Ben Simner

ben.simner@cl.cam.ac.uk
https://2plus2a.com/

Positions & Experience

2025– **Research Associate**, Computer Science, University of Cambridge.
2023-2025 **Research Assistant**, Computer Science, University of Cambridge.
2018-2025 **Ph.D.**, Computer Science, University of Cambridge.
Autumn'18 **Intern**, University of Cambridge.
Summer'17 **Intern**, Microsoft Research, Cambridge.
Summer'16 **Intern**, University of York.

Qualifications

2018-2025 **Ph.D**., Computer Science, University of Cambridge
Thesis: Arm system semantics
(*no corrections*)

2014-2018 **MEng**., Computer Science, University of York
First-class honours (with distinction)
Dissertation: Automated comparison of implementations of purely functional data structures

Publications

Preamble In these fields, the primary venue for publication is peer-reviewed conference proceedings. Publications order authors by contribution, often (but not always) with the PI being last. Those listed are all full-length papers, and not extended abstracts or pre-publications. The Symposium on Operating System Principles (SOSP), International Symposium on Computer Architecture (ISCA), Computer Aided Verification (CAV), and the European Symposium on Programming (ESOP) are flagship venues in their fields, which the Australian CORE rankings rate as A or A*, signifying the top 20% of leading Computer Science venues. These venues are all competitively peer-reviewed, with acceptance rates typically around 15-25%, e.g. SOSP'25 accepted 66 of 368 submissions (18%), ISCA'25 accepted 132 of 570 submissions (23%).

Synopsis My Ph.D. investigates the behaviour of modern multiprocessors at the interface with software, and this work forms the base for three of the listed papers: ESOP'20, ESOP'22 (nominated for best paper), and ISCA'25 (won best paper). During the course of my Ph.D. I have been part of a wider group with shared software infrastructure, some of which I have played a role in developing or maintaining, and this comprises two of the papers: CAV'21 and FMSD'23. Previous to my Ph.D. I was part of a project to improve confidence in software by constructing concurrent program logics, which I continue to work on; this comprises 2 papers: CAV'17 and ISMM'23. Finally, a recently started line of research investigates improved software testing for low-level systems, and this accounts for the final paper, SOSP'25.

- ▷ **Ghost in the Android Shell: Pragmatic Test-oracle Specification of a Production Hypervisor** [pdf]
  Kayvan Memarian[1], **Ben Simner**[1], David Kaloper Meršinjak, Thibaut Pérami, Peter Sewell
  in Proceedings of ACM SIGOPS 31st Symposium on Operating Systems Principles (SOSP'25, *to appear*), 15pp
  [1]These authors contributed equally.
- ▷ **Arm system semantics** [pdf]
  Ben Simner
  Ph.D. thesis, 2025
- ▷ **Precise exceptions in relaxed architectures** [pdf]
  Ben Simner, Alasdair Armstrong, Thomas Bauereiss, Brian Campbell, Ohad Kammar, Jean-Pichon Pharabod, and Peter Sewell
  in Proceedings of the 52nd International Symposium on Computer Architecture (ISCA'25), pp211-224 (14pp) BEST PAPER AWARD

- ▷ **Wait-Free Weak Reference Counting** [pdf]
  Matthew J. Parkinson, Sylvan Clebsch, and **Ben Simner**
  in Proceedings of the 2023 ACM SIGPLAN International Symposium on Memory Management, pp85-96 (12pp)
- ▷ **Isla: Integrating full-scale ISA semantics and axiomatic concurrency models (extended version)** [pdf]
  Alasdair Armstrong, Brian Campbell, **Ben Simner**, Christopher Pulte, and Peter Sewell
  in Formal Methods in System Design, Volume 63, May 2023, pp110-133 (24pp)
- ▷ **Relaxed Virtual Memory in Armv8-A** [pdf]
  Ben Simner, Alasdair Armstrong, Jean Pichon-Pharabod, Christopher Pulte, Richard Grisenthwaite, and Peter Sewell
  in Programming Languages and Systems: 31st European Symposium on Programming (ESOP'22), pp143-173 (31pp) NOMINATED FOR BEST PAPER
- ▷ **Isla: Integrating full-scale ISA semantics and axiomatic concurrency models** [pdf]
  Alasdair Armstrong, Brian Campbell, **Ben Simner**, Christopher Pulte, and Peter Sewell
  in Proceedings Part I of Computer Aided Verification: 33rd International Conference (CAV'21), pp303-316 (14pp)
- ▷ **ARMv8-A system semantics: instruction fetch in relaxed architectures** [pdf]
  Ben Simner, Shaked Flur, Christopher Pulte, Alasdair Armstrong, Jean Pichon-Pharabod, Luc Maranget, Peter Sewell
  in Programming Languages and Systems: 29th European Symposium on Programming (ESOP'20), pp626-655 (30pp)
- ▷ **Starling: Lightweight Concurrency Verification With Views** [pdf]
  Matthew Windsor, Mike Dodds, **Ben Simner**, Matthew J Parkinson
  in Proceedings Part I of Computer Aided Verification: 29th International Conference (CAV'17), pp544-569 (26pp)

Note: Awards are in GREEN. Nominations are in BLUE.

Summary of individual contribution For the ESOP'20, ESOP'22 and ISCA'25 papers, I was the primary technical lead: engaged directly in the discussions with Arm architects and others on the contributions, in the construction of the theoretical models, in the validation of the models, and the writing of the paper. Other co-authors engaged in supplementary modelling and discussions with architects (Flur, Pulte, Kammar, Pichon-Phararbod), in the maintenance of the tooling and hardware testing (Armstrong, Maranget), or construction and maintenance of related artifacts (namely Bauereiss and Campbell, working on the Arm ISA), or are industry contacts (Richard Grisenthwaite is the Arm chief architect, and closely collaborated on the ESOP'22 paper). For the CAV'21 paper I produced the model used to exercise the tool, and jointly maintained some parts of the tooling (the parts relevant to axiomatic modelling).

For the ISMM'23 paper I contributed a specific artifact: a mathematical proof of one of the concurrent algorithms presented in the paper. For the CAV'17 paper I was responsible for one of the primary technical contributions of the paper: the implementation of iterated views in the tool, and the proof it enabled.

For the SOSP'25 paper, I am a joint lead with Memarian; responsible for much of the machinery which forms a primary contribution of the the paper, as well as the writing of the paper.

Research statement

Modern computer systems are core to our information infrastructure, and we are increasingly reliant on them in our day-to-day lives. Concerningly, these systems do not always work in the way we need them to: software errors cause disruption of public services; malicious attacks gain access to confidential data; and buggy software interfere with peoples' lives.

Engineers follow industry best practice, but often this is insufficient to prevent such failures. Those practices usually involve imperfect test-and-debug processes, which further rely on inadequate documentation to describe the components' intended operation. This is particularly worrying for the core infrastructure: the systems which coordinate software and enforce security of confidential data; the tools those systems are built with; and, ultimately, the hardware they run on. Ensuring they perform correctly in *all* scenarios requires mathematical proof, which is out of reach for the majority of modern software. Disturbingly, even the hardware-software interface has important questions left open, leaving no solid foundation for that proof.

Crystallized down, the challenge I address is then **how to build confidence in our software infrastructure, when the foundations are shaky and the existing tools unwieldy?**

Attacking this needs new theory, and the application of that theory in new tools, all in the context of industry-owned systems. My research does all this along three threads of work:

(1) **Building robust foundations for the hardware-software interface**: working with Arm Limited, whose processor designs are in most modern devices, I build mathematical models capturing the allowed behaviour of their processors, with particular focus on those parts relied on by critical systems software. This forms the majority of the work in my Ph.D.

(2) **Exploring more approachable techniques for formal proof**: constructing tooling to enable simpler proof of algorithms, and using the tools on examples deployed in real software.

(3) **Constructing foundational safety nets for programmers**: developing lightweight approaches, to give software developers the confidence they need to develop software. Including, in collaboration with Google, building lightweight verification tools which found security vulnerabilities and bugs in Android.

My work **spans multiple fields**: by testing real software (*systems*); in clarifying the interfaces our software rely on (*computer architecture*); and building the theory and tooling which underpins it all (*programming languages and verification*). I tackle **pure research problems**, with the construction of new mathematical models and techniques, but **in a way which benefits the computing industry as a whole**, working on artifacts people really care about: Android, used by billions; and the Arm architecture, powering over a hundred billion devices; giving an exciting opportunity to directly impact practice. My research is **highly collaborative**. I had the unique opportunity to engage in deep technical discussions, and build a productive relationship with, Arm and their Chief Architect. As a member of RISC-V, I have been part of a cross-industry effort into open architecture, advising on research-relevant parts. In contracting with Google, we explored how my research scales to industry software, and formed close connections to the Android kernel team. Academic co-authors are many and distributed worldwide, in: Edinburgh, Aarhus (Denmark), SNU (South Korea), INRIA (France), and TAU (Israel).

I now give a short overview of each thread of work so far, plus a plan for future work.

Thread I: Clarifying the Arm architecture (the dissertation)

Challenge: **how to give a solid and trustworthy specification of the hardware-software interface which enables mathematical reasoning about systems software?**

This requires taking architecture away from prose documents, as they had historically been, and towards precise mathematics. For 15 years there has been a significant push in this direction. However, those parts required for our systems' security have remained mostly obscure. My dissertation develops mathematical models for those aspects, in a way which allows one to probe the specification, either manually or for automated testing of small programs, and validates those models against currently available hardware and the architectural intent. In doing so it *clarifies* the architecture, increasing confidence that future hardware and software will conform.

Thread II: Reasoning about concurrent algorithms

Challenge: **how can proof of concurrent algorithms be made more accessible?**

Multiple components accessing the same memory at the same time is a pattern often found in critical components of software. The correctness of these components often rely on implicit, intangible, concepts, which has historically made proof challenging to achieve. *Starling*, designed by Matt Windsor during the course of his Ph.D. (2019), automatically checks simple facts, greatly reducing the burden on the engineer; I joined that project in 2016, and became a co-author, by extending it with support for richer facts, and we used that to prove a Rust-style atomic reference counter. Starling has since been used to prove some small concurrent algorithms, including a proof I did for an algorithm used in Microsoft's experimental Verona language. This work is entirely independent of the Ph.D. and the work at Cambridge.

Thread III: Building confidence in systems software

Challenge: **how can we help systems programmers gain confidence in their code without requiring full mathematical proof?**

Without proof, programmers rely on testing. This thread of research develops lightweight strategies, somewhere in-between testing and proof, and demonstrate they work at scale by employing them on *pKVM*: a security component of Android, deployed since Android 13. My research tackles two distinct challenges here: (1) **how to check the hardware-software interface**, especially in the context of concurrent systems features (c.f. Thread I); and (2) **harmonising full formal specifications with testing**, by writing specifications just as one would when writing proofs, but integrating them into the testing process, giving more confidence than just testing alone but without the investment of full proof.

Plan of future work

The success of the threads of work enable a rich set of possibilities of new work. For building new theory, investigating new techniques, and building deeper connections. My research plan over the next 5 years is split into three major tasks.

Unifying the hardware interface (1-2 py) We now have, in no small part thanks to my work here, a number of models of various features of the Arm architecture. However, real software typically uses a mix of features. Thus, **one needs a single unified model** of them all. My plan is to construct such a model, for Arm, **enabling full mathematical proof of real systems software**, a goal not achievable today.

Reasoning about systems software (2-3 py) Building on the lessons learned from Threads II and III, my research will **produce tools which can be used to reason abstractly about systems software**. In particular, to produce simplified architectural models, to make reasoning about low-level software easier; and to take the work from Thread III(1), and lift its lightweight check into something more foundational, which can be used to verify real software.

Scaling to industry speed (1+ py) Thread III built lightweight verification which scales to industry-sized software. However, the process was slow, often only finding bugs months after the code was written – or worse, after it was deployed to users. My future research will **scale those techniques up to industry *speed***: to **find bugs as they are written**, not months after.

A long-term research agenda Each thread contributes a deeper understanding of the way software and hardware interact. This moves us one step closer to being able to build confidence in our infrastructure, but, of course, with much still left to do. An independent position would give me the opportunity to push those boundaries, beyond merely clarifying architectural interfaces, as has historically been the subject of my supervisor's research, and progress towards *reasoning above* those interfaces: continuing the lines of research in Threads 2 and 3, and the planned future tasks. Over a longer timescale, this research will seek to make reasoning at that hardware-software interface possible and feasible: with more robust foundations, less unwieldy tools for proof, and moving towards making those tools scale to industry size and speed.