ZILIN CHEN PhD.

Email: zilin.chen@unswalumni.com

Homepage: https://zilinc.github.io/

WORK EXPERIENCE

Nov 2022 – Mar 2024

Software Engineer

Ghost Autonomy
Sydney, Australia

Ghost was a start-up company headquartered in Mountain View, CA, USA, whose mission was to bring technical innovations to software-defined consumer autonomy systems. I was hired to join the team responsible for developing and maintaining a custom deep learning model language written in Scala. Later, as the technical stack shifted and the custom language replaced by PyTorch, I began to work in a model engineering role, experimenting with deep learning models for vision-based perception systems for autonomous driving. During my employment at Ghost, my technical contributions include:

- *Implemented a semantic segmentation model in our proprietary model language as described in an academic paper.*
- *Implemented a new C/OpenCL backend in our model language compiler (in Scala), which previous had had only Scala backends.*
- *Refactored the implementation of a core clustering algorithm, and conducted property-based testing on it (in Scala).*
- *Surveyed relevant academic papers and experimented with several algorithms and deep learning models (in Python/PyTorch) for 3D lane detection and ego car pitch angle estimation under complicated road conditions.*

Feb 2014 – Jun 2021

Research Engineer (since Jan 2016)
Research Assistant (until Jan 2016)

(Part-time when I was enrolled in my PhD program)

Trustworthy Systems, CSIRO's Data61 (Formerly NICTA)
Sydney, Australia

I mainly worked on the Cogent *project (https://trustworthy.systems/projects/TS/cogent.pml). I was responsible for implementing the* Cogent *compiler (in Haskell), maintaining the codebase, the CI infrastructure, the repository, connecting with the community, developing documentations, writing academic papers, and doing research team administration work. I also contributed to the design of the language, the verification framework, and proof engineering. I played a central role in the team as the agent between systems programmers and verification engineers: assisting systems programmers in learning and using the* Cogent *framework, and collecting feedback from them to improve* Cogent*.*

On the side, I also co-supervised students (≈ 10), ranging from summer interns, honours thesis students and master students.

Feb 2014 – Jun 2014

Tutor

School of Computer Science and Engineering, UNSW
Sydney, Australia

I ran tutorials and laboratory sessions for a first-year compulsory course for computer science and software engineering students (COMP1927: Computing 2. https://legacy.handbook.unsw.edu.au/undergraduate/courses/2014/COMP1927.html).

Nov 2013 – Feb 2014
Nov 2012 – Feb 2013

Summer Scholar

Software Systems Research Group, NICTA
Sydney, Australia

I worked on the design, implementation, and verification of a domain-specific language for data (de-)serialisation. The language was used for developing formally verified file systems.

Dec 2011 – Jan 2012

Software Engineer Intern

Analog Devices, Inc.
Beijing, China

I worked on the memory management and optimisations of an assembler (written in C) for a pixel preprocessing device.

EDUCATION

Aug 2015 – Mar 2023

Doctor of Philosophy, Computer Science

Faculty of Engineering, UNSW Sydney, Australia

On program leave: Sep 2019 – Sep 2021

PhD Thesis: Towards A Practical High-Assurance Systems Programming Language
Supervised by Prof. Gabriele Keller, Dr. Christine Rizkallah and Prof. Gernot Heiser
Examined by Prof. Anil Madhavapeddy and Dr. Jonathan Protzenko

Abstract:
Writing correct and performant low-level systems code is a notoriously demanding job, even for experienced developers. To make the matter worse, formally reasoning about their correctness properties introduces yet another level of complexity to the task. It requires considerable expertise in both systems programming and formal verification. The development can be extremely costly due to the sheer complexity of the systems and the nuances in them, if not assisted with appropriate tools that provide abstraction and automation.
Cogent *is designed to alleviate the burden on developers when writing and verifying systems code. It is a high-level functional language with a certifying compiler, which automatically proves the correctness of the compiled code and also provides a purely functional abstraction of the low-level program to the developer. Equational reasoning techniques can then be used to prove functional correctness properties of the program on top of this abstract semantics, which is notably less laborious than directly verifying the C code.*
To make Cogent *a more approachable and effective tool for developing real-world systems, we further strengthen the framework by extending the core language and its ecosystem. Specifically, we enrich the language to allow users to control the memory representation of algebraic data types, while retaining the automatic proof with a data layout refinement calculus. We repurpose existing tools in a novel way and develop an intuitive foreign function interface, which provides users a seamless experience when using* Cogent *in conjunction with native C. We augment the* Cogent *ecosystem with a property-based testing framework, which helps developers better understand the impact formal verification has on their programs and enables a progressive approach to producing high-assurance systems. Finally we explore refinement type systems, which we plan to incorporate into* Cogent *for more expressiveness and better integration of systems programmers with the verification process.*

Scholarships and awards:
- *2015 University International Postgraduate Award*
- *2016 University International Postgraduate Award*
- *2017 University International Postgraduate Award*
- *2018 University International Postgraduate Award*
- *2019 University International Postgraduate Award*

Feb 2011 – Nov 2013 **Bachelor of Science (Honours Class 1), Computer Science**
Faculty of Engineering, UNSW Sydney, Australia

Honours thesis: Towards Verified Serialisation and Deserialisation in File Systems
Supervisor: A.Prof. Gabriele Keller Assessor: Prof. Gerwin Klein

Scholarships and awards:
- *2012 Taste of Research Summer Scholarships*
- *2013 Taste of Research Summer Scholarships*

Prizes:
- *2012 The CSE Undergraduate Performance Prize Year 3 (8th Place) for academic excellence*
- *2013 The CSE Undergraduate Performance Prize Year 4 (5th Place) for academic excellence*
- *2013 The Jane Street COMP3141 Software Design and Implementation Prize, 2nd Place*

Sep 2008 – Jan 2011 **Undergraduate program, Information Security**
School of Information Science and Technology, Sun Yat-Sen University, China

Degree discontinued and credits partly transferred to UNSW

Prize:
- *The University Scholarship for Outstanding Students, 2nd prize (2008 – 2009)*

RESEARCH INTERESTS

My main research interests are in functional programming, embedded domain-specific languages, type theory, formal semantics, compilation, program synthesis, and software verification (in particular interactive theorem proving). I am also interested in logic, metamathematics, proof theory and other forms of formal methods. My research has a strong focus on applying formal methods to low-level systems programming, with special attention to user experience and industrial adoption.

TECHNICAL SKILLS

My main programming language is Haskell, which I use for almost everything due to its purely functional nature and the powerful static type system. In the past, I also used C, Scala and Python for work at various stages, and have had exposure to a lot of real-world languages and research languages (including domain-specific languages) for certain tasks. Agda is the proof assistant that I primarily use for work. I also have some experience with Isabelle/HOL, Coq and Idris. I am confident in learning new programming languages, not only to use the language to accomplish programming tasks, but also to understand the nuances in the design of the language as a researcher in programming languages.

As a software engineer:

- Linux/Unix is the family of operating systems on which all my work was done.
- Version control (Git, Mercurial, Darcs) is part of my daily routine.
- I prepare technical documentations using LaTeX, reStructuredText and various markdown languages.
- `Vim` is gold, but `Emacs` is occasionally very useful.
- Experience with various CI/CD platforms, including Travis CI, Jenkins, Bamboo, Github Actions.
- Working knowledge of Agile project management principles and practice.

PUBLICATIONS

- Zilin Chen. 2023. *Towards A Practical High-Assurance Systems Programming Language*. PhD thesis. CSE, UNSW Sydney, UNSW Sydney, Australia, (Mar. 2023). doi: https://doi.org/10.26190/unsworks/24733
- Zilin Chen, Ambroise Lafont, Liam O'Connor, Gabriele Keller, Craig McLaughlin, Vincent Jackson and Christine Rizkallah. 2023. Dargent: a silver bullet for verified data layout refinement. *Proc. ACM Program. Lang.*, 7, Article 47, (Jan. 2023), 27 pages. doi: 10.1145/3571240
- Zilin Chen, Christine Rizkallah, Liam O'Connor, Partha Susarla, Gerwin Klein, Gernot Heiser and Gabriele Keller. 2022. Property-based testing: climbing the stairway to verification. In *ACM SIGPLAN International Conference on Software Language Engineering* (SLE 2022). ACM, Auckland, New Zealand, (Dec. 2022), 14 pages. doi: 10.1145/3567512.3567520. 🏆 ACM SIGPLAN Distinguished Artifact Award.
- Zilin Chen. 2022. A Hoare logic style refinement types formalisation. In *Proceedings of the 7th ACM SIGPLAN International Workshop on Type-Driven Development* (TyDe '22). ACM, Ljubljana, Slovenia, (Sept. 2022), 14 pages. doi: 10.1145/3546196.3550162
- Liam O'Connor, Zilin Chen, Christine Rizkallah, Vincent Jackson, Sidney Amani, Gerwin Klein, Toby Murray, Thomas Sewell and Gabriele Keller. 2021. Cogent: uniqueness types and certifying compilation. *Journal of Functional Programming*, 31. doi: 10.1017/S095679682100023X
- Zilin Chen, Matt Di Meglio, Liam O'Connor, Partha Susarla Ajay, Christine Rizkallah and Gabriele Keller. 2019. A data layout description language for Cogent. at PriSC. Lisbon, Portugal, (Jan. 2019)
- Liam O'Connor, Zilin Chen, Partha Susarla Ajay, Christine Rizkallah, Gerwin Klein and Gabriele Keller. 2018. Bringing effortless refinement of data layouts to Cogent. In *International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*. Springer, Limassol, Cyprus, (Nov. 2018), 134–149. doi: https://doi.org/10.1007/978-3-030-03418-4_9
- Zilin Chen, Liam O'Connor, Gabriele Keller, Gerwin Klein and Gernot Heiser. 2017. The Cogent case for property-based testing. In *Workshop on Programming Languages and Operating Systems (PLOS)*. ACM, Shanghai, China, (28th Oct. 2017), 1–7. isbn: 9781450351539. doi: https://doi.org/10.1145/3144555.3144556
- Zilin Chen. 2017. Cogent$^{\Uparrow}$: giving systems engineers a stepping stone (extended abstract). In *The workshop on Type-Driven Development* (TyDe'17). Oxford, UK. Retrieved April 2019 from https://www.cse.unsw.edu.au/~zilinc/tyde17.pdf
- Sidney Amani, Alex Hixon, Zilin Chen, Christine Rizkallah, Peter Chubb, Liam O'Connor, Joel Beeren, Yutaka Nagashima, Japheth Lim, Thomas Sewell, Joseph Tuong, Gabriele Keller, Toby Murray, Gerwin Klein and Gernot Heiser. 2016. Cogent: verifying high-assurance file system implementations. In *International Conference on Architectural Support for Programming Languages and Operating Systems*. Atlanta, GA, USA, (Apr. 2016), 175–188. doi: 10.1145/2872362.2872404
- Liam O'Connor, Zilin Chen, Christine Rizkallah, Sidney Amani, Japheth Lim, Toby Murray, Yutaka Nagashima, Thomas Sewell and Gerwin Klein. 2016. Refinement through restraint: bringing down the cost of verification. In *International Conference on Functional Programming*. Nara, Japan, (Sept. 2016)
- Christine Rizkallah, Japheth Lim, Yutaka Nagashima, Thomas Sewell, Zilin Chen, Liam O'Connor, Toby Murray, Gabriele Keller and Gerwin Klein. 2016. A framework for the automatic formal verification of refinement from Cogent to C. in *International Conference on Interactive Theorem Proving*. Nancy, France, (Aug. 2016)
- Liam O'Connor, Gabriele Keller, Sidney Amani, Toby Murray, Gerwin Klein, Zilin Chen and Christine Rizkallah. 2014. CDSL Version 1: Simplifying Verification with Linear Types. Technical Report. NICTA, Sydney, Australia, (Oct. 2014)
- Gabriele Keller, Toby Murray, Sidney Amani, Liam O'Connor, Zilin Chen, Leonid Ryzhyk, Gerwin Klein and Gernot Heiser. 2013. File systems deserve verification too! In *Workshop on Program-*

ming Languages and Operating Systems (PLOS). Farmington, Pennsylvania, USA, (Nov. 2013), 1–7. doi: 10.1145/2525528.2525530