Final Year Project

Professor Matthew Leeke
School of Computer Science
University of Birmingham

Report Writing Guidance

The Golden Rules

Your supervisor is the most important person to speak to about your project

Taking ownership and making sensible decisions is vital to success

Developing a plan and executing on it always beat genius IQ

Project Admin

Project Milestones

Project Proposal (0%) - Deadline at 5pm on Monday of Week 7, Term 1

Project Demonstration (0%) - Delivered at the end of Term 2 (Week 8-10)

Project Report (100%)

Project Proposal

Deadline at 5pm on Monday of Week 7, Term 1

… but you should really be trying to submit early - I'd suggest Week 5

Counts for 0% module credit

… but it's an important check on your idea and progress

Flexible structure

… but an opportunity to write content that will follow through to your final report

Project Proposal - Feedback

Comments on Project Proposals will be provided by your project inspector

Feedback by meeting with your Supervisor

Establishing a working relationship with your supervisor is important!

Final Report Structure - An Example

Writing Reports and Academic Papers

What is the purpose of a coursework report?

What do your readers want to find out?

How will they be reading your report?

Is there a difference between a coursework report and an academic paper?

A Final Report Structure for Software Projects

For software based projects:

Abstract - Usually 100-300 words stating the salient points of the report. It should help your reader to decide whether the report is relevant to her or his interests.

Introduction - The introduction should provide content for the report, discuss relevant background material, identify stakeholders, and state the aim(s) of the work.

Research - This is often referred to the 'literature review' section. It is one of the most important section of the Project Proposal and the Final Report. It is where you demonstrate that you understand the state-of-the-art in the field you're working. Towards the end of this sections it's normally a good idea to explain how your aim / work / idea / contribution differs from the nearest work in the field.

What Does The Structure of a Final Report Look Like

Legal, Social, Ethical and Professional Issues - Address the legal, social, ethical, and professional issues associated with your project. Find something to say for each. For example, listening for legal, BCS Code of Conduct for professional, etc..

System Requirements - This section should detail your understanding of what you are planning to create. The section should aim to break down the overarching aims of the work into clear, measurable requirements that can be used in the evaluation of the project. This is why we often function of functional and non-functional requirements.

Design - Communicating how you think about the composition of your system and how it works. You might detail the ways in which the overall system will be broken down into subsystems. Detail should then be provided on the design of each of these subsystems.

What Does The Structure of a Final Report Look Like

Implementation - This section should discuss how you went about developing a system that was consistent with your design to meet your stated requirements. The implementation of subsystems should be accurately documented, with any implementation difficulties being acknowledged. The Design and Implementation sections can be grouped in the Final Report, if these are tightly coupled. Likely omitted for the Project Proposal, though should mention your proposed implementation technologies somewhere.

Testing and Success Measurement - As well as documenting system testing, this section should also describe any unit testing or integration testing performed. If you are not familiar with unit, integration or system testing then it would be a good idea to investigate these notions and consider about how they relate to your project. This section might also detail any performance, reliability or usability testing performed, with quantification, i.e., numeric measurements, being used wherever possible. All those points are systems focused through. If you're doing something that's research focused or more of a social / analytical study then you need to think about how you'll measure success.

What Does The Structure of a Final Report Look Like

Project Management - What is the timeline for your project? What software development methodology will you use? Can you identify the major milestones? These types of questions are important for project planning and should be addressed directly.

Evaluation - Usually you evaluate your project with regard to the functional and non-functional requirements you set out in the earlier chapter. This doesn't necessary mean that your project was successful but, if these requirements were appropriately specified, you it's likely that your project was successful. You might be reiterating some points from the Testing and Success Measurement chapter in your Final Report.

What Does The Structure of a Final Report Look Like

Conclusion - Generally speaking, section can take different forms - as a minimum you would normally provide a brief summary of your project work and a discussion of possible future work. You may also wish to reiterate the main outcomes of your project and give some idea of how you think the ideas dealt with by your project relate to real-world situations, etc.. For the Proposal, don't mention future work but do summarise your document.

References - Referencing is vital when building on the work of others. You may use any style of referencing but you should be consistent. All forms of plagiarism are taken extremely seriously. This is the first section that most markers will turn to when presented with a proposal or report.

What Does The Structure of a Final Report Look Like

Appendices - Some information, for example program listings, is useful to include within the report for completeness, but would distract the reader from the flow of the discussion if it were included within the body of the document. Short extracts from major programs may be included to illustrate points but full program listings should only ever be placed within an appendix. Remember that the point of appendices is to make your report more readable. I don't expect to see apprentices in any Project Proposals.

Subjectively, I would hope not to see any in Final Report but sometimes they're necessary.

Final Report Structure - Caveats

If your project is less systems / development focused then it's likely that the System Requirements makes less sense. It's often the case that research focused and social / analytical study projects will have a Methodology section, where you lay out the process by which you will meet the aims you've laid out in the Introduction. These structure for these types of projects would be more like Introduction -> Research-> LSEPIs -> Methodology -> Step 1 -> Step 2 -> [...] -> Step N -> Project Management -> Evaluation -> Conclusion -> References.

- If your project is more systems / development focused then your Research section might focus more on existing system rather than research papers. This doesn't absolve you of the responsibility to understand the state-of-the-art or use research papers but it's more permissible not to have conference and journal papers dominating your list of references.

- It's absolutely fine (and expected) for your aim / requirements / focus / project to change.

What About the Project Proposal?

It would be normal for a Project Proposal to omit many sections, including Design, Implementation and Evaluation sections

Testing and Success Measurement section and would be more about what you're planning to do to measure success than what you've already done

Ultimately, you should speak to your Supervisor about what they're expecting

Academic Writing

Writing Coursework and Academic Papers

What is the purpose of a coursework report?

What do your readers want to find out?

How will they be reading your report?

Is there a difference between a coursework report and an academic paper?

Characterising Academic Writing

Communicates with its reader clearly and persuasively

High quality content

Logical structure

Precise style

Conventions to show how source material has been used

Translating Research Activities to Academic Writing

Identify a research question

Be clear about your contribution - is there something new in what you're doing?

Critique and evaluate your results - perspective, history, assumptions, authority

Consider your audience

Language

Avoid exaggeration, colloquialisms and anecdotal language

It's nice that you think your idea is "excellent" but your reviewer may not agree

Avoid contradiction, cliches and abbreviations

At the end of the day, it's like chalk and cheese

Be specific and assertive where possible, avoiding words like 'may', 'could', 'perhaps', 'might', etc. unless you intend to be speculative

Precision and restraint are important when expressing yourself

Can You Be Precise?

Can you shorten the sentence below to no more than 17 words?

"There has been much less research in education during the past year due to the fact that there is a complete lack of public funding for it." (27 words)

Can I Be Precise?

“There has been much less research in education during the past year due to the fact that there is a complete lack of public funding for it.” (27 words)

“A lack of public funding has led to less research in education this year.” (14 words)

Title

Writing a Compelling Title

Identify the SUBJECT

Describe the CONTENT

ACCURATE, CONCISE, and SPECIFIC

Contains KEYWORDS

The Purposes of Titles

Indicative

Indicates the subject matter of the dissertation but gives no indication of results or conclusions

Informative

Gives an indication of the results and conclusions drawn as well as the subject matter

The Purposes of Titles

Question

Establishes the question the dissertation will seek to answer

Subtitled

Indicates a broad subject area and then narrows the focus

Sample Titles

"Exploring the Use of Micro Note-Taking with Social Interaction Features for Education "

"A Pedagogical Evaluation of E-learning from Users' Perspective: The Case of MOODLE in the University of Warwick"

"A Spatially-Intensive Decision Tree Approach to Predicting the Traffic Flow of the entire UK Road Network"

"A cross-sectional analysis of green space prevalence and mental wellbeing in England"

Abstract

Writing an Abstract

Abstract should make clear:

- Field of study
- Research problem and contributions
- Methods / procedure / approach
- Key result(s)
- Keywords

Abstract Pitfalls

Making it overly long

Using references to tables or figures

Using references to literature cited unless you are specifically addressing a key theory or piece of literature from which to build your argument

Drawing conclusions which are not contained in / do not follow from the paper

Introduction

Writing an Introduction

Brief and focused - typically 10-15% of your total word count

1. Provide a broad background or context to the field or discipline and topic, to orient the reader so that they understand what follows

2. Setup the issue – What's the problem? What is the scope and limitation existing solutions?

3. State your contributions or argument – What are you brining that's new?

4. Signpost and outline the structure of your work

Evaluating an Introduction

Does it provide the background to allow engagement with what follows?

Can you find the problem statement?

Can you identify the contribution the work will make?

Does it provide a definition and explanations of key terms?

How the work will be structured and presented?

Critique Me - Title

Simultaneous Fault Models for the Generation and Location of Efficient Error Detection Mechanisms

MATTHEW LEEKE

Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK
Email: matthew.leeke@warwick.ac.uk

Critique Me - Title

The application of machine learning to software fault injection data has been shown to be an effective approach for the generation of efficient error detection mechanisms (EDMs). However, such approaches to the design of EDMs have invariably adopted a fault model with a single-fault assumption, limiting the relevance of the detectors and their evaluation. Software containing more than a single fault is commonplace, with safety standards recognising that critical failures are often the result of unlikely or unforeseen combinations of faults. This paper addresses this shortcoming, demonstrating that it is possible to generate efficient EDMs under simultaneous fault models. In particular, it is shown that (i) efficient EDMs can be designed using fault injection data collected under models accounting for the occurrence of simultaneous faults, (ii) exhaustive fault injection under a simultaneous bit flip model can yield improved EDM efficiency, (iii) exhaustive fault injection under a simultaneous bit flip model can be made non-exhaustive and (iv) EDMs can be relocated within a software system using program slicing, reducing the resource costs of experimentation to practicable levels without sacrificing EDM efficiency.

Keywords: Detection; Error; Model; Location; Machine Learning

Critique Me - Introduction

1. INTRODUCTION

The design of error detection mechanisms (EDMs) is integral to the development of dependable software systems [1]. EDMs are fundamentally concerned with the detection of erroneous software states. Once detected by an EDM, erroneous software states can be handled by error recovery mechanisms (ERMs) to maintain proper function. A failure to contain the propagation of erroneous state is known to make recovery more difficult, leading to a focus on the efficiency of EDMs through measures such as coverage and latency [2].

The effectiveness of an EDM has been shown to depend on two factors. These factors are (i) the error detection predicate that it implements and (ii) its location in a software system [3]. This gives rise to two related problems. Firstly, the EDM design problem, which is concerned with the derivation of an error detection predicate over program variables that can be used for the detection of erroneous system states. Secondly, the EDM location problem, which is concerned with the identification of those software locations at which an EDM will be most effective. Though often treated as orthogonal for simplicity, the interaction of the implemented error detection predicate and the software location are demonstrably critical to the efficiency of an EDM [4].

The efficiency of an EDM can be characterised by completeness and accuracy [3]. Completeness is the capability of an EDM to detect erroneous states, i.e., its associated true positive rate. In contrast, accuracy is the capability of an EDM to avoid incorrectly detecting erroneous states, i.e., its associated false positive rate. An erroneous state is one that will lead to system failure if the error is not handled, where a failure is characterised as a violation of a system specification. An EDM that is complete and accurate is commonly known as a perfect detector. Due to implementation constraints, it is not generally possible to generate or guarantee the existence of a perfect detector for a particular software location [5].

The role of a fault model is to provide a means for analysing the response of software system to the presence a well defined set of faults, such that appropriate EDMs and ERMs can be designed to impart dependability. The assumption that faults do not occur simultaneously or interact is a limitation of many fault models and the software fault injection frameworks that implement them, not least because software containing more than a single fault are now commonplace. Indeed, numerous existing safety standards recognise that critical system failures are often the result of unlikely or unforeseen interactions combinations of faults [6].

It has been shown that efficient error detection predicates for EDMs can be designed through the application of machine learning algorithms to data sets generated during software fault injection [7]. This approach demonstrated, under a transient data value fault model, that it was possible to generate error detection predicates for specified locations with a true positive rate of nearly 100% and a false positive rate close to 0% for the detection of failure-inducing states. As is consistent with the overwhelming majority of software fault injection frameworks, these results were achieved under a single-fault assumption, calling into question their relevance in the context of real-world software systems.

Critique Me - Introduction

1.1. Problem Statement

In generating error detection predicates for EDMs through the application of machine learning to fault injection data, an implication of the single-fault assumption is that the efficiency of the EDMs generated is relevant in the context of a single fault. This implication limits the application of these EDMs in practical software systems.

This paper addresses this problem by demonstrating that practicable simultaneous fault injection can be used to generate efficient EDMs for a specified location in a software system. In doing this it is shown that the adoption of a simultaneous fault model enables a larger set of faults to be captured than existing models that make the single-fault assumption. The viability of simultaneous fault models is based on results demonstrating that exhaustive fault injection under a simultaneous bit flip model can be made non-exhaustive and that EDMs can be relocated without regeneration.

1.2. Contributions

This paper makes several specific contributions to the design of efficient EDMs. In particular, the research presented demonstrates that:

- Efficient EDMs can be designed using fault injection data collected under models accounting for the occurrence of simultaneous faults;
- Exhaustive fault injection under a simultaneous bit flip model can yield better EDM efficiency than under a non-simultaneous fault model;
- Exhaustive fault injection under a simultaneous bit flip model can be made non-exhaustive without sacrificing the efficiency of the resultant EDMs, thus reducing the resource costs of experimentation to a practicable level.
- Efficient EDMs can be relocated within a software system using program slicing without sacrificing the efficiency of those EDMs.

1.3. Paper Structure

The remainder of this paper is structured as follows: Section 2 provides an overview of research relating to fault models for detector design. System 3 details the adopted system, data and fault models. Section 4 provides an overview of how machine learning algorithms can be used to generate detection predicates for specified software locations. Section 5 provides details of the experiments conducted in this paper, including details of target software systems and applied machine learning algorithms. Section 6 presents the results of the experiments conducted, alongside a discussion of their significance in the context of efficient error detector design. Section 7 concludes the paper with a summary of findings and a brief discussion of future work in EDM design.

Literature Review

Literature Reviews

A literature review allow you to demonstrate relevance and significance of the work you're doing to address a research question

Positioning your work amongst what has been done in the past is absolutely essential when it comes to other people appreciating your contribution

Writing a Literature Review

Situates the problem within its academic context

- Critically evaluates and analyses the existing literature in order to identify the gaps in knowledge or the problem

They allow you to:

- Demonstrate originality
- Demonstrate your ability to critically engage with the topic through perceptive analysis
- Relate your research to existing scholarship

Structuring a Literature Review - Chronological

Chronological – charting the evolution of the literature over time

Works best for the topics where there is only one key problem or issue

Charts the evolving attitudes of scholars to a domain

Not always good for problems which are multi-faceted

Gives a clear structure to the writing process

Structuring a Literature Review - Thematic

Thematic – chart scholarly responses to the issue problem by problem

Excellent for complex subjects

Need to prioritise the seminal material and the closest work to your own

Structuring can difficult unless you’re clear from the outset

Can deal with different aspects of the problem individually, potentially without messing up a chronological structure

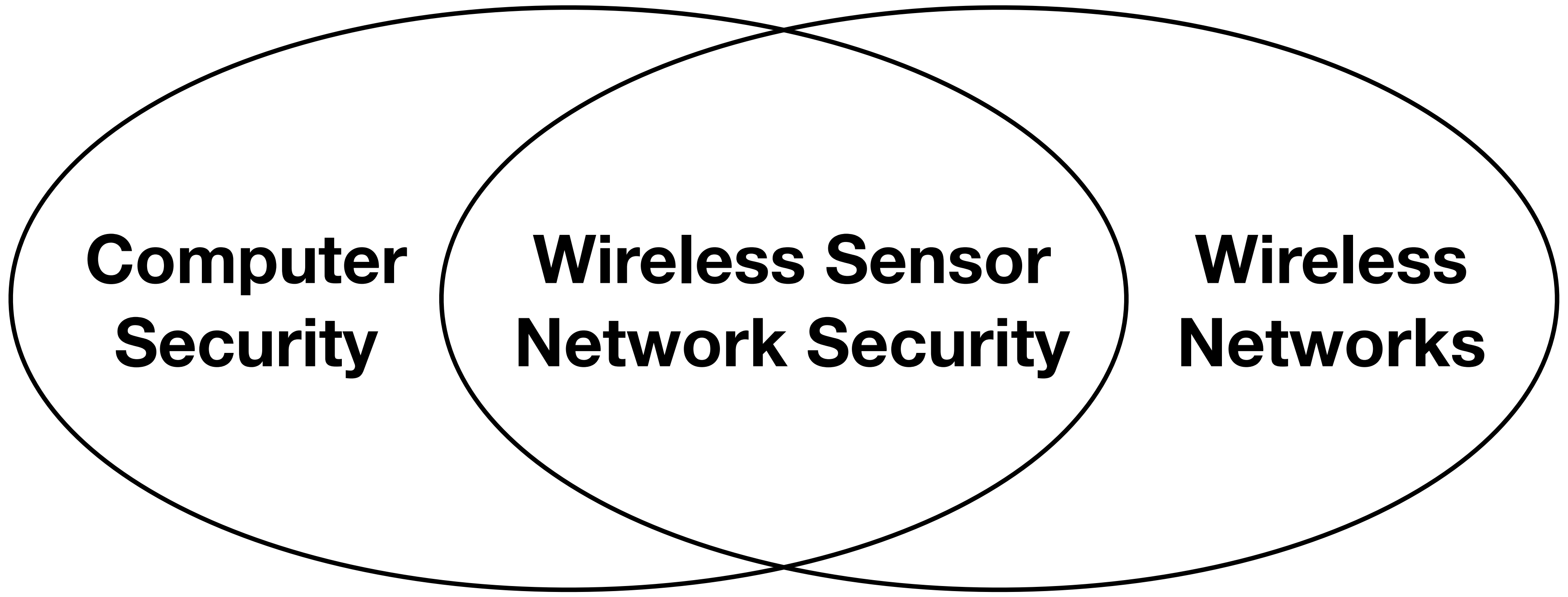
Fields of Research

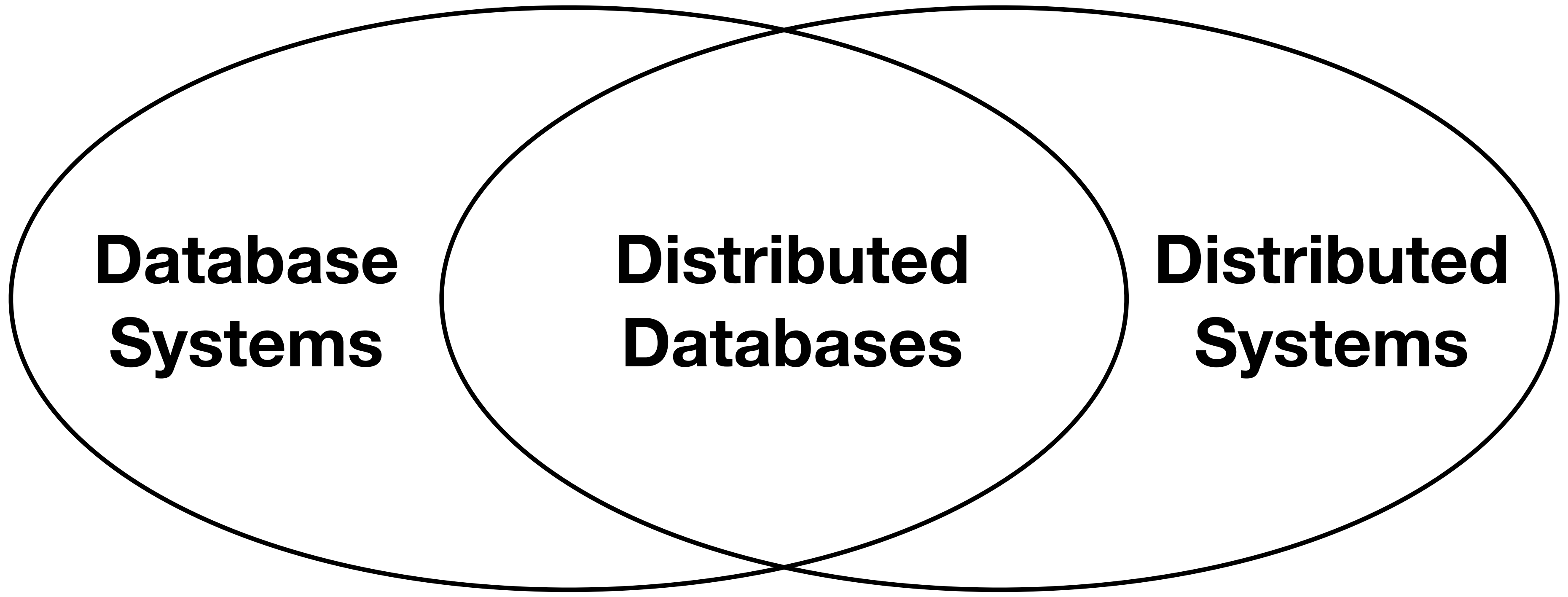
The concept of a research field is loosely defined

For example, it can be argued that Image Processing is a research field, but it cannot be said the same about Image Compression

Scientific researchers operate across many fields of research

Collaborative research is a product of overlapping interests





Fields of Research

Is Computer Science a field?

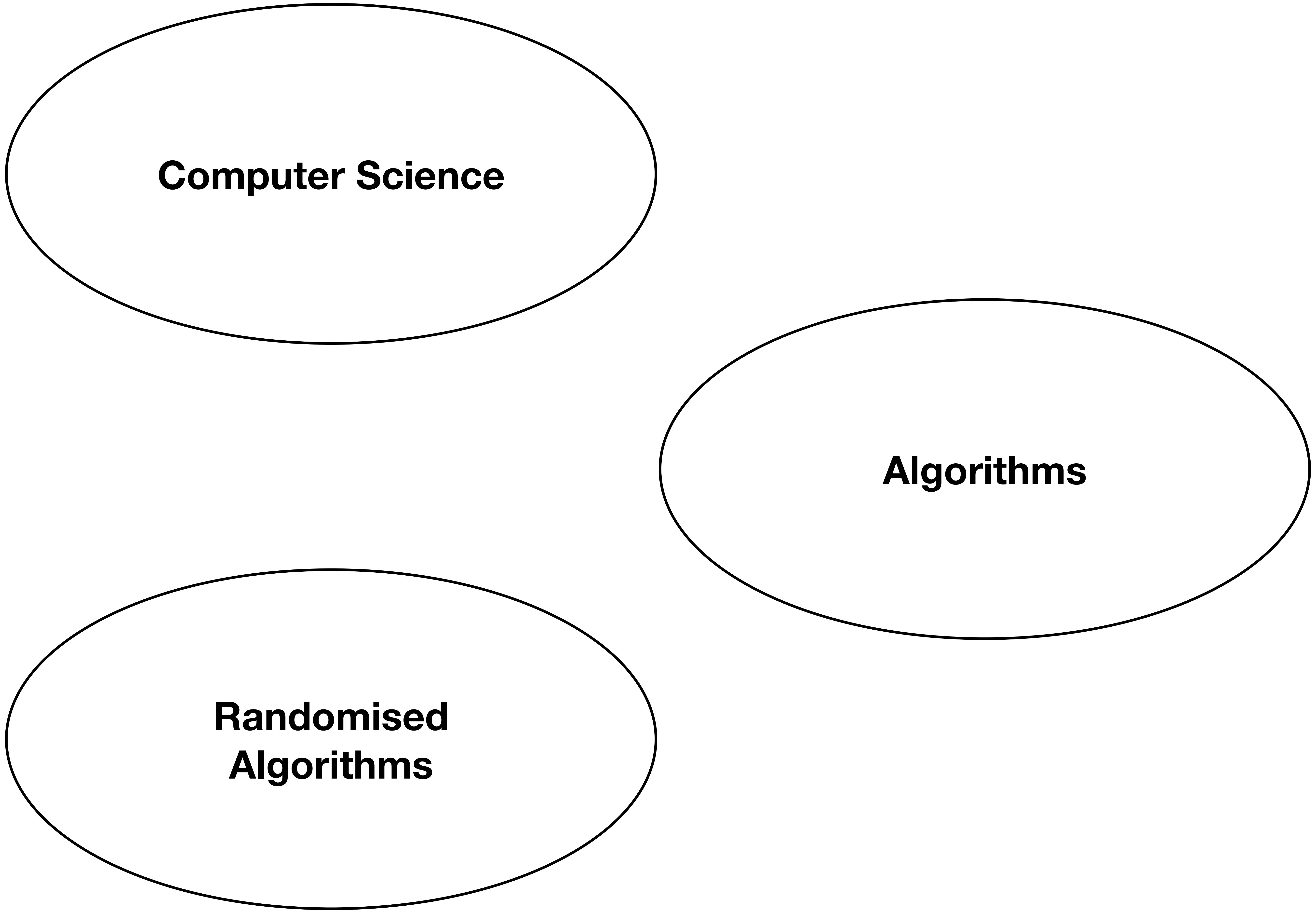
That’s very big!

Is Algorithms a field?

That’s pretty big!

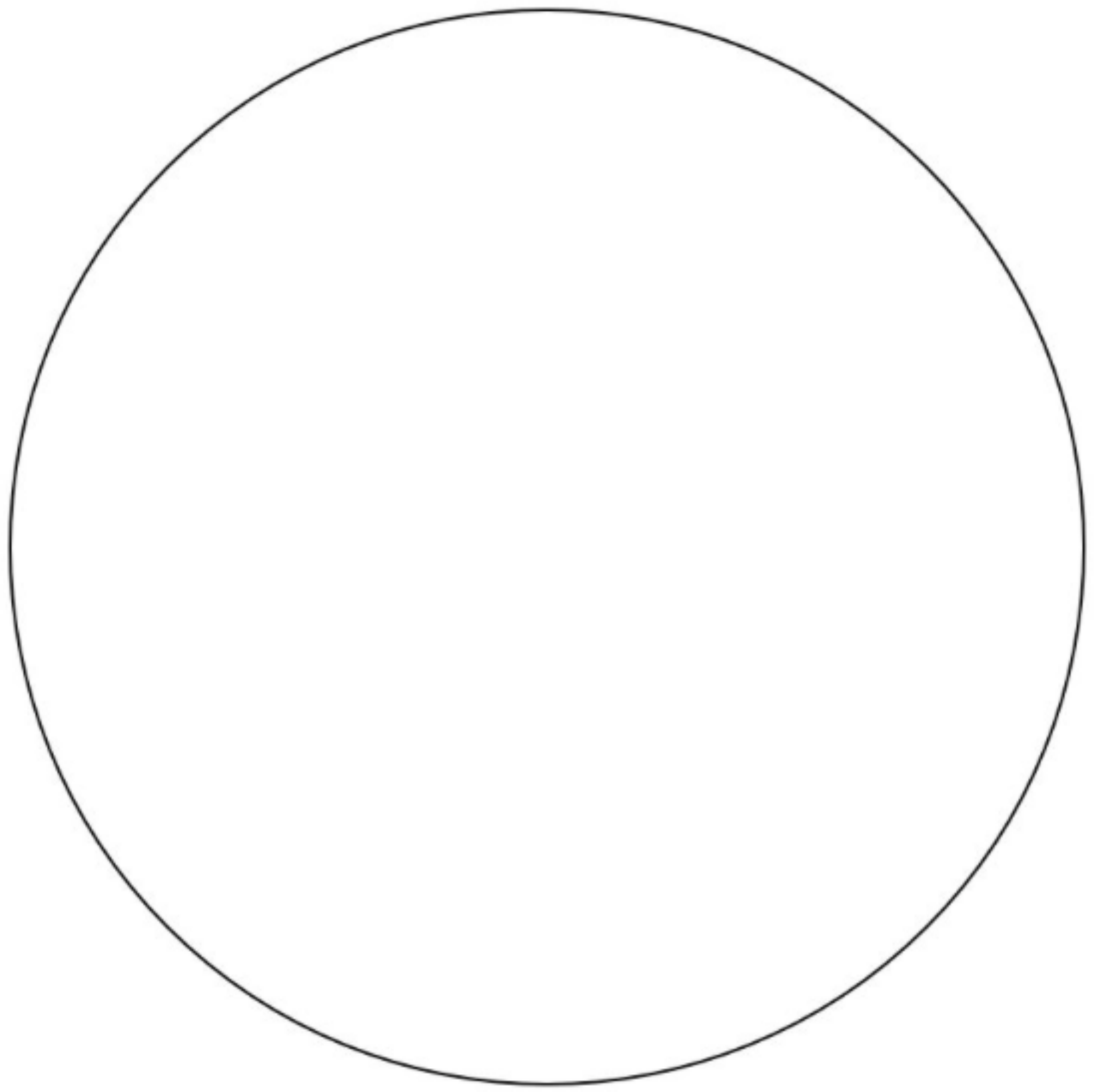
Is Randomised Algorithms a field?

That’s still pretty big! So…

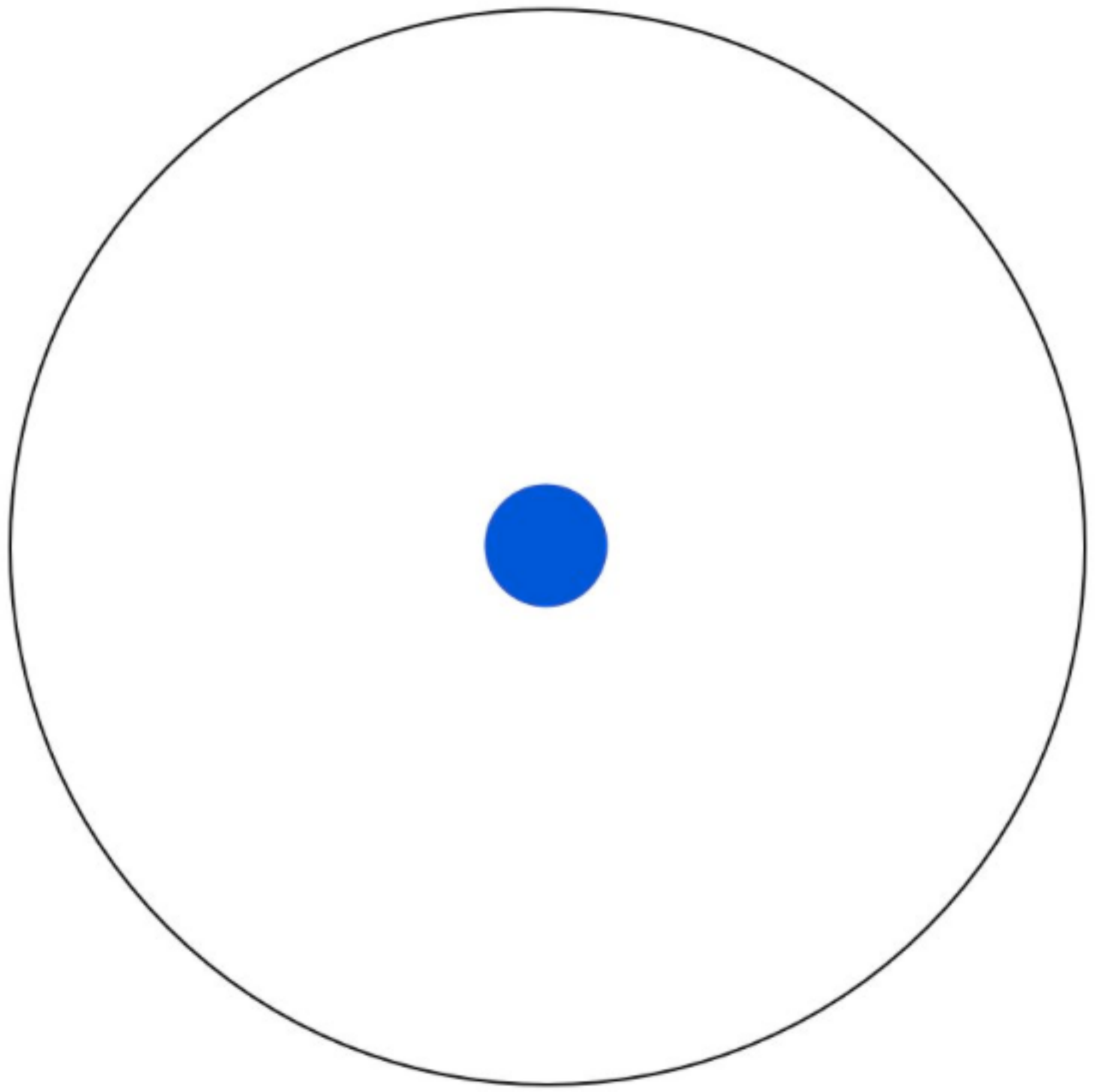


A View of Science

Imagine a circle that contains all of human knowledge:

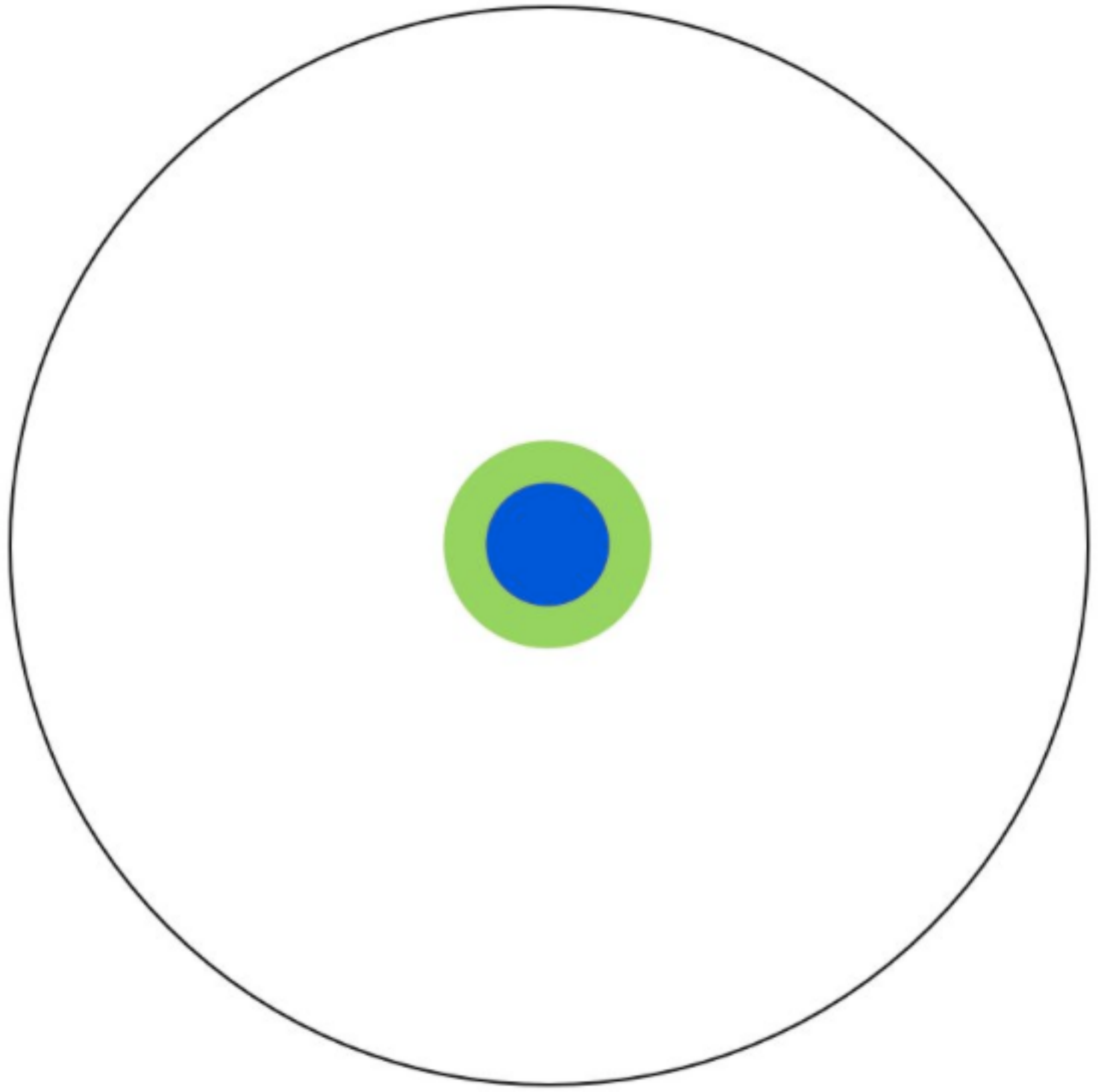


By the time you finish elementary school, you know a little:

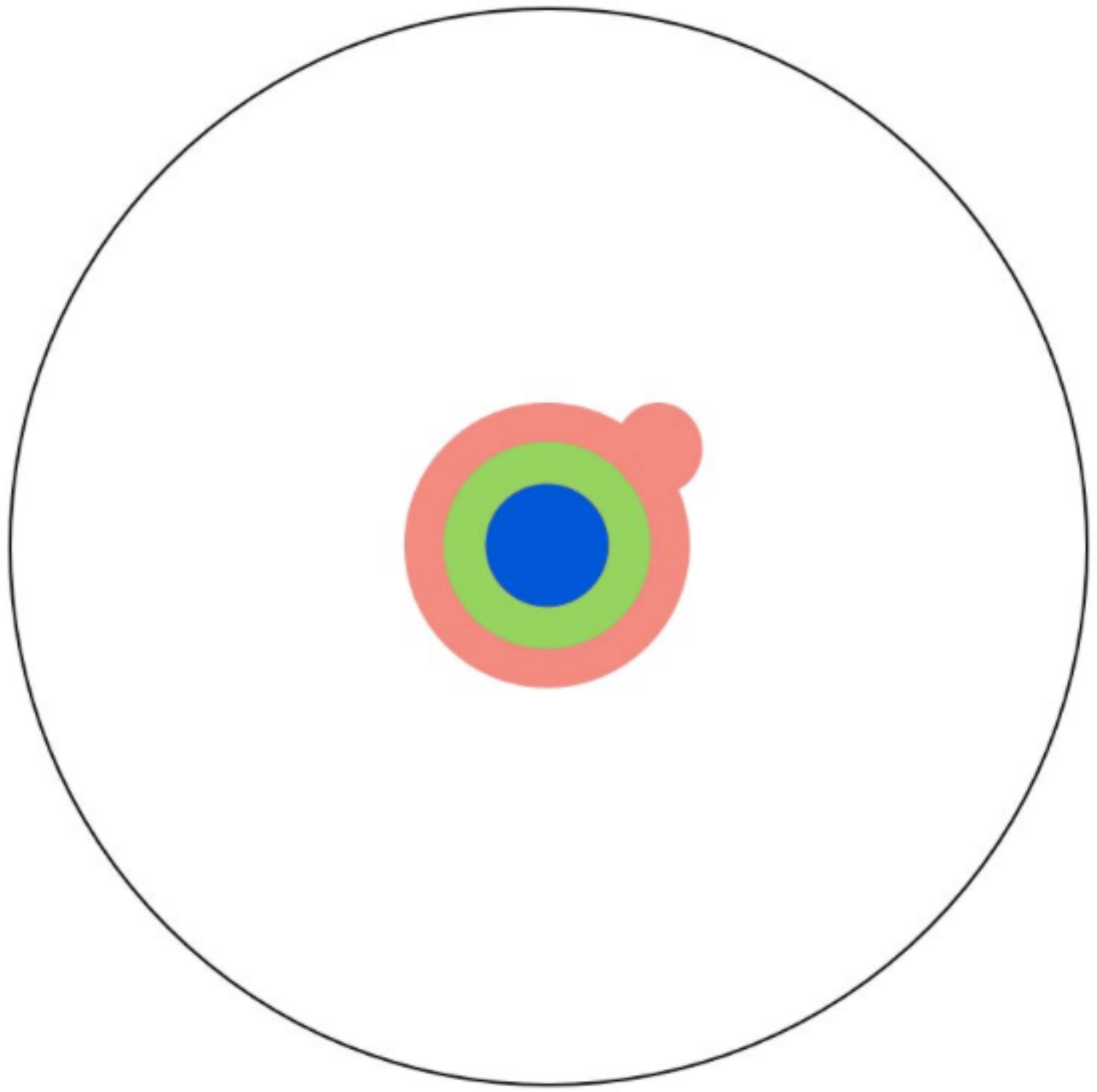


A View of Science

By the time you finish high school, you know a bit more:

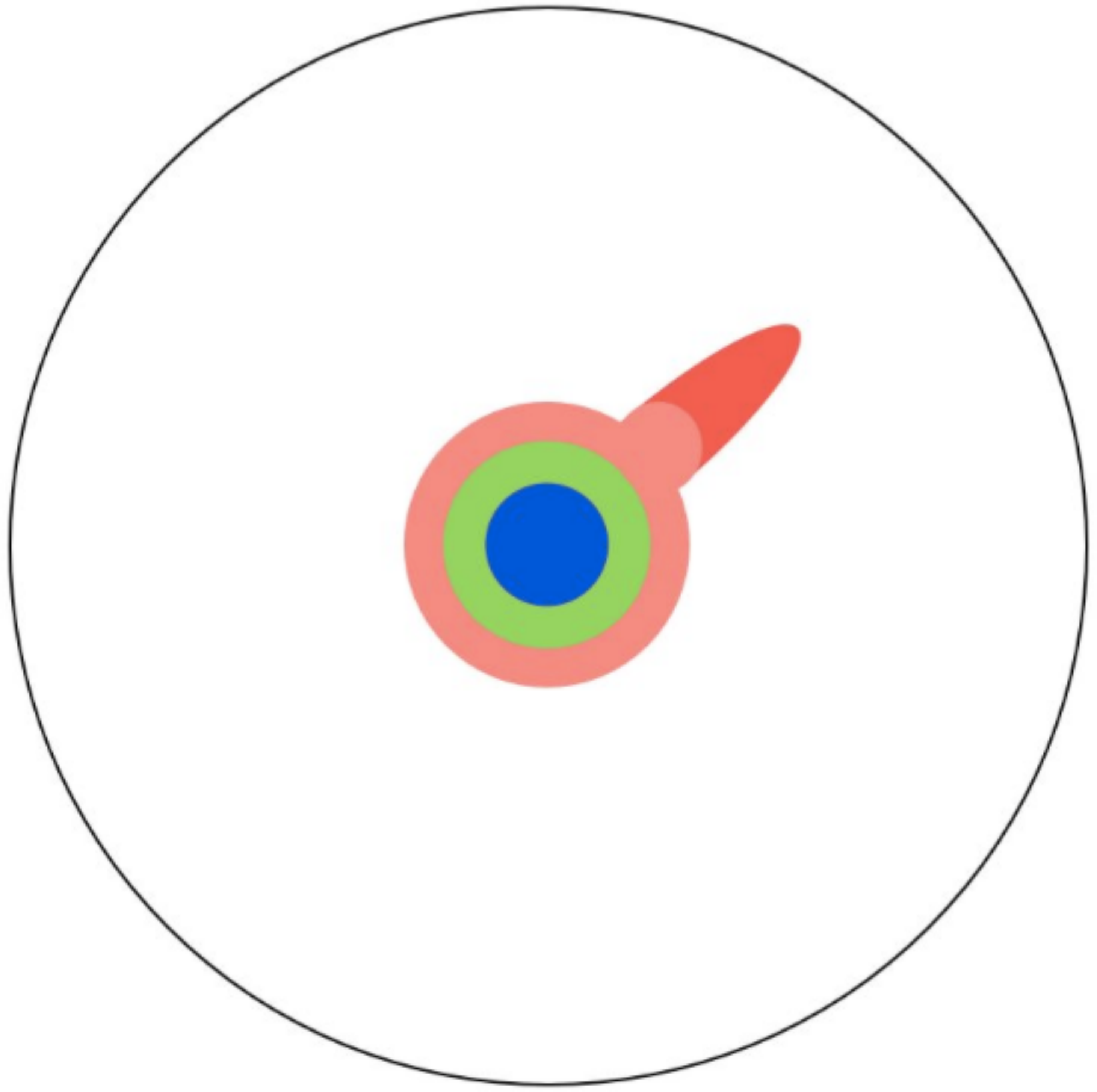


With a bachelor's degree, you gain a specialty:

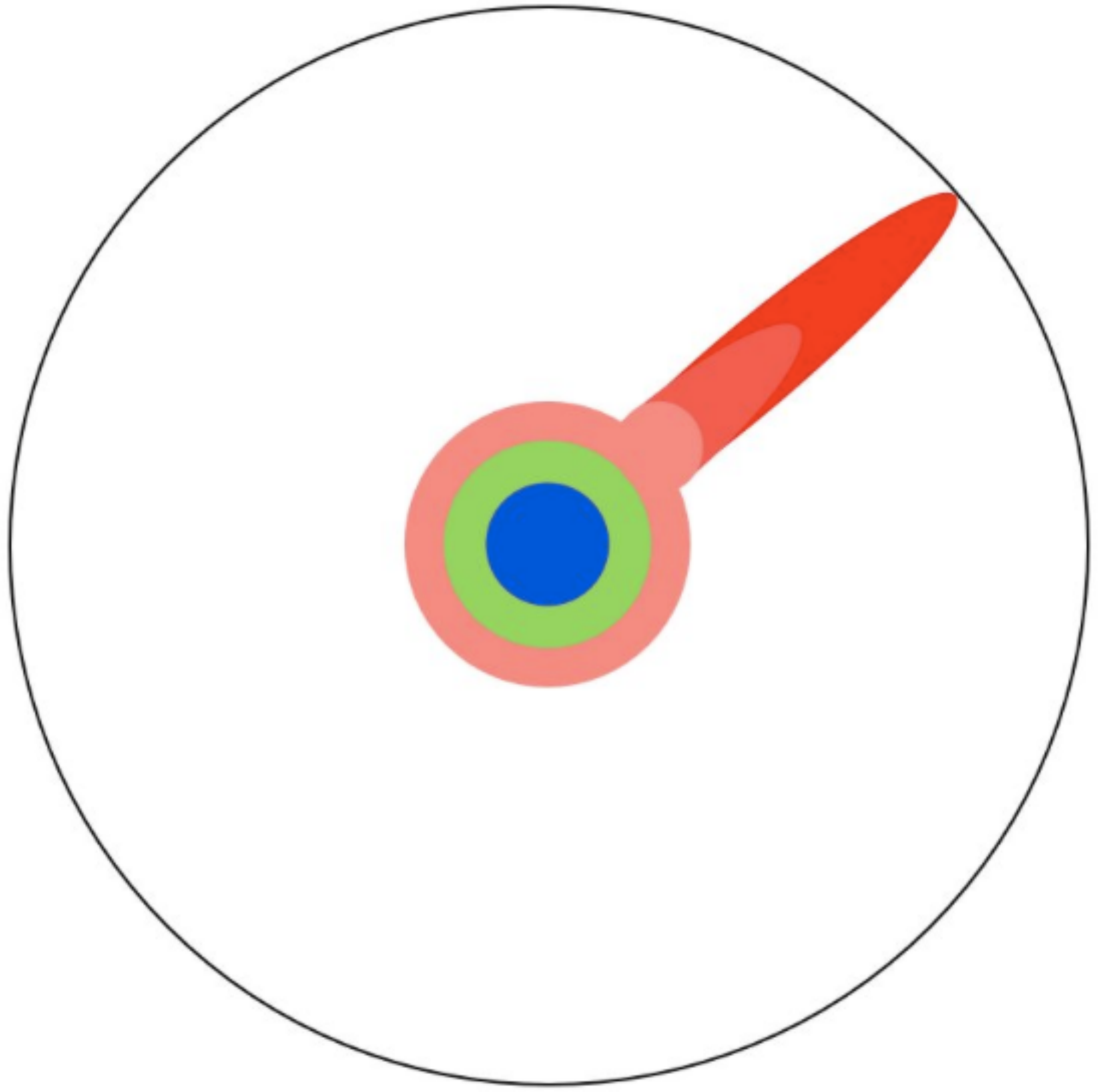


A View of Science

A master's degree deepens that specialty:

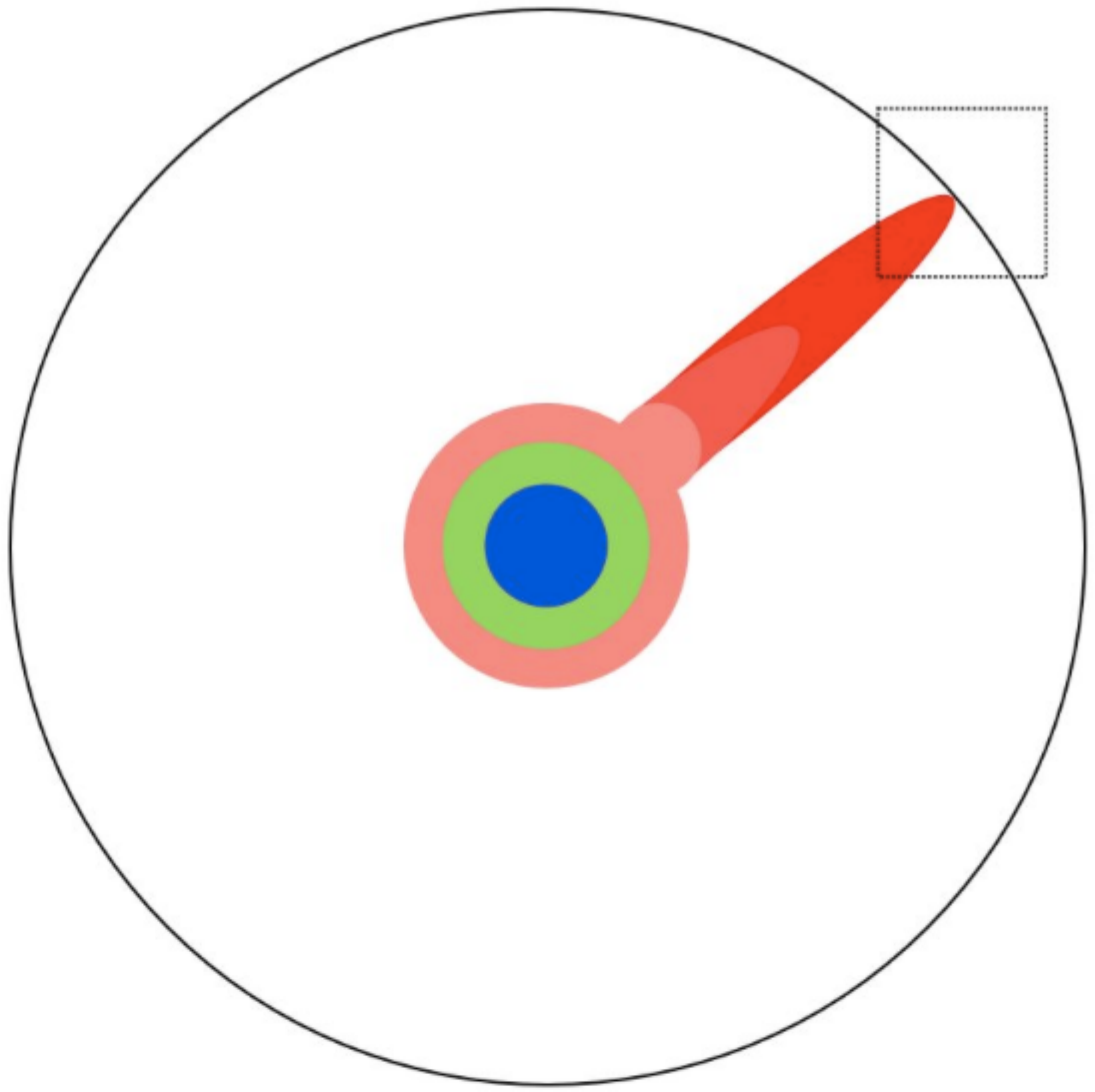


Reading research papers takes you to the edge of human knowledge:

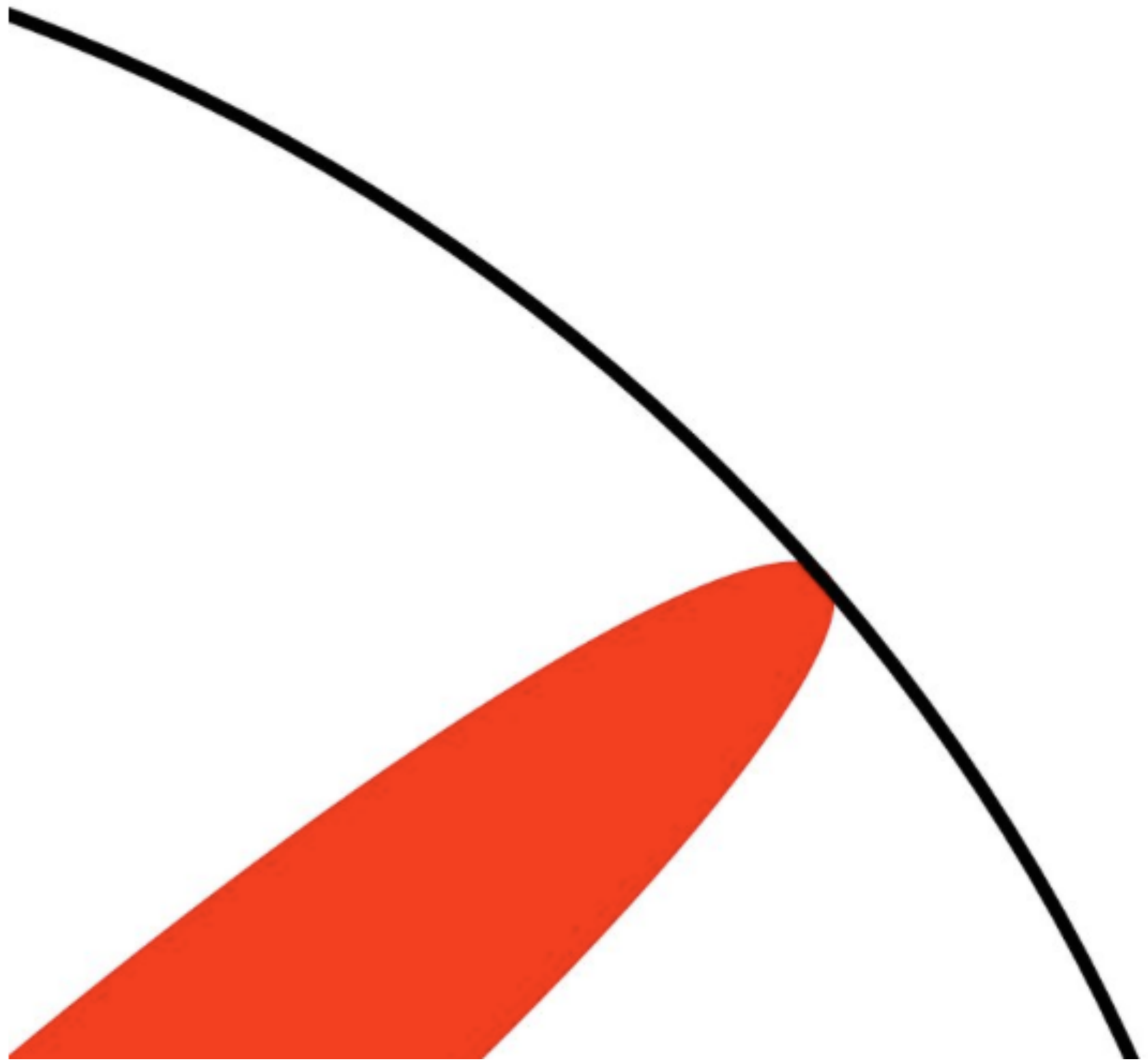


A View of Science

Once you're at the boundary, you focus:

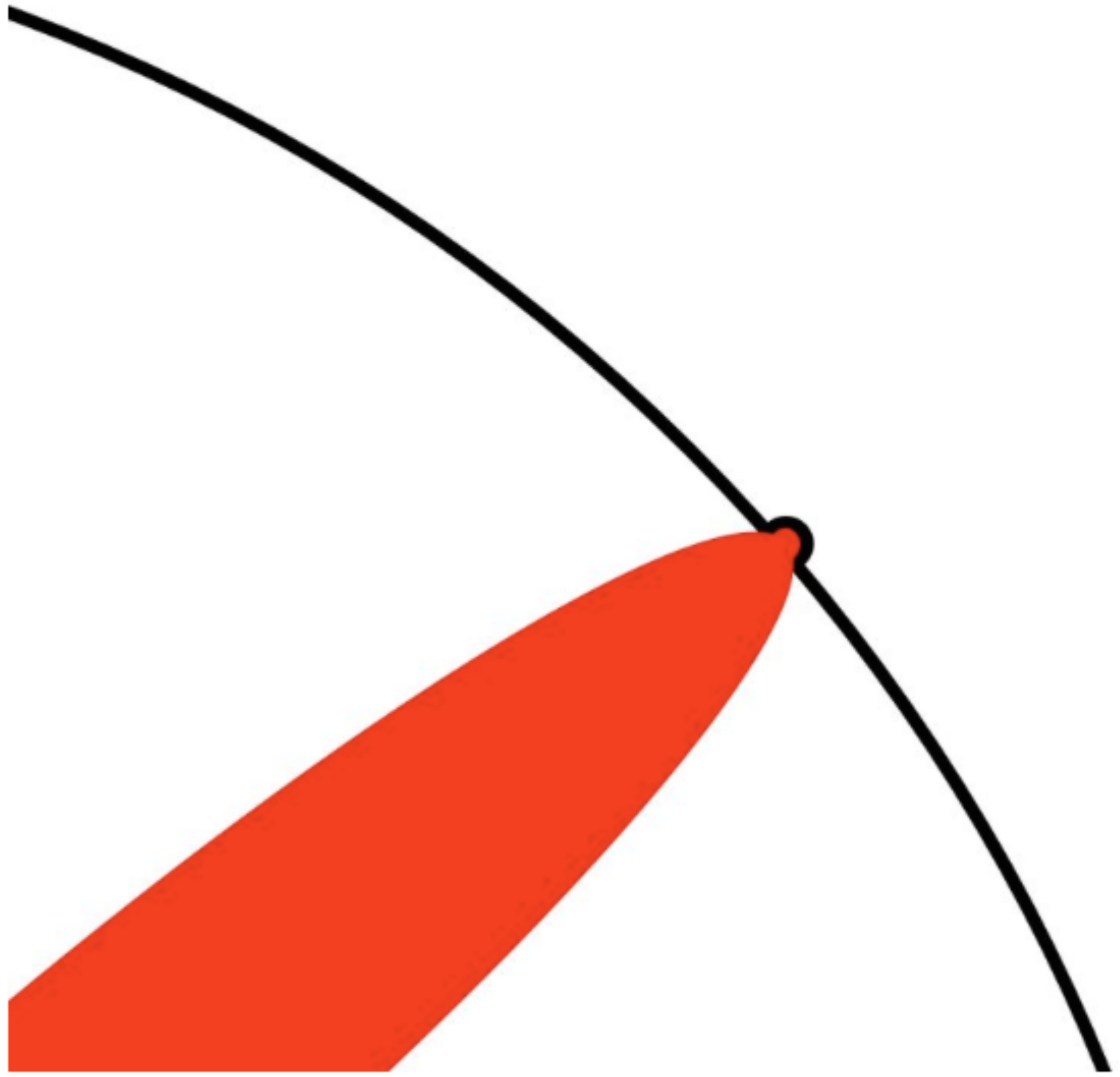


You push at the boundary for a few years:

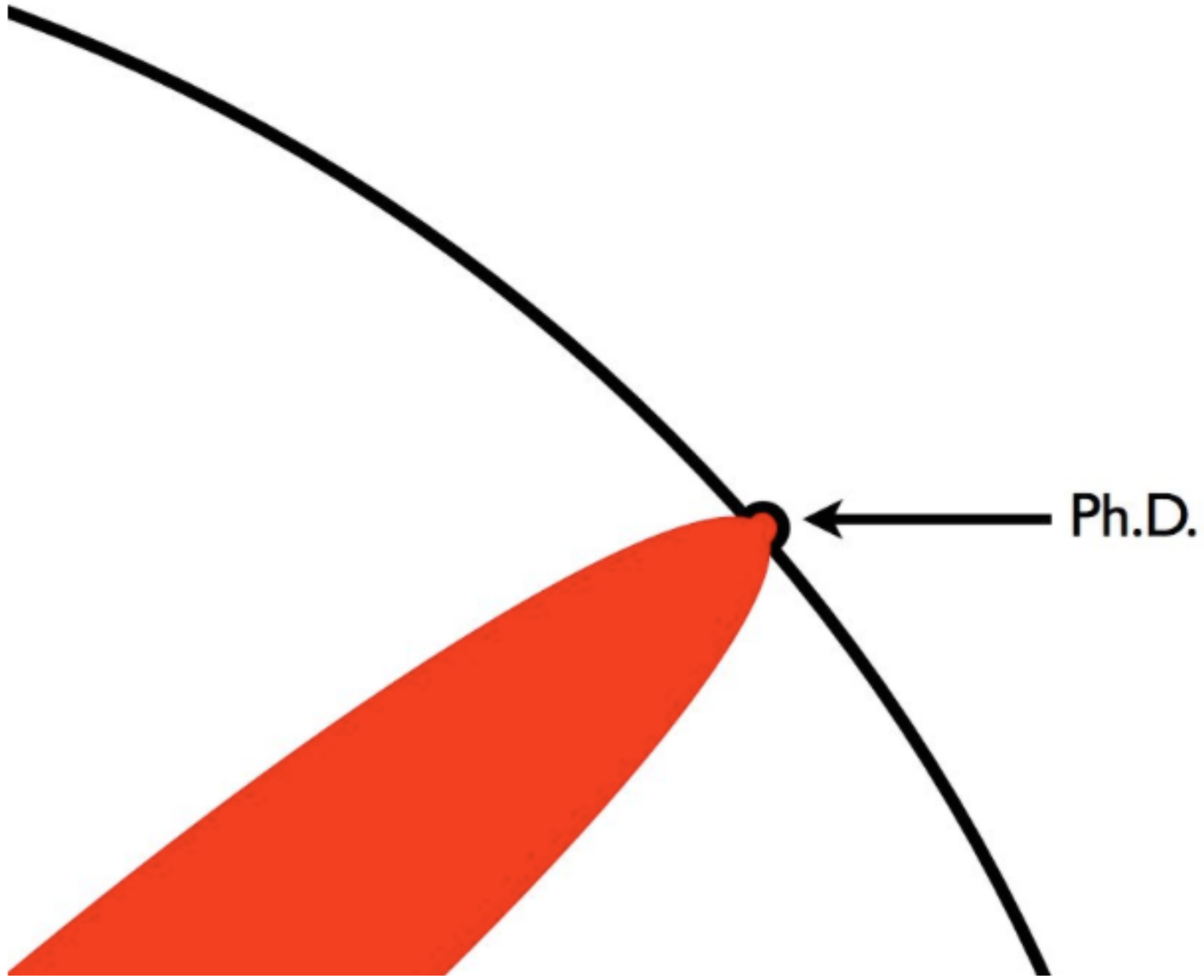


A View of Science

Until one day, the boundary gives way:



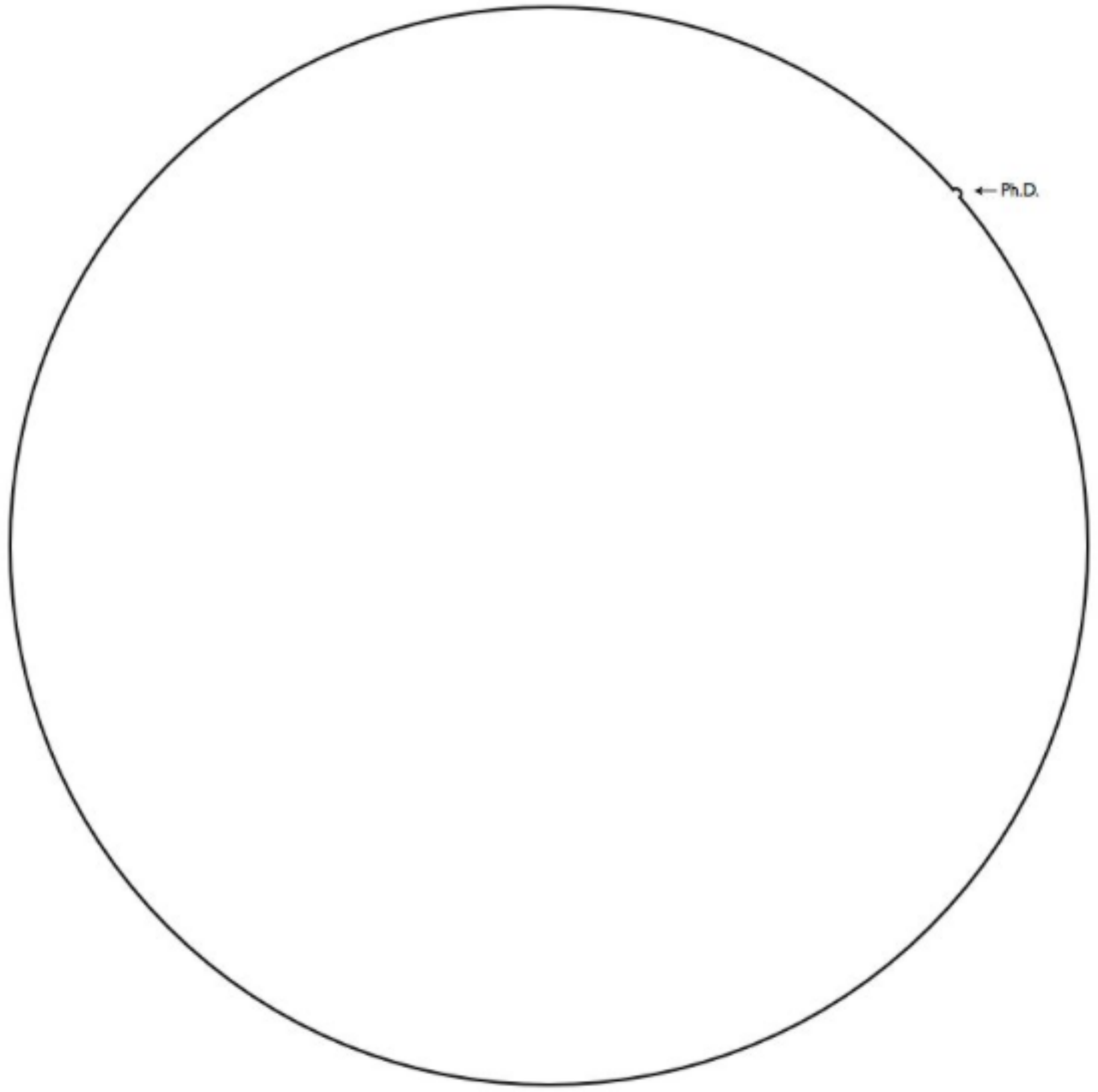
And, that dent you've made is called a Ph.D.:



A View of Science

Of course, the world looks different to you now:

So, don't forget the bigger picture:



Importance of Understanding a Research Field

To operate in a research field the work previously done in that field should be appreciated.

To this end

- Appreciate the significance of your research
- Position own work relative to existing research

Understanding a Research Field

Facts on their own are not sufficient

- Need to appreciate relationships between research conducted in the field

Critical analysis is fundamental

- Identifying research problems and potential advances
- Good to know that **"X showed Y"**, but even better to know that **"X showed Y by Z, which means A, B and C for the research being conducted"**

Understanding a Research Field

Fundamentally scientific research is judged based on **significance** and **quality of contribution**

- Positioning allows others to appreciate the significance of the research field and the quality of contributions
- Excellent research that is poorly positioned has no meaning

How to Start Writing a Literature Review

Read to identify the research field

Read to identify the focus of the review within that research field

Read and analyse with respect to appropriate measures and themes

Read and **reread** as the review is built up

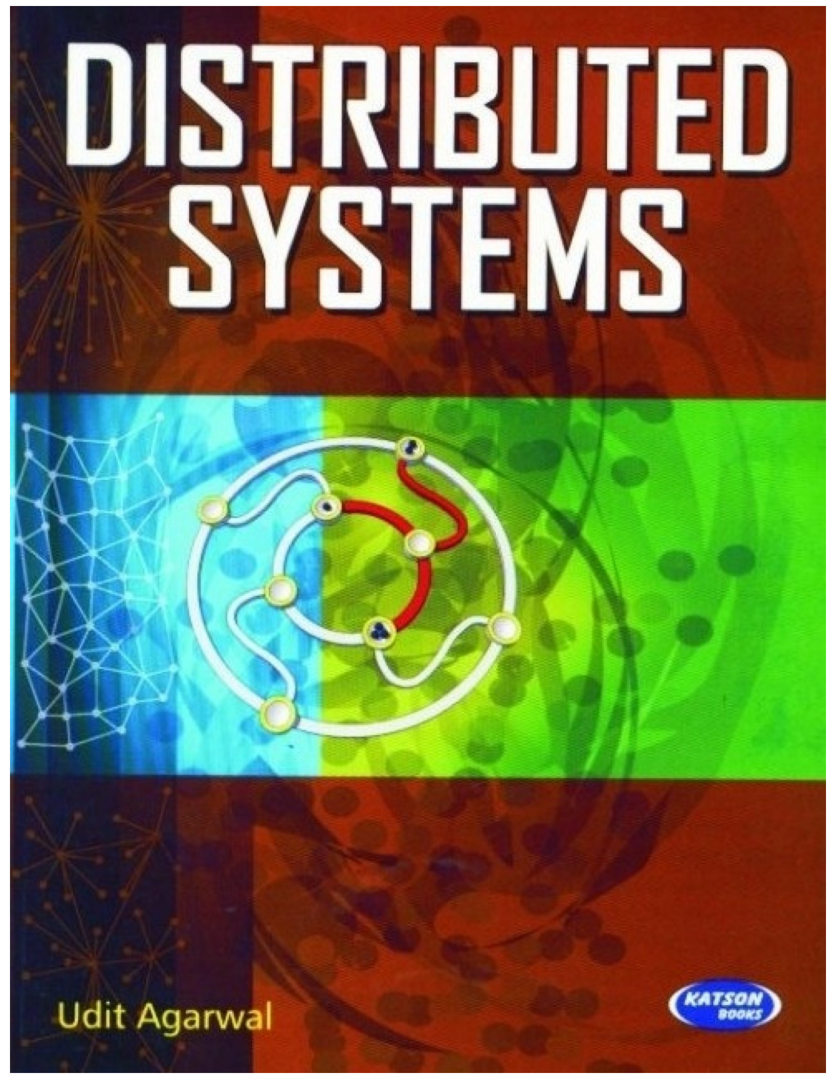
This should ensure a basis for further discussions or analysis to own research

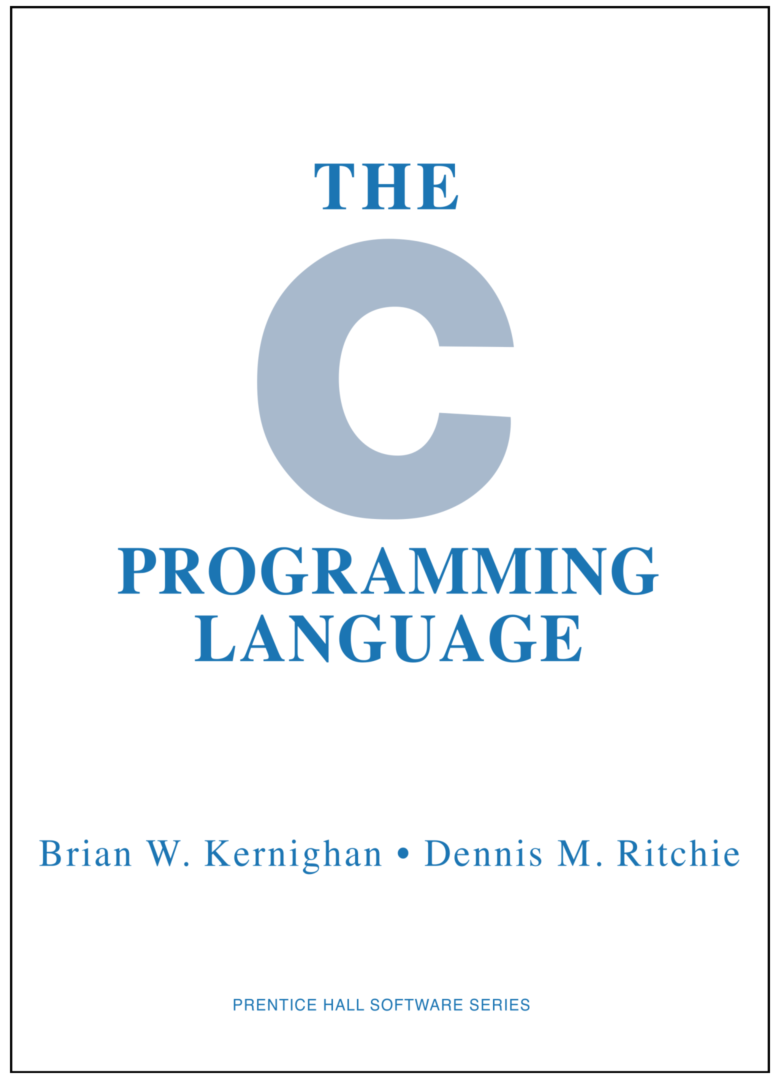
Literature Sources

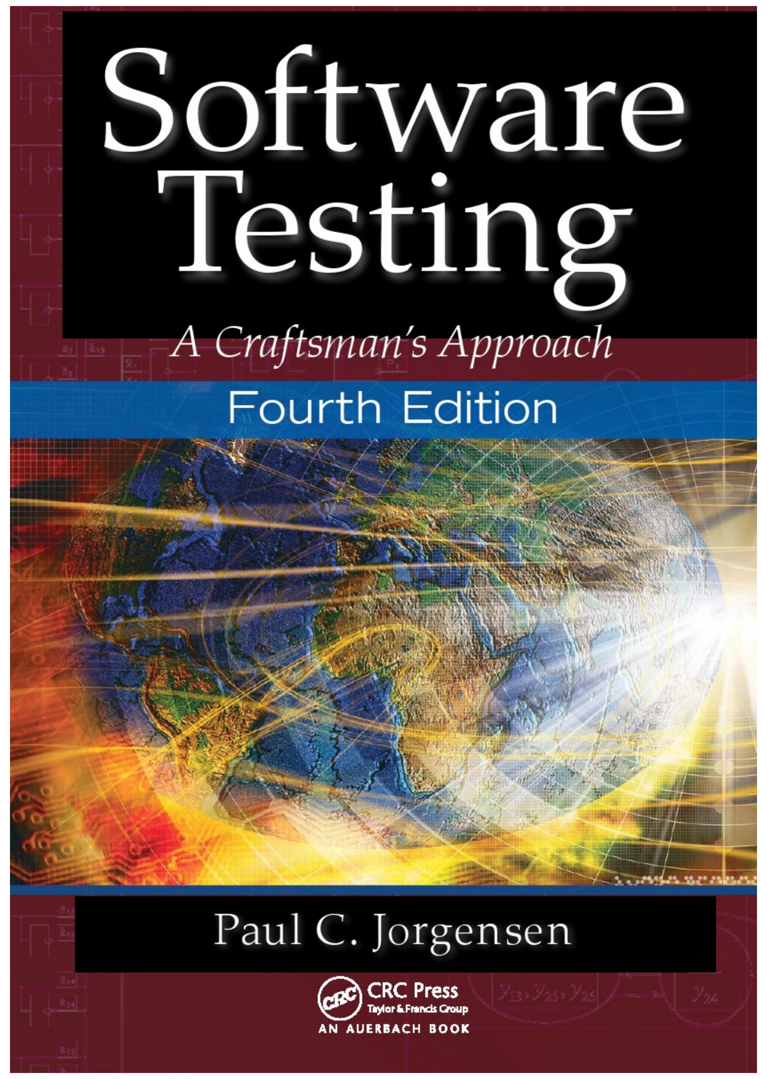
The term "literature" should be understood as content produced by researchers for other researchers

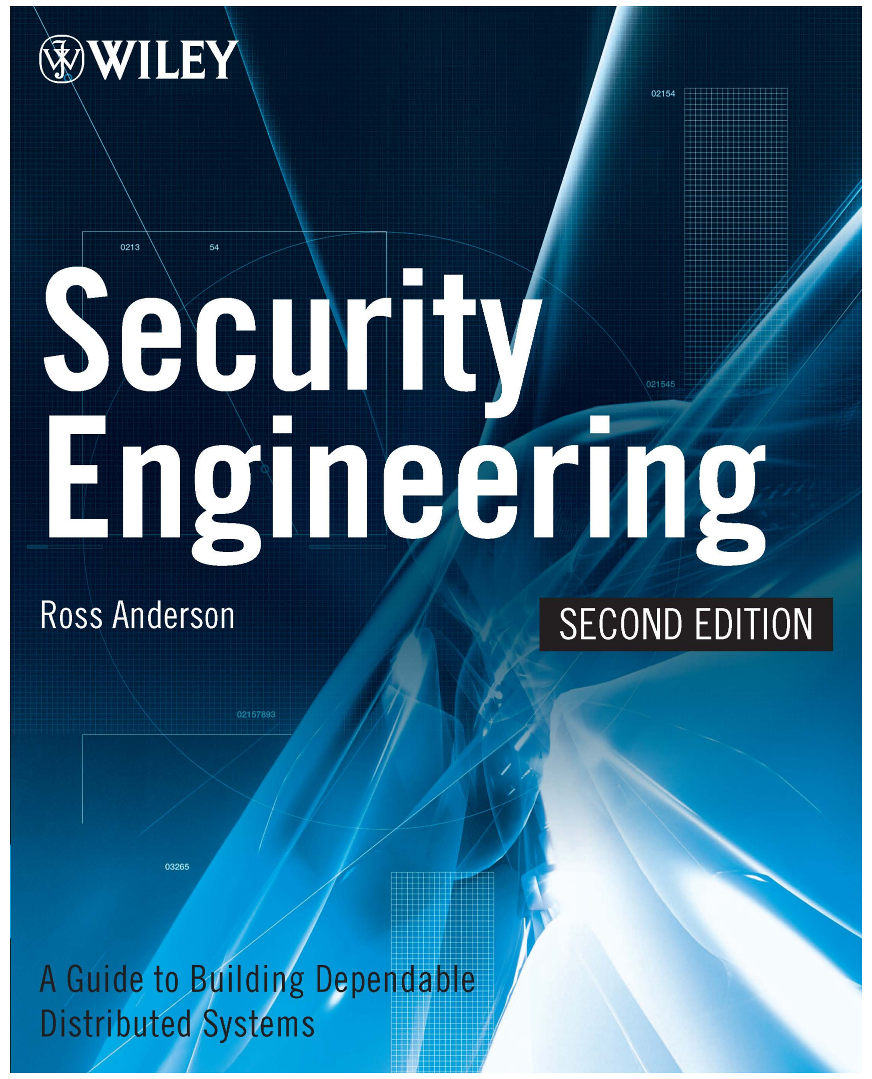
- Books
- Conference and journal papers
- Dissertations, theses and technical reports
- Patents
- Technological standards

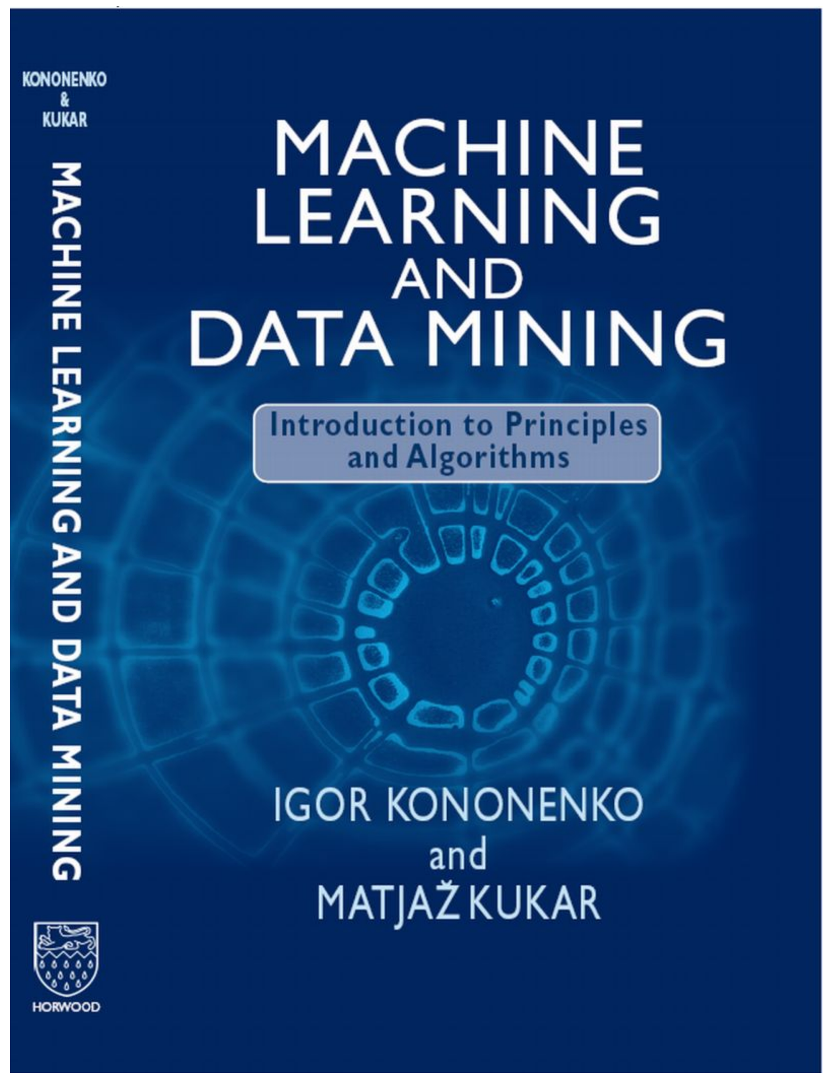
Books


DISTRIBUTED SYSTEMS
Udit Agarwal


THE C PROGRAMMING LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE HALL SOFTWARE SERIES


Software Testing
A Craftsman's Approach
Fourth Edition
Paul C. Jorgensen
CRC Press


WILEY
Security Engineering
Ross Anderson
SECOND EDITION
A Guide to Building Dependable Distributed Systems


MACHINE LEARNING AND DATA MINING
Introduction to Principles and Algorithms
IGOR KONONENKO and MATJAŽ KUKAR
MACHINE LEARNING AND DATA MINING

Conference and Journal Papers

IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. X, NO. X, X 201X 1

Loop Tiling in Large-Scale Stencil Codes at Run-time with OPS

István Z. Reguly, *Member, IEEE,* Gihan R. Mudalige and Michael B. Giles

Abstract—The key common bottleneck in most stencil codes is data movement, and prior research has shown that improving data locality through optimisations that optimise across loops do particularly well. However, in many large PDE applications it is not possible to apply such optimisations through compilers because there are many options, execution paths and data per grid point, many dependent on run-time parameters, and the code is distributed across different compilation units. In this paper, we adapt the data locality improving optimisation called tiling for use in large OPS applications both in shared-memory and distributed-memory systems, relying on run-time analysis and delayed execution. We evaluate our approach on a number of applications, observing speedups of $2\times$ on the Cloverleaf 2D/3D proxy applications, which contain 83(2D)/141(3D) loops, $3.5\times$ on the linear solver TeaLeaf, and $1.7\times$ on the compressible Navier-Stokes solver OpenSBLI. We demonstrate strong and weak scalability on up to 4608 cores of CINECA's Marconi supercomputer. We also evaluate our algorithms on Intel's Knights Landing, demonstrating maintained throughput as the problem size grows beyond 16GB, and we do scaling studies up to 8704 cores. The approach is generally applicable to any stencil DSL that provides per loop nest data access information.

Index Terms—DSL, Tiling, Cache Blocking, Memory Locality, OPS, Stencil, Structured Mesh

1 Introduction

Modern architectures now include ever-larger on-chip caches to help exploit spatial and temporal locality in memory accesses: latency and energy benefits of accessing data from cache can be up to 10x compared to accessing it with a load from off-chip memory. Unfortunately, most scientific simulations are structured in a way that limits locality: the code is structured as a sequence of computations, each streaming a number of data arrays from memory,

on the stencil pattern, leading to wavefront schemes.

There is a large body of research on the combination of fusion and loop schedule optimisations [4], [5], [6]: techniques that extend loop blocking to work across subsequent loop nests, generally called *tiling*. Tiling carries out dependency analysis similar to what is required for loop fusion, but instead of fusing the bodies of subsequent loops, it

Beyond 16GB: Out-of-Core Stencil Computations

Istán Z Reguly
Faculty of Information Technology and Bionics,Pázmány Péter Catholic University
Budapest, Hungary
reguly.istvan@itk.ppke.hu

Gihan R. Mudalige
Department of Computer Science, University of Warwick
Coventry, UK
g.mudalige@warwick.ac.uk

Michael B. Giles
Maths Institute, University of Oxford
Oxford, UK
mike.giles@maths.ox.ac.uk

ABSTRACT

Stencil computations are a key class of applications, widely used in the scientific computing community, and a class that has particularly benefited from performance improvements on architectures with high memory bandwidth. Unfortunately, such architectures come with a limited amount of fast memory, which is limiting the size of the problems that can be efficiently solved. In this paper, we address this challenge by applying the well-known cache-blocking tiling technique to large scale stencil codes implemented using the OPS domain specific language, such as CloverLeaf 2D, CloverLeaf 3D, and OpenSBLI. We introduce a number of techniques and optimisations to help manage data resident in fast memory, and minimise data movement. Evaluating our work on Intel's Knights Landing Platform as well as NVIDIA P100 GPUs, we demonstrate that it is possible to solve 3 times larger problems than the on-chip memory size with at most 15% loss in efficiency.

ACM Reference format:
Istán Z Reguly, Gihan R. Mudalige, and Michael B. Giles. 2017. Beyond 16GB: Out-of-Core Stencil Computations. In *Proceedings of XXX, XXX, XXX,*

resident for the entirety of the application - yielding the aforementioned size limitation.

In data streaming type applications, where a chunk of data is uploaded, processed, then downloaded, the workload (usually larger than GPU memory) is partitioned into small chunks, so it's possible to overlap copies in both directions and computations. To efficiently utilise accelerator bandwidth, this also means that any data uploaded has to be accessed about as many times as this ratio between upload bandwidth and accelerator bandwidth; otherwise performance will be limited by upload speed. To efficiently utilise the accelerator's computational resources, the ratio is even more extreme: for a P100 GPU one would need to carry out about 2500 floating point operations for every float variable uploaded (10 TFlops/s, 16 GB/s PCI-e BW, 4 bytes/float).

Going into the exascale era, most of the upcoming large supercomputers will be built with chips featuring on-chip high-bandwidth memory: Intel's Knight's Landing and later generations have at least 16GB MCDRAM, with bandwidths over 500 GB/s, and NVIDIA's P100 and later GPUs also feature 16GB with 720 GB/s and more. To tackle the issue with slow upload speeds, both have moved away

Achieving Performance Portability for a Heat Conduction Solver Mini-Application on Modern Multi-core Systems

R. O. Kirk*, G. R. Mudalige*, I. Z. Reguly†, S. A. Wright*, M. J. Martineau‡, S. A. Jarvis*
* Department of Computer Science, University of Warwick, Coventry, UK
{R.Kirk, G.Mudalige, Steven.Wright, S.A.Jarvis}@warwick.ac.uk
† Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, Budapest, Hungary
reguly.istvan@itk.ppke.hu
‡ Department of Computer Science, Merchant Venturers Building, Woodland Road, Clifton, Bristol, UK
m.martineau@bristol.ac.uk

***Abstract*—Modernizing production-grade, often legacy applications to take advantage of modern multi-core and many-core architectures can be a difficult and costly undertaking. This is especially true currently, as it is unclear which architectures will dominate future systems. The complexity of these codes can mean that parallelisation for a given architecture requires significant re-engineering. One way to assess the benefit of such an exercise would be to use mini-applications that are representative of the legacy programs.**

In this paper, we investigate different implementations of TeaLeaf, a mini-application from the Mantevo suite that solves the linear heat conduction equation. TeaLeaf has been ported to use many parallel programming models, including OpenMP, CUDA and MPI among others. It has also been re-engineered to use the OPS embedded DSL and template libraries Kokkos and RAJA. We use these different implementations to assess the performance portability of each technique on modern multi-core systems.

models and languages, and then maintaining these different versions, is infeasible.

One common strategy is to use small representative applications to test and evaluate new technologies, programming models, frameworks and optimisations. The use of such programs, called proxy or mini-applications, is not new. The idea can be traced to the development of small benchmark codes such as LINPACK [1] and the NAS Parallel Benchmarks [2]. More recent efforts include the Mantevo [3] and UK Mini-Application Consortium [4] suites. Due to their small size, mini-apps are much more manageable than production applications and can feasibly be re-written in different programming languages and with specific optimisations. They are also unrestricted and/or devoid of any commercially sensitive code, allowing them to be readily distributed to many parties and sites.

Auto-Vectorizing a Large-scale Production Unstructured-mesh CFD Application

G.R. Mudalige
Oxford eResearch Centre, University of Oxford,U.K.
gihan.mudalige@oerc.ox.ac.uk

I.Z. Reguly
Faculty of Information Technology and Bionics
Pázmány Péter Catholic University, Hungary
reguly.istvan@itk.ppke.hu

M.B. Giles
Oxford e-Research Centre, University of Oxford, U.K.
mike.giles@maths.ox.ac.uk

ABSTRACT

For modern x86 based CPUs with increasingly longer vector lengths, achieving good vectorization has become very important for gaining higher performance. Using very explicit SIMD vector programming techniques has been shown to give near optimal performance, however they are difficult to implement for all classes of applications particularly ones with very irregular memory accesses and usually require considerable re-factorisation of the code. Vector intrinsics are also not available for languages such as Fortran which is still heavily used in large production applications. The alternative is to depend on compiler auto-vectorization which usually have been less effective in vectorizing codes with irregular memory access patterns. In this paper we present recent research exploring techniques to gain compiler auto-vectorization for unstructured mesh applications. A key contribution is details on software techniques that achieve auto-vectorisation for a large production grade un-

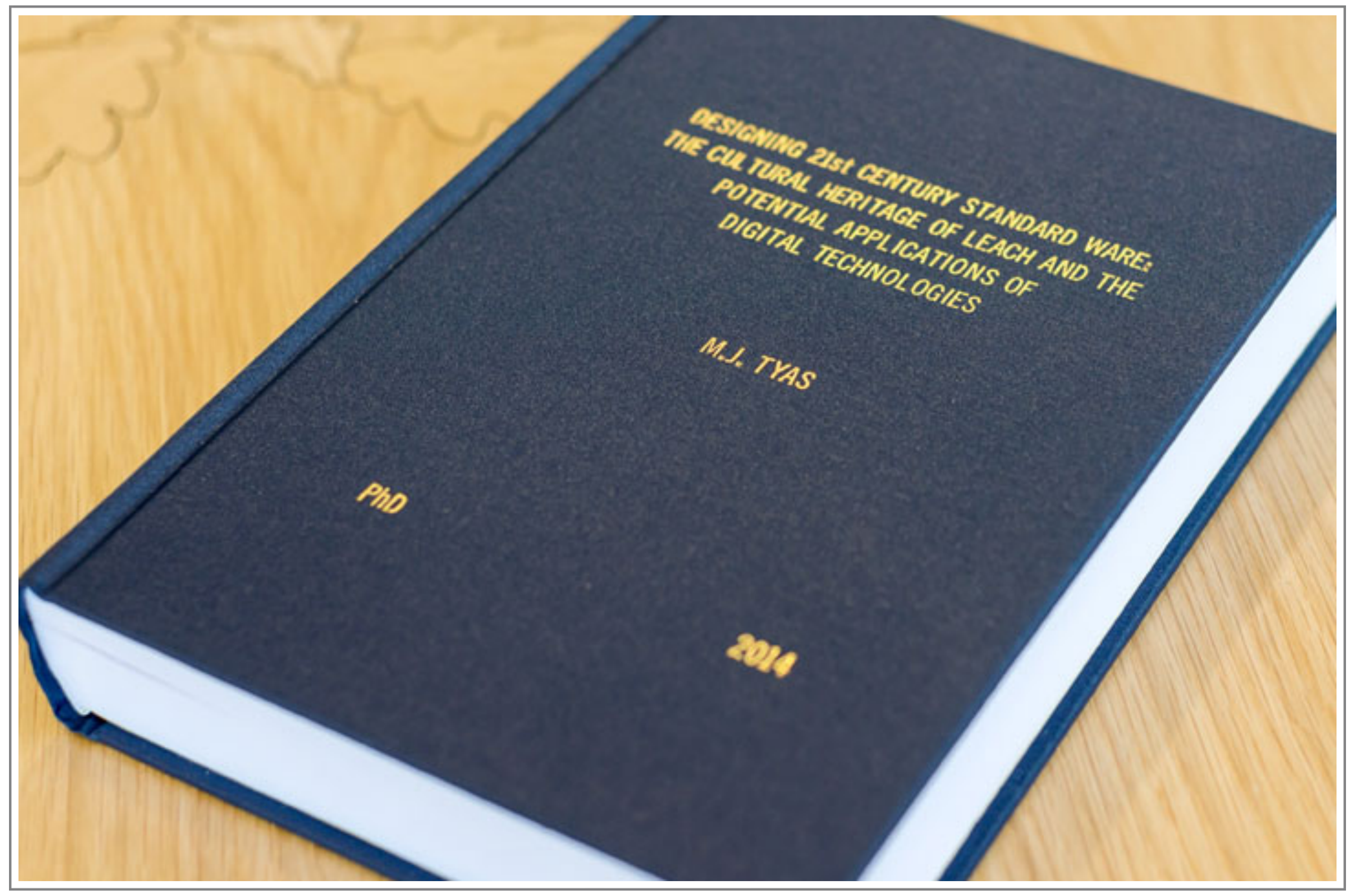
Keywords

Automatic Vectorization, SIMD, OP2, Unstructured mesh, GPU

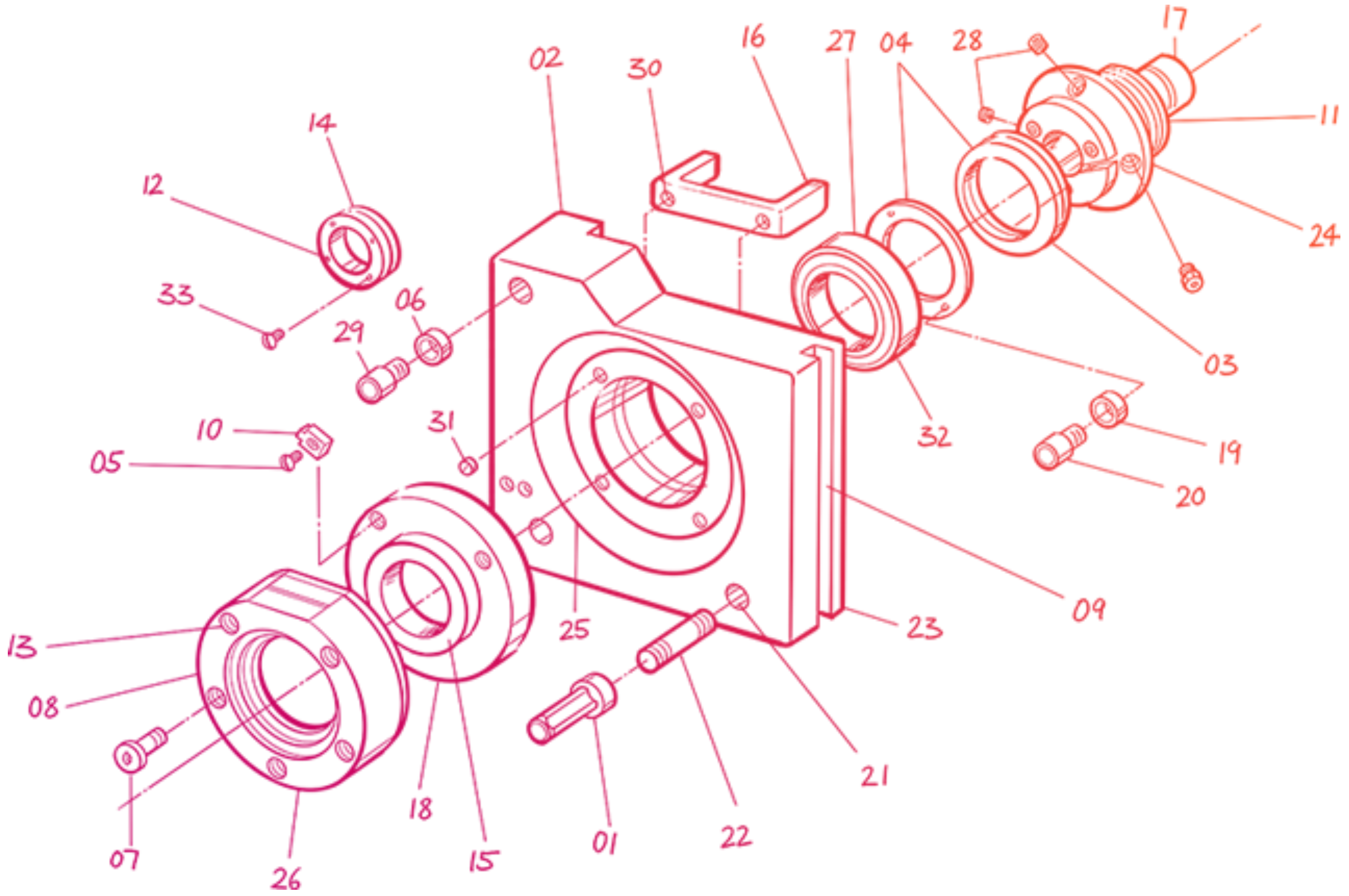
1. INTRODUCTION

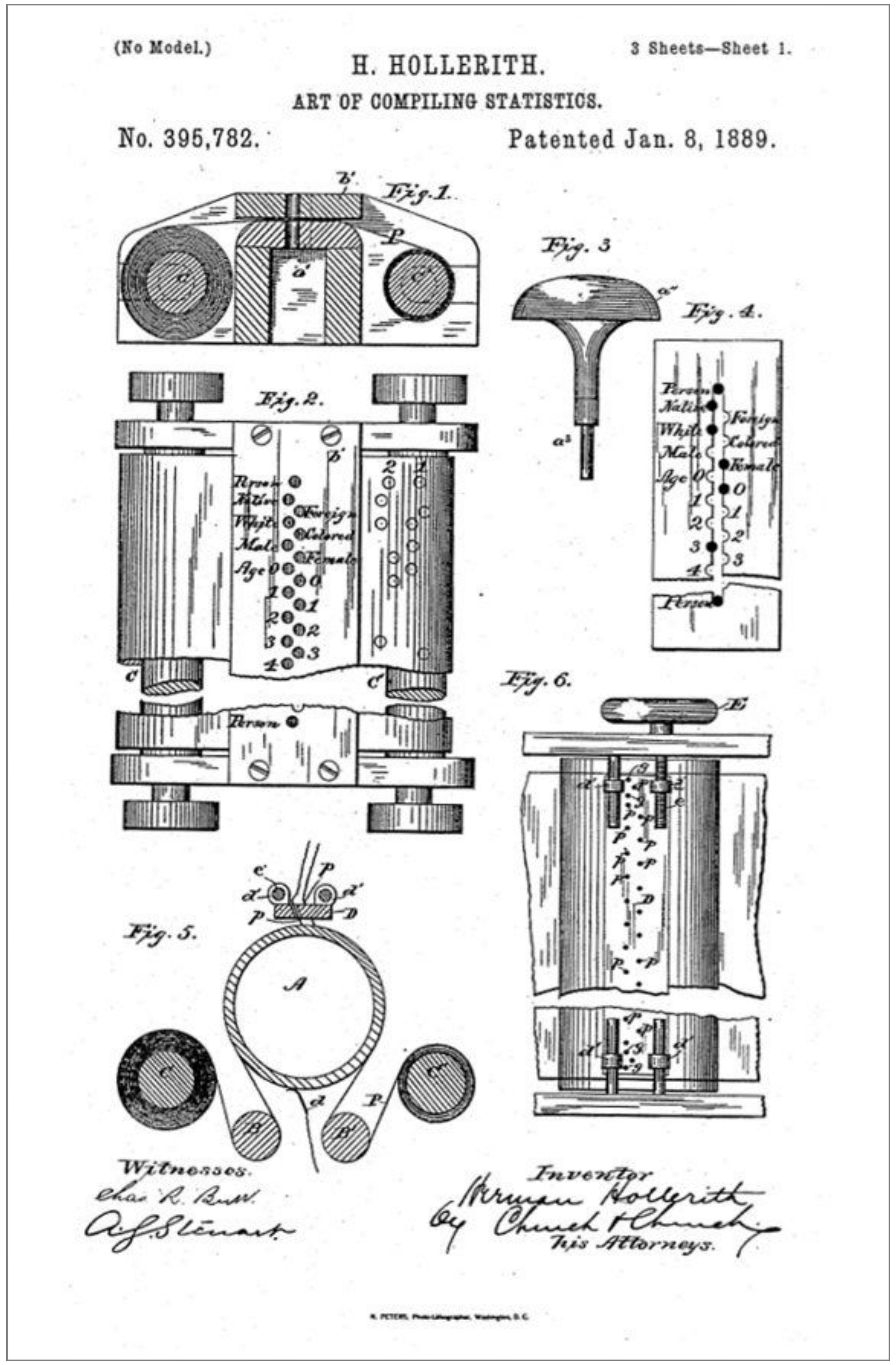
The importance of achieving good vectorisation was not significant in previous generations of micro-processors. Vector lengths were short (128 bits or less) and compiler auto-vectorization was used to gain modest speedups if opportunities were present in the application code. Even then such compilers at best could only vectorize a few classes of applications that had very regular memory access and computation patterns such as from the structured-mesh or multimedia application domains. However, a significant portion of the capabilities of the latest processors depends on the utilization of their vector units. Particularly for modern x86 based CPUs with increasingly longer vector lengths, achiev-

Dissertations, Theses and Technical Reports

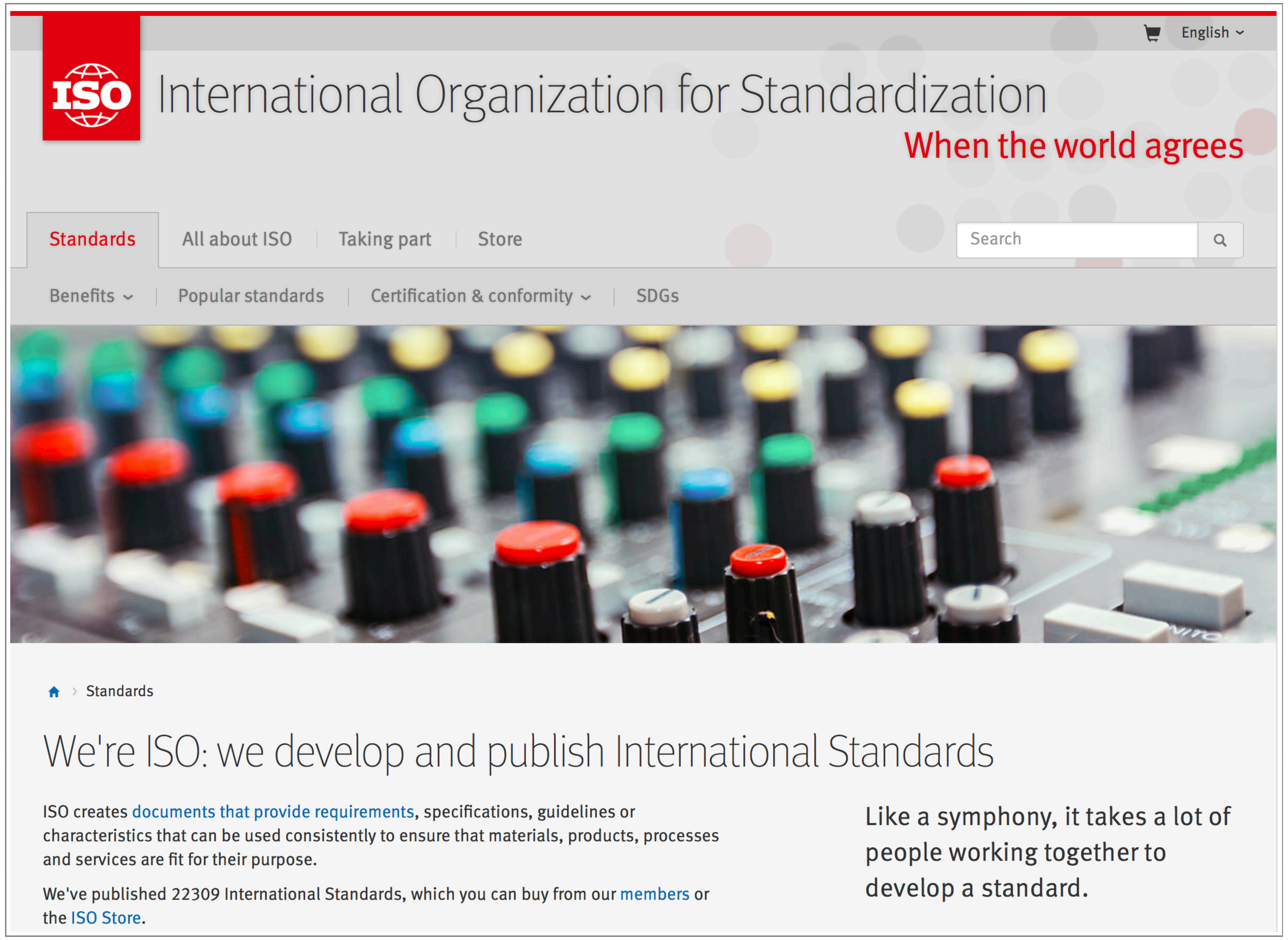


Patents


02
30
16
27
04
28
17
11
14
12
24
33
06
29
03
31
32
10
19
05
20
09
23
25
13
08
21
07
26
18
15
01
22


(No Model.)
3 Sheets—Sheet 1.
H. HOLLERITH.
ART OF COMPILING STATISTICS.
No. 395,782.
Patented Jan. 8, 1889.
Fig. 1.
Fig. 2.
Fig. 3
Fig. 4.
Fig. 5.
Fig. 6.
Witnesses.
Inventor
Herman Hollerith
by Church & Church
his Attorneys.

Technological Standards



The Importance of Peer Review

Peer reviews subjects an author's scholarly work, research or ideas to the scrutiny of others who are experts in the same field

It has many benefits but two primary purposes

1. A filter to ensure that only high quality research is published, especially in reputable journals, by determining validity, significance and originality

2. A mechanisms to improve the quality of manuscripts that are deemed suitable for publication

Peer review is at the heart of scientific research

Reputable Sources

Peer review is at the heart of scientific research

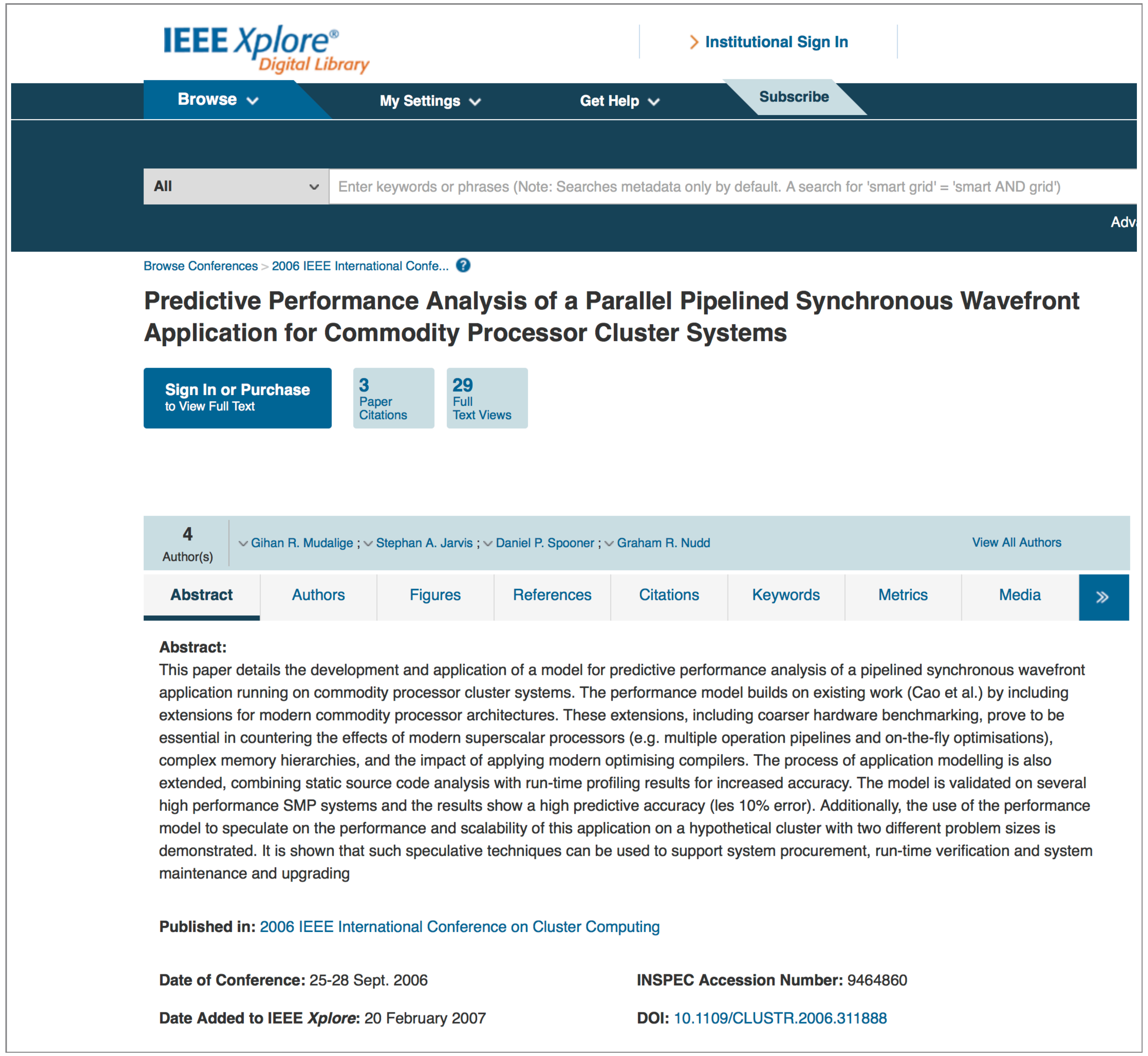
Publication in peer-reviewed conferences and journals ensures a certain level of confidence in research quality

Magazines, websites, blogs, news articles and similar sources can be useful in scientific research but their role is not to form the basis of a sound argument

It may alert the reader to the existence of reputable work, but is rarely worth citing

Learning might be built on a wider literature, but the arguments in the write-up should be based on knowledge from refereed sources

Reputable Sources?



IEEE Xplore® Digital Library

> Institutional Sign In

Browse | My Settings | Get Help | Subscribe

All | Enter keywords or phrases (Note: Searches metadata only by default. A search for 'smart grid' = 'smart AND grid')

Adv

Browse Conferences > 2006 IEEE International Confe...

Predictive Performance Analysis of a Parallel Pipelined Synchronous Wavefront Application for Commodity Processor Cluster Systems

Sign In or Purchase to View Full Text | 3 Paper Citations | 29 Full Text Views

4 Author(s) | Gihan R. Mudalige ; Stephan A. Jarvis ; Daniel P. Spooner ; Graham R. Nudd | View All Authors

Abstract | Authors | Figures | References | Citations | Keywords | Metrics | Media

Abstract:
This paper details the development and application of a model for predictive performance analysis of a pipelined synchronous wavefront application running on commodity processor cluster systems. The performance model builds on existing work (Cao et al.) by including extensions for modern commodity processor architectures. These extensions, including coarser hardware benchmarking, prove to be essential in countering the effects of modern superscalar processors (e.g. multiple operation pipelines and on-the-fly optimisations), complex memory hierarchies, and the impact of applying modern optimising compilers. The process of application modelling is also extended, combining static source code analysis with run-time profiling results for increased accuracy. The model is validated on several high performance SMP systems and the results show a high predictive accuracy (les 10% error). Additionally, the use of the performance model to speculate on the performance and scalability of this application on a hypothetical cluster with two different problem sizes is demonstrated. It is shown that such speculative techniques can be used to support system procurement, run-time verification and system maintenance and upgrading

Published in: 2006 IEEE International Conference on Cluster Computing

Date of Conference: 25-28 Sept. 2006

INSPEC Accession Number: 9464860

Date Added to IEEE *Xplore*: 20 February 2007

DOI: 10.1109/CLUSTR.2006.311888

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons

Languages
বাংলা
Deutsch
Ελληνικά
Français
Italiano
മലയാളം
ਪੰਜਾਬੀ
Scots
中文

Article | Talk | Read

Wiki Loves Monuments: Photograph a monument, help Wikiped

Supercomputer

From Wikipedia, the free encyclopedia
(Redirected from High performance computing)

"High-performance computing" redirects here. For narrower definitions of HPC, see High-throughput computing and Many-task computi

A **supercomputer** is a computer with a high level of performance compared to a general-purpose computer. Performance of a supercompu floating-point operations per second (FLOPS) instead of million instructions per second (MIPS). As of 2017, there are supercomputers whic nearly a hundred quadrillion FLOPS.[3] As of November 2017, all of the world's fastest 500 supercomputers run Linux-based operating syst research is being conducted in China, the United States, the European Union, Taiwan and Japan to build even faster, more powerful and m superior exascale supercomputers.[5]

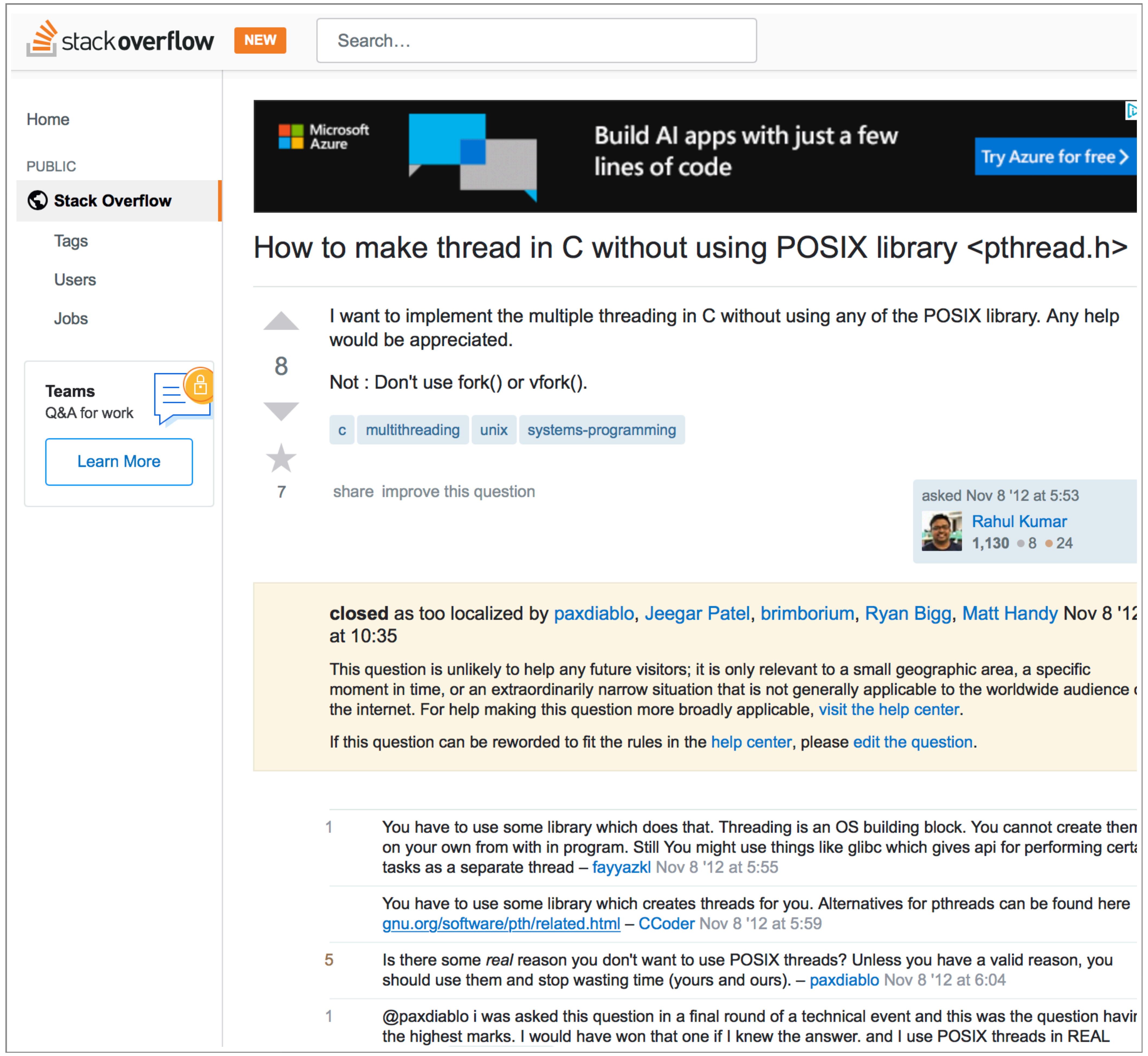
Supercomputers play an important role in the field of computational science, and are used for a wide range of computationally intensive tas including quantum mechanics, weather forecasting, climate research, oil and gas exploration, molecular modeling (computing the structures chemical compounds, biological macromolecules, polymers, and crystals), and physical simulations (such as simulations of the early mome airplane and spacecraft aerodynamics, the detonation of nuclear weapons, and nuclear fusion). Throughout their history, they have been es cryptanalysis.[6]

Supercomputers were introduced in the 1960s, and for several decades the fastest were made by Seymour Cray at Control Data Corporati Research and subsequent companies bearing his name or monogram. The first such machines were highly tuned conventional designs tha their more general-purpose contemporaries. Through the 1960s, they began to add increasing amounts of parallelism with one to four proc From the 1970s, the vector computing concept with specialized math units operating on large arrays of data came to dominate. A notable e successful Cray-1 of 1976. Vector computers remained the dominant design into the 1990s. From then until today, massively parallel super of thousands of off-the-shelf processors became the norm.[7][8]

The US has long been a leader in the supercomputer field, first through Cray's almost uninterrupted dominance of the field, and later throug field in the 1980s and 90s, but since then China has become increasingly active in the field. As of June 2018, the fastest supercomputer on a LINPACK benchmark score of 122.3 PFLOPS, exceeding the previous record holder, Sunway TaihuLight, by around 29 PFLOPS.[3][9] Sur Chinese computer to enter the TOP500 list without using hardware from the United States. As of June 2018, China had more computers (2 computers held eight of the top 20 positions;[10][11] the U.S. has six of the top 10 and China has two.

Contents [hide]

Reputable Sources?



Digital Libraries and Search Engines

Many sources for published research in Computer Science

- ACM Digital Library, Google Scholar, IEEE Xplore Digital Library etc..
- Important to carefully consider the origins of every resource used

Be An Effective Reader

Reading a paper is not similar to studying for an exam. There is rarely a need to understand every line.

Give each paper neither more nor less time than it deserves

Skim through a new paper to identify the extent to which it is relevant - **only read thoroughly if there is likely to be value in doing so**

If it is **important** for the research, it is necessary to properly understand the details, but always beware of details that may be wrong, or garbled.

Not Enough Information?

This is common and there is no single cause

- A narrow research topic
- A specific theme within a sufficiently broad research topic

Consider the scope and focus of the literature review

Spend time searching for related articles

- **You you cannot finish this up in one day**
- **Come back to searching frequently - ask yourself if you've found everything relevant**

Too Much Information?

It is easy to become overwhelmed by useful information

Consider the overarching aim of the research being written

Consider how related is the found research to your own

Consider the scope and focus of the literature review

Common Pitfalls - Example 1

Example: “We present the first results that show...”

Problem: You need to be sure that **nobody else has presented results** that show the same thing - and results are correct!

Solution: Apply scientific rigour and read extensively

Common Pitfalls - Example 2

Example: "Scientists at CERN demonstrated that it was possible for us to exceed the speed of light."

Problem: This might be a good newspaper headline but it is a clear **misrepresentation** of the research carried out

Solution: Discuss research accurately

Common Pitfalls - Example 3

Example: "We definitely answer a long-standing open question relating to the verification of computer programs"

Problem: Definitely? Really? No more research left to do for that question then? Maybe that's true..maybe!

Solution: Appraise honestly and write with precision

Common Pitfalls - Example 4

Example: "The main contributions of [X], a paper that is critical to the work presented in this paper were related to the development of a scheduling algorithm for commodity clusters. Scheduling algorithms, such as those proposed in [Y] and [Z] are important in many applications but [X] is the most important for the work presented"

Problem: Lots of words but very little being said

Solution: Focus on important points of relevant research

Common Pitfalls - Example 5

Example: “We use the same system as in [X]”

Problem: I haven’t read [X]

Solution: Don’t make assumptions about your reader

Common Pitfalls

Omitting seminal research

Misrepresenting the research of others

Under/over-stating own contributions

Focusing on irrelevant aspects of existing research

Expecting a reader to be familiar with a particular paper

Literature Review - Advice

A critical summary and synthesis of current knowledge relating to a topic of research

More than just a summary of existing research

Establishing the scope of a review can be difficult

Extensive reading and critical analysis are important

System Requirements

System Requirements

Functional requirements

- What the system should do
- Measurable and usually quantifiable

Non-functional requirements

- Qualities that a system should possess (e.g., usability, scalability, etc.)

Hardware and Software Constraints

- What resources do you have available and how does that impact the scope of the work?

System Requirements - Advice

Break down functional requirements into areas

You can use tables of functional and non-functional requirements to save on the word count (figures, tables, listings and references don’t count towards word count)

Most relevant if you’re writing a coursework report rather than developing an academic paper

Design

Design

Most relevant if you're writing a coursework report rather than developing an academic paper

- Academic paper would be more likely to have Introduction-> Related Work-> Models-> Theory Sections-> Results -> Discussion -> Conclusion

Design sections are normally divided into the same sections as the systems

- First subsection is usually a Systems Architecture, such that the reader can see the different components of the system
- Subsequent subsections are normally names for each component

Design - Advice

It’s a subjective point but, in my estimation, a good design section should be agnostic of technology

Expressing your designs in the abstract allows you to specialise them when it comes to evidencing and documenting your implementation

Establish your systems architecture first

Having a global view will help you to break down what you’ve done into subsections and provide insight as to whether you’ve been consistent with your system model

Implementation

Implementation

Broadly speaking you're trying to make your abstract design work more concrete and evidence that your design has been realised

Common to mirror the structure of the Design section in your Implementation section

> Having the same subsections doesn't means having the same content - the focus shift from the abstract composition of your system to the specifics of the technology and nuances of what you've produced
>
> Identify the key features of your system that you'd like to emphasise ,e.g., don't spend 1000 words on how you validated CSS when you've got 20 words on the purpose of your system

Implementation - Advice

Characterise the role and necessity of each component

If everything you've included is necessary and has a purpose, it's hard to argue

Use code snippets to highlight key features of what you've done

Copying in lots of code might look nice but it doesn't help anyone, especially if your source code is also available

Focuses your mind on the key feature of your implementation

Testing

Testing

Evidencing and documenting your testing process really depends on your approach to software development

Traditional development models can lend themselves to producing testing tables

Agile methodologies generally have some form of continuous testing, which you might like to explain and present sample tests - you can still do testing tables to show samples

Remember that both the process and the tests themselves are of interests

Testing - Advice

It’s hard for your reader to verify your tests so they are more or less compelled to believe you unless you’re contradicting yourself

Classically you might like to focus on unit, integration and systems testing as your three subsections, particularly in the context of a coursework report

Establish the structure and what you want say first, before populating the structure with content

Results

Results

Presenting results well is hard and take effort, which means you should start early

It takes a long time to condense findings into an intelligible form that communicates the quality of what you've produced

Be as quantitative and objective as you can be when presenting results

Graphs, tables and figures are your friend - use them as much as you can to communicate your findings

Try to write with precision, even if the results don't quit make the case you want to them to make

There is a wider point here about being precise with the claims you make

Results - Advice

It’s ideal if you can summarise what you’ve found or produced in a sentence or two

That statement can then be used across your abstract, introduction, conclusion and elsewhere in your work to capture exactly what your were driving towards and have achieved

Figures, tables and listings don’t count towards your word count but they still need some prose in the body of your report to explain them

Tell your reader what a figure /tables / listing shows and what they should take from it

Evaluation

Evaluation

Not a feature of most academic papers within computer science, since most contributions would be evaluated through the consideration of results

Projects with a software development focus or a blend of development and research can focus on the functional and non-functional requirements laid out in the System Requirements section

Dissertation projects might like to consider evaluation subsection for (i) functions requirements, (ii) non-functional requirements, (iii) legal, social, ethical and professional issues, and (iv) project management

Evaluation - Advice

Projects with a software development focus or a blend of development and research might be able to reuse element of the System Requirements section

Indicating the extent to which each functional and non-functional requirement has been met

Honest evaluations will gain credit, even where a project hasn't necessarily lived up to expectations or has experienced challenges

Conclusion

Conclusion

Usually a summary of what has been achieved and some coverage of future work

Purpose of summary is to emphasise the problem, the methods of addressing the problem and the results / contributions of the work

Conclusion - Advice

Most conclusion sections can easily be constructed once the other sections have been written, with the possible exception of future work

- Those three key sentences that capture the problem, method and results are particularly value in the summary

Typically to include two or three points of key future work

- Avoid “The project should have…” in favour of “Leading on from the project…”
- Perhaps include one little idea, one substantial idea and one long-term / forward looking idea