898, pages 178–192. Springer, 2013.

[23] J. Divasón and J. Aransay. Rank-nullity theorem in linear algebra. *Archive of Formal Proofs*, Jan. 2013. `http://afp.sf.net/entries/Rank_Nullity_Theorem.shtml`, Formal proof development.

[24] M. Eberl. Sturm's theorem. *Archive of Formal Proofs*, Jan. 2014. `http://afp.sf.net/entries/Sturm_Sequences.shtml`, Formal proof development.

[25] C. Edwards. Automating proofs. *Commun. ACM*, 59(4):13–15, Mar. 2016.

[26] U. Furbach and N. Shankar, editors. *Automated Reasoning — Third International Joint Conference, IJCAR 2006*, LNAI 4130. Springer, 2006.

[27] M. Ganesalingam. *The Language of Mathematics - A Linguistic and Philosophical Investigation.* LNCS 7805. Springer, 2013. Based on his Cambridge PhD thesis.

[28] M. Ganesalingam and W. T. Gowers. A fully automatic theorem prover with human-style output. *Journal of Automated Reasoning*, doi:10.1007/s10817-016-9377-1, 2016.

[29] G. Gonthier. The four colour theorem: Engineering of a formal proof. In D. Kapur, editor, *Computer Mathematics*, LNCS 5081, pages 333–333. Springer Berlin Heidelberg, 2008.

[30] G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. Le Roux, A. Mahboubi, R. O'Connor, S. Ould Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi, and L. Théry. A machine-checked proof of the odd order theorem. In Blazy et al. [12], pages 163–179.

[31] M. J. C. Gordon, R. Milner, and C. P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation.* LNCS 78. Springer, 1979.

[32] T. C. Hales. An overview of the Kepler conjecture. *ArXiv Mathematics e-prints*, Nov. 1998.

[33] T. C. Hales. Mathematics in the age of the Turing machine. In R. Downey, editor, *Turing's Legacy*, pages 253–298. Cambridge University Press, 2014. Cambridge Books Online.

[34] T. C. Hales et al. A formal proof of the Kepler conjecture. *arXiv.org*, abs/1501.02155, Jan. 2015.

[35] T. C. Hales, J. Harrison, S. McLaughlin, T. Nipkow, S. Obua, and R. Zumkeller. A revision of the proof of the Kepler conjecture. *Discrete & Computational Geometry*, 44(1):1–34, 2010.

[36] J. Harrison. A HOL theory of Euclidean space. In J. Hurd and T. Melham, editors, *Theorem Proving in Higher Order Logics: TPHOLs 2005*, LNCS 3603, pages 114–129. Springer, 2005.

[37] J. Harrison. Towards self-verification of HOL Light. In Furbach and Shankar [26], pages 177–191.

[38] J. Harrison. Formalizing basic complex analysis. In R. Matuszewski and A. Zalewska, editors, *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*, volume 10(23) of *Studies in Logic, Grammar and Rhetoric*, pages 151–165. University of Białystok, 2007.

[39] J. Harrison. Verifying nonlinear real formulas via sums of squares. In K. Schneider and J. Brandt, editors, *Theorem Proving in Higher Order Logics: TPHOLs 2007*, LNCS 4732, pages 102–118. Springer, 2007.

[40] J. Harrison. Formalizing an analytic proof of the prime number theorem. *Journal of Automated Reasoning*, 43(3):243–261, 2009.

[41] J. Harrison and L. Théry. A skeptic's approach to combining HOL and Maple. *Journal of Automated Reasoning*, 21(3):279–294, 1998.

[42] J. Heras and E. Komendantskaya. Acl2(ml): Machine-learning for ACL2. In *Twelfth International Workshop on the ACL2 Theorem Prover and its Applications*, pages 61–75, 2014.

[43] J. Heras and E. Komendantskaya. Recycling proof patterns in Coq: Case studies. *Mathematics in Computer Science*, 8(1):99–116, 2014.

[44] J. Hölzl. Proving inequalities over reals with computation in Isabelle/HOL. In G. D. Reis and L. Théry, editors, *ACM SIGSAM 2009 International Workshop on Programming Languages for Mechanized Mathematics Systems (PLMMS'09)*, pages 38–45, Munich, August 2009.

[45] J. Hölzl and A. Heller. Three chapters of measure theory in Isabelle/HOL. In M. Eekelen, H. Geuvers, J. Schmaltz, and F. Wiedijk, editors, *Interactive Theorem Proving — Second International Conference*, LNCS 6898, pages 135–151. Springer, 2011.

[46] J. Hölzl, F. Immler, and B. Huffman. Type classes and filters for mathematical analysis in Isabelle/HOL. In Blazy et al. [12], pages 279–294.

[47] Z. Huang. *Machine Learning and Computer Algebra.* PhD thesis, University of Cambridge, 2016. Online at `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-884.pdf`.

[48] J. Hurd. The OpenTheory standard theory library. In M. G. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, editors, *NASA Formal Methods — NFM 2011*, LNCS 6617, pages 177–191. Springer, 2011.

[49] T. J. Jech. *The Axiom of Choice.* North-Holland, 1973.

[50] G. Klein, J. Andronick, K. Elphinstone, G. Heiser, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. sel4: Formal verification of an operating-system kernel. *Commun. ACM*, 53(6):107–115, June 2010.

[51] R. Kumar, R. Arthan, M. O. Myreen, and S. Owens. Self-formalisation of higher-order logic - semantics, soundness, and a verified implementation. *J. Autom. Reasoning*, 56(3):221–259, 2016.

[52] H. Lee. Vector spaces. *Archive of Formal Proofs*, Aug. 2014. `http://afp.sf.net/entries/VectorSpace.shtml`, Formal proof development.

[53] W. Li. The Sturm-Tarski theorem. *Archive of Formal Proofs*, Sept. 2014. `http://afp.sf.net/entries/Sturm_Tarski.shtml`, Formal proof development.

[54] W. Li and L. C. Paulson. A modular, efficient formalisation of real algebraic numbers. In J. Avigad and A. Chlipala, editors, *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, Saint Petersburg, FL, USA, January 20-22, 2016*, pages 66–75. ACM, 2016.

[55] D. MacKenzie. *Mechanizing Proof: Computing, Risk, and Trust.* MIT Press, 2004.

[56] S. McLaughlin and J. Harrison. A proof-producing decision procedure for real arithmetic. In R. Nieuwenhuis, editor, *Automated Deduction — CADE-20 International Conference*, LNAI 3632, pages 295–314. Springer, 2005.

[57] T. Mhamdi, O. Hasan, and S. Tahar. Formalization of measure theory and Lebesgue integration for probabilistic analysis in HOL. *ACM Trans. Embedded Comput. Syst.*, 12(1):13, 2013.

[58] C. Muñoz and A. Narkawicz. Formalization of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, 51(2):151–196, 2013.

[59] A. Narkawicz and C. Muñoz. Formal verification of conflict detection algorithms for arbitrary trajectories. *Reliable Computing*, 17:209–237, Dec. 2012.

[60] T. Nipkow, G. Bauer, and P. Schultz. Flyspeck I: tame graphs. In Furbach and Shankar [26], pages 21–35.

[61] L. C. Paulson. The relative consistency of the axiom of choice — mechanized using Isabelle/ZF. *LMS Journal of Computation and Mathematics*, 6:198–248, 2003. `http://www.lms.ac.uk/jcm/6/lms2003-001/`.

[62] L. C. Paulson. A machine-assisted proof of Gödel's incompleteness theorems for the theory of hereditarily finite sets. *Review of Symbolic Logic*, 7(3):484–498, Sept. 2014.

[63] G.-C. Rota. *Indiscrete Thoughts.* Springer, 2009.

[64] University of Cambridge. Undergraduate mathematics: Course information, 2014. `http://www.maths.cam.ac.uk/undergrad/course/`.

[65] V. Voevodsky. The origins and motivations of univalent foundations. *The Institute Letter*, Summer 2014. Institute for Advanced Study . Online at `https://www.ias.edu/ideas/2014/voevodsky-origins`.

[66] M. Wenzel. Isabelle/Isar — a generic framework for human-readable proof documents. In R. Matuszewski and A. Zalewska, editors, *From Insight to Proof — Festschrift in Honour of Andrzej Trybulec*. University of Białystok, 2007. Studies in Logic, Grammar, and Rhetoric 10(23).

[67] M. Wenzel. Asynchronous user interaction and tool integration in Isabelle/PIDE. In G. Klein and R. Gamboa, editors, *Interactive Theorem Proving — 5th International Conference, ITP 2014*, LNCS 8558, pages 515–530. Springer, 2014.