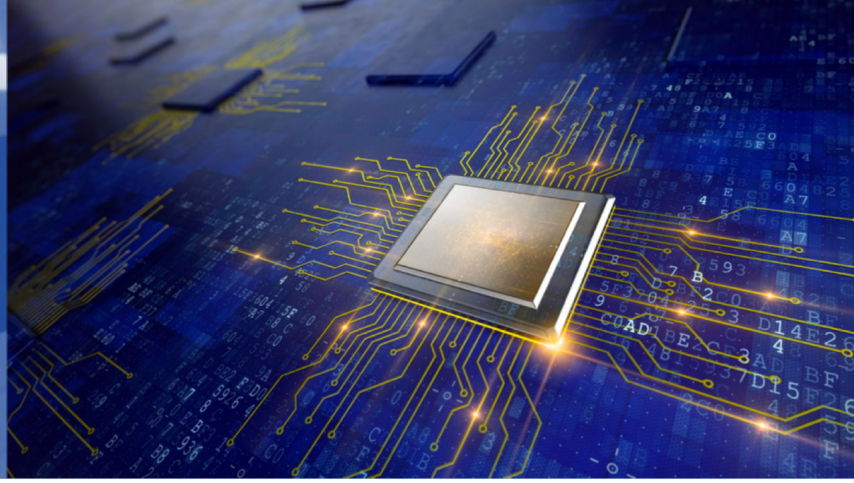
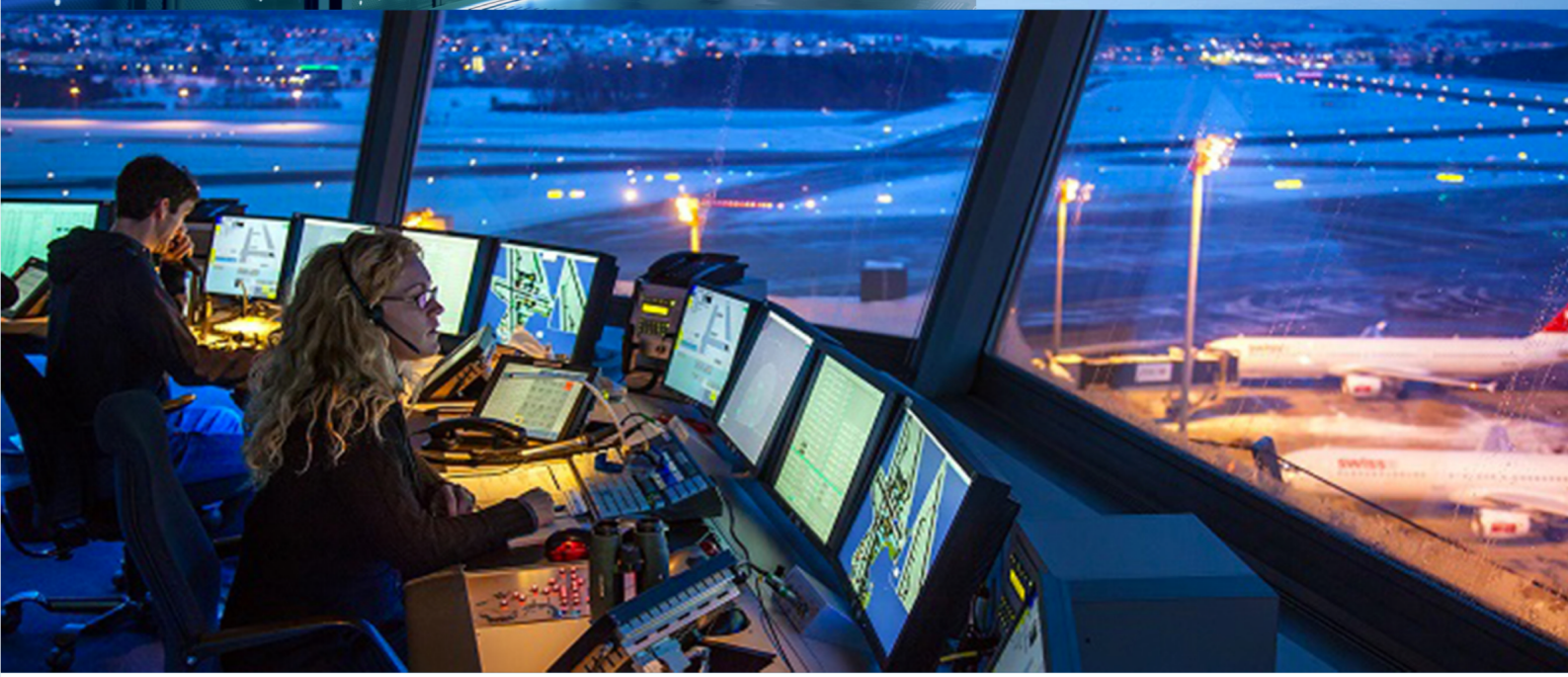


RESEARCH INSTITUTE IN VERIFIED TRUSTWORTHY SOFTWARE SYSTEMS

UK's second research institute in cyber-security

Annual Report 2018/2019



EPSRC
Engineering and Physical Sciences Research Council

National Cyber Security Centre
a part of GCHQ

Imperial College London

FOREWORD

Philippa Gardner, Director of VeTSS

The Research Institute in Verified Trustworthy Software Systems (VeTSS) is the UK's second Academic Research Institute in cyber security, funded by the Engineering and Physical Sciences Research Council (EPSRC) for five years, from April 2017. The purpose of VeTSS is to bring together and support world-class UK academics, industrialists and government employees, unified by a common interest in software analysis, testing and verification. VeTSS stands at the forefront of research developments in fundamental theories and industrial-strength tools, targeting real-world applications. It succeeds the previous three-year Research Institute in Automated Program Analysis and Verification, funded by EPSRC and GCHQ.

The National Cyber Security Centre (NCSC) anticipates giving approximately £2.5 million to VeTSS over five years to support academic research projects in software analysis, testing and verification. This annual report provides a description of the projects funded for the first two years, from April 2017 to March 2019. It demonstrates the deep connection between the VeTSS academic research, the standard bodies and industry. For example, Batty's two VeTSS projects on understanding the concurrent behaviour of C++ have directly influenced the ISO C++ standard, and Livshits's and Donaldson's VeTSS project relates to an academic start-up of Donaldson that was bought by Google in 2018. This report also describes how VeTSS funding has led directly to a total of £10.54M of further funding from EPSRC, the EU, government bodies and industry. For example, Yoshida's work on her VeTSS project played a crucial part in her obtaining a £1.46M UKRI Established Career Fellowship, 2020-2025. Furthermore, the interaction between NCSC and VeTSS has led to an additional invited call by NCSC on "Verified High Assurance Software" in 2019.

We have held a number of events since the start of VeTSS, including our main annual workshop "Formal Methods and Tools for Security" at Microsoft Cambridge in September 2017 and 2018. We have been part of a successful bid for the Cambridge Isaac Newton Institute programme on "Verified Software" in 2020, organised by de Moura (Microsoft Redmond), Farzan (Toronto), Hoare (Microsoft), Gardner (Imperial), Larsen (Aalborg), Leroy (Inria Paris), McMillan (Microsoft Redmond), O'Hearn (Facebook and UCL), Sewell (Cambridge), Shankar (SRI, California, lead organiser) and Vardi (Rice). This meeting will bring international academics and industrialists to the UK for six weeks, laying the groundwork for the next generation of verification grand challenges. We have, therefore, combined our annual workshop with a preparatory workshop on "Verified Software" at the Newton Institute in Cambridge in September, 2019.

I hope that you will find this annual report of interest.

Professor Philippa Gardner
Director of VeTSS

VeTSS 2017/2018

Mechanising the Metatheory of SQL with Nulls

James Cheney
University of Edinburgh

Automated Testing for Web Browsers

Benjamin Livshits
Imperial College London

PrideMM Web Interface

Mark Batty
University of Kent

Verifying Efficient Libraries in CakeML

Scott Owens
University of Kent

SuperVectorizer

Greta Yorsh
Queen Mary Univ. of London

EASTEND: Efficient Automatic Security Testing for Dynamic Languages

Johannes Kinder
Royal Holloway Univ. of London

Automated Reasoning with Fine-Grained Concurrent Collections

Ilya Sergey
University College London

Mechanised Assume-Guarantee Reasoning for Control Law Diagrams via Circus

Jim Woodcock
University of York

MECHANISING THE METATHEORY OF SQL WITH NULLS

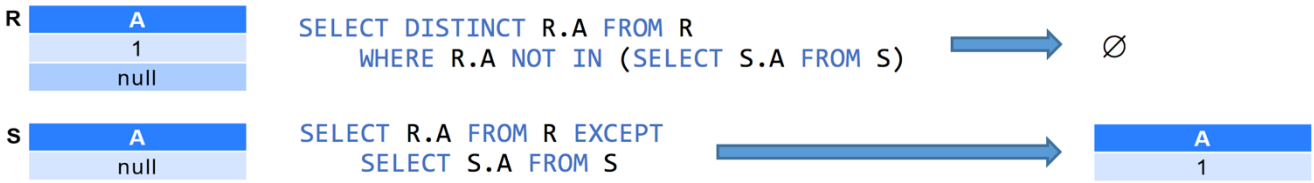

JAMES CHENEY WILMER RICCIOTTI

- SQL is the standard query language used by the multi-billion-dollar relational database industry
- SQL semantics is notoriously subtle: it is written in natural language and is inconsistent across implementations
- Previous attempts to verify SQL transformations have ignored widely-used features, such as null values
- We present the first mechanised semantics that models these features, making it possible to formally verify that real query optimisers are correct for real-world databases.

The Structured Query Language, SQL, is by far the most common language used by relational databases, which are the basis of a multi-billion-dollar industry. The SQL standard is described by a large and comprehensive definition (ISO/IEC 9075:2016), based on natural language rather than a formal specification; due to the lack of an agreed-on formal semantics, commercial SQL implementations interpret the standard in different ways, so that, given the same input data, the same query can yield different results depending on the SQL system it is run on.

SQL systems first run a *query optimiser* which applies a set of rewrite rules to obtain an equivalent query that can be processed more efficiently. However, due to the lack of a well-understood formal semantics, it is very difficult to validate the soundness of such rewrite rules, and incorrect implementations are known in the literature. Bugs in query optimisers could lead to corruption or errors in critical data.

Among SQL's features, its ability to deal with incomplete information, in the form of *null values*, accounts for a great deal of semantic complexity. To express uncertainty, logical predicates on tuples containing null values employ three truth values: *true*, *false*, and *unknown*. As a consequence, queries equivalent in the absence of null values can produce different results when applied to tables with incomplete data, as illustrated in the diagram below.



Although there are some previous formalisations of SQL or relational query languages, all of them ignore null values, so they "prove" query equivalences that are unsound in the presence of these features. Our project builds on a recent (on-paper) formal semantics for SQL with nulls by Guagliardo and Libkin, providing the validation of key meta-theoretic properties in the Coq proof assistant. We view this as a first step towards a future in which query optimisers are *certified*. Our development can be publicly accessed at its GitHub repository (https://github.com/wricciot/nullSQL).

PUBLICATIONS. An article is under submission to a leading conference on verification.

RELATED GRANTS. Dr James Cheney, ERC Consolidator Grant: "Skye: Bridging theory and practice for scientific data curation", 2016-2021, £1.75M.

IMPACT STATEMENT. "Database queries and query languages are widely used in industry, yet their implementations and optimisation rules are error-prone due to complications, such as the semantics of nulls. This can easily lead to subtle bugs in relational database engines or incorrect queries, and work on formalising the semantics of existing query languages, including the real-world semantics of nulls, is very important and likely to have a tangible impact on making systems more reliable. For example, optimisation rules proposed in Kim's seminal work on query un-nesting contained the famous count bug, which led to incorrect query results in the presence of null values and could have been prevented if formal verification techniques were used."
– Matthias Brantner, Oracle –

AUTOMATED TESTING FOR WEB BROWSERS

BENJAMIN LIVSHITS ALASTAIR DONALDSON

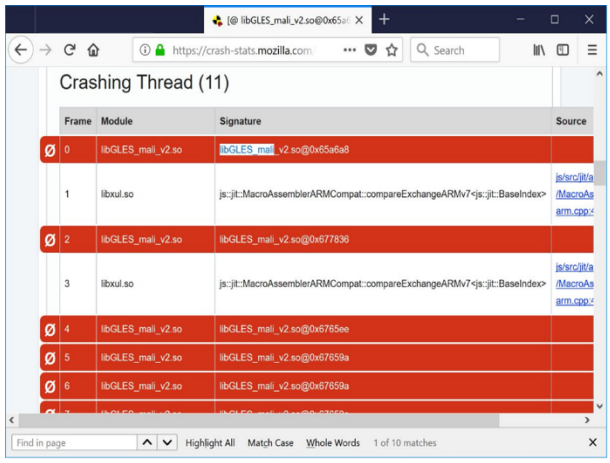
- Web browsers are among the most critical infrastructure on which society depends
- Testing web browsers to find semantic defects is fundamentally challenging
- We have employed mutation-based structural fuzzing to help address this problem, focussing on testing WebGL implementations inside major web browsers

The research work undertaken on this project at Imperial College London led to the development of an automated approach to finding defects in web browsers using mutation-based structural fuzz testing. The investigators decided to focus on testing components of web browsers related to high-performance graphics processing via the WebGL API, because the interaction between web browsers and graphics processing units has become a prominent attack surface in recent years. Two complementary approaches were explored: applying semantics-preserving transformations to WebGL pages to detect rendering problems, where a semantics-preserving change (which, by definition, should have no impact) leads to a change in what is rendered, and applying semantics-changing mutations to a well-formed page in order to test the browser's robustness to adversarial inputs. This led to the discovery and reporting of a number of issues in the Firefox and Chrome browsers, triggered by underlying defects in GPU drivers from a range of vendors. The associated tool in which the techniques are implemented will be open sourced in due course.

The funding from VeTTS was incredibly useful in allowing us to explore this emerging area. The work undertaken will form the basis for future publications, and has put us in a good position to apply for follow-on projects – a research grant from the Google Chrome University Research Program has already been secured (more details below). The work is strongly related to a line of work Donaldson has been pursuing for several years on *metamorphic testing* for graphics compilers, which led to the GraphicsFuzz start-up company (www.graphicsfuzz.com) that was acquired by Google and has since been open-sourced (https://github.com/google/graphicsfuzz). The VeTTS work on fuzzing WebGL has been integrated into GraphicsFuzz, which is actively being used to find defects in the Chrome web browser, including the vulnerabilities linked to below, which have all now been fixed.[1]

PUBLICATIONS. The GraphicsFuzz team is thriving at Google, and just submitted an experience report paper to a leading programming languages conference on the work.

RELATED GRANTS. Dr A. Donaldson, EPSRC Fellowship "Reliable Many-Core Programming", 10/2016-09/2021, £1M. Dr A. Donaldson (Co-I), with C. Cadar (PI), EPSRC Grant "Automatically Detecting and Surviving Exploitable Compiler Bugs", 01/2018-12/2020, £672K. Dr A. Donaldson, Google Chrome University Research Program project "Automatic Detection of Rendering-Related Security Vulnerabilities in Web Browsers", 01/2018-04/2019, £130K.



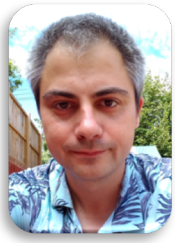
A crash in Firefox caused by a driver bug discovered by our techniques

IMPACT STATEMENT. "From a technical standpoint, the GraphicsFuzz work to which this VeTTS project is closely related has been highly successful in developing basic technologies for improving the security and reliability of billions of deployed mobile devices. From a broader point of view, this work has gotten widespread visibility and, of course, was seen by Google as being so valuable that they bought it."
– John Regehr, Professor, University of Utah –

[1] https://bugs.chromium.org/p/chromium/issues/list?q=metzman_graphicsfuzz_crash_fuzzer%20status%3AVerified&can=1

PRIDEMM
WEB INTERFACE

University of Kent



MARK BATTY RADU GRIGORE

- Prose specifications of relaxed memory behaviour are imprecise and lead to bugs in language specifications, processors, compilers and vendor-endorsed programming idioms
- Mechanised formal models used in academia to unambiguously specify and verify relaxed memory behaviour
- PrideMM is a Solver for Relaxed Memory Models, which improves on state-of-the-art descriptions of the concurrency behaviour of programming languages
- PrideMM provides a platform for comparison, testing, and refinement of relaxed memory models

Modern computer systems have *relaxed memory*: they exhibit highly unintuitive memory behaviour as a result of aggressive processor and compiler optimisations. At the same time, these systems are specified with relatively imprecise prose specifications, leading to bugs in language specifications, deployed processors, compilers and vendor-endorsed programming idioms. A push from academia has, in place of prose, introduced mechanised formal models that unambiguously specify relaxed memory behaviour, together with proofs and simulation tools that allow the validation of key design goals.

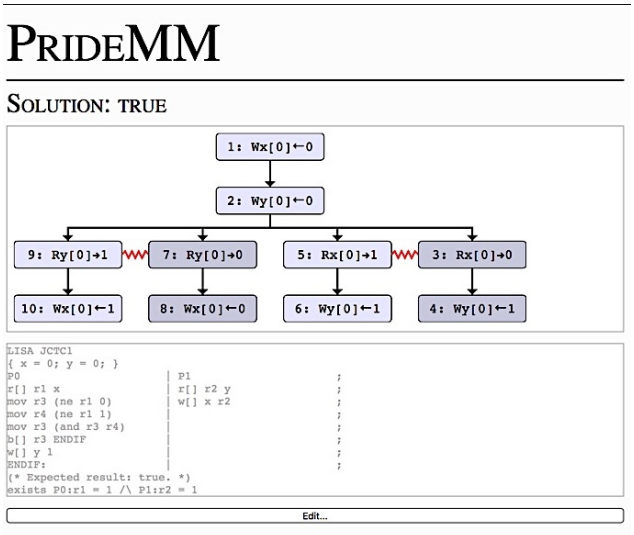
This project concerns PrideMM: a solver that allows one to run tests over state-of-the-art descriptions of the concurrency behaviour of programming languages. Previous relaxed memory simulators were based on ad-hoc backends or SAT solvers. Additional computational complexity arises in cutting-edge language models that must consider multiple paths of control flow, so the simulator backend embodies a problem outside of the scope of SAT. The problem is, however, within the scope of rapidly improving QBF solvers, atop which PrideMM is built.

The Web Interface to PrideMM, available at https://www.cs.kent.ac.uk/projects/prideweb/, is an essential outcome of this project. It allows one to run large batteries of automatically generated tests, and compare its runtime to those of the existing state of the art. The goal of PrideMM is to facilitate discussion with the specifiers of industrial concurrency models, promoting the latest academic solutions to open problems faced by industry.

PUBLICATIONS. [1] M. Batty et al. “PrideMM: A Solver for Relaxed Memory Models”, draft paper on PrideMM, detailing representations of key memory models, a proof-of-concept backend, and a specification language that marries expressiveness and ease of solving. [2] M. Janota, R. Grigore, V. Manquinho. “On the Quest for an Acyclic Graph”, draft paper on finding acyclic graphs under a set of constraints, a general problem central to the PrideMM backend.

RELATED GRANTS. Dr Mark Batty, EPSRC Grant: “Compositional, dependency-aware C++ concurrency”, PI, £98,786, 04/2018-03/2020. Dr Mark Batty, EPSRC Grant “Verifiably Correct Transactional Memory”, Co-I, £82,904, 07/2018-06/2021. PrideMM is the starting point for tools envisaged by these two grants.

IMPACT STATEMENT. “I believe that a well-reasoned memory model is the most important feature of any parallel programming platform, and that Mark Batty’s work has contributed to building confidence in these models more than anyone else’s.”
– Olivier Giroux, Distinguished Architect at NVIDIA, Chair of Concurrency & Parallelism for ISO C++ –



PrideMM screenshot. One specifies a test, model, and outcome and PrideMM works out whether the outcome is allowed or not. “True” indicates the outcome is allowed, and the graph indicates the underlying mathematical structure justifying this outcome.

VERIFYING EFFICIENT LIBRARIES IN CAKEML

SCOTT OWENS

University of Kent

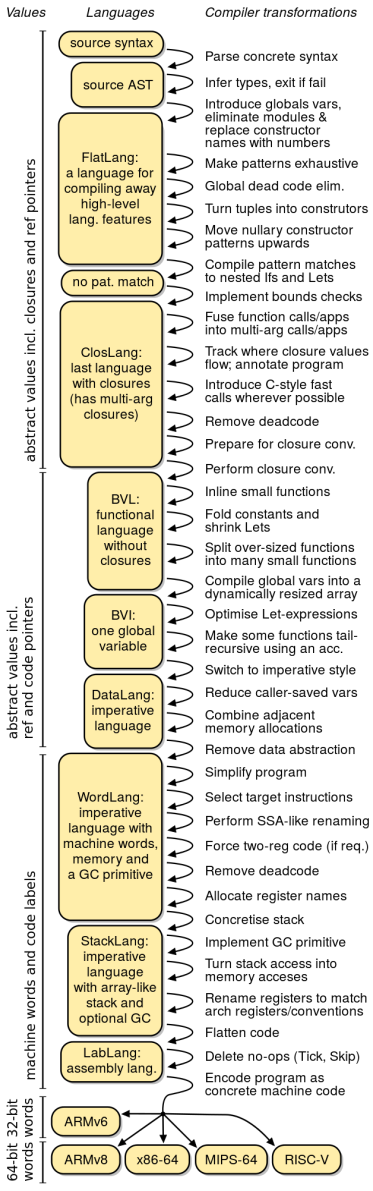
- CakeML is a functional programming language and an ecosystem of associated proofs and tools, including a formally verified compiler to various processor architectures
- CakeML lacks support for verifying libraries that use unsafe features, e.g., array accesses w/o bounds checks
- The RustBelt project (Dreyer) uses the Iris framework to reason about unsafe features of Mozilla's Rust language
- This exploratory project investigated the feasibility of using RustBelt's Iris to verify CakeML programs: we established that it is not possible to use Iris as-is, and that it is necessary to develop an Iris-like logic for CakeML

CakeML is a dialect of the ML family of programming languages, designed to play a central role in trustworthy software systems. The CakeML project is an ongoing collaboration between S. Owens (Kent, UK), M. Myreen (Chalmers, Sweden), and J. Pohjola and M. Norrish (Data61, Australia). The project's main accomplishment is the first fully verified compiler for a practical, functional programming language.

The RustBelt project aims to put the safety of Mozilla's Rust programming language on a firm semantic foundation. Rust's standard libraries make widespread internal use of *unsafe* blocks, which enable them to opt out of the type system when necessary. The hope is that such unsafe code is properly encapsulated, preserving language-level safety guarantees from Rust's type system. However, subtle significant bugs with such code have already been discovered by RustBelt.

This project explored the way in which fundamental mathematical insights from RustBelt could be incorporated into CakeML's suite of verification tools, setting the foundation for follow-up projects with greater scope for more advanced unsafe features, such as C's *malloc* and *free*, or passing CakeML data to C functions. Such features are important, as they bring end-to-end verification to performance-critical areas, such as uni-kernel operating systems, or distributed systems where even (non-end-to-end) verified systems are known to be buggy.

We have established, while Iris technology can, in principle, solve the problems observed in CakeML, we would need to re-design the logical foundations of Iris to accommodate the CakeML proof ecosystem. In particular, the HOL4 theorem prover of CakeML has foundational differences from RustBelt's Coq theorem prover. This is the subject of our subsequent VeTSS project.



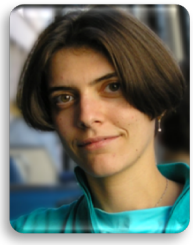
CakeML Infrastructure

RELATED GRANTS. Dr Scott Owens, EPSRC Grant: "Trustworthy Refactoring", 09/2016-03/2020, £728,766.

PUBLICATIONS. H. Férée, J. Å. Pohjola, R. Kumar, S. Owens, M. O. Myreen, and S. Ho "Program Verification in the Presence of I/O Semantics, verified library routines, and verified applications", VSTTE 2018.

IMPACT STATEMENT. "At Rockwell Collins, we use CakeML in projects to build avionics components with formally proven behavioural guarantees: these components have to exhibit high performance. In some cases, this can be achieved by algorithmic transformations already justifiable in CakeML. Beyond that, a great deal more performance can be obtained by unsafe (formally verified) compilation steps, and we are eager to take advantage of such advances when they become available."
– Konrad Slind, Senior Industrial Logician, Rockwell Collins –

SUPERVECTORIZER



GRETA YORSH

- Optimising compilers for Single-Instruction-Multiple-Data (SIMD) architectures rely on sophisticated program analyses and transformations
- Correctness hard to prove due to interaction between optimisation passes and SIMD semantics/costs
- Supervectorizer: integration of *unbounded superoptimization* with *auto-vectorisation* enables software to take full advantage of SIMD capabilities of existing and new microprocessor designs
- Potential for fundamental advances in SMT solvers and industrial-strength SIMD optimizing compilers

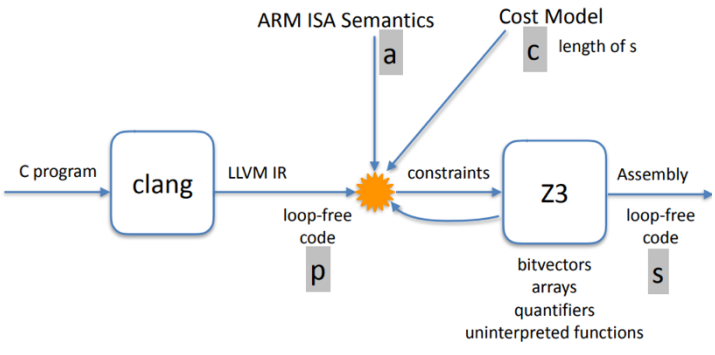
Optimising compilers for Single Instruction Multiple Data (SIMD) architectures rely on sophisticated program analyses and transformations. In particular, auto-vectorisation is designed to automatically identify and exploit data-level parallelism. To deliver expected performance improvements, compiler writers resort to changing optimisation passes, heuristics, and cost models. This process is highly challenging even for the few experts who possess the required range of skills, and any errors introduced affect the entire software stack, likely compromising its reliability and security.

Ensuring correctness of these compiler optimisations is hard due to implicit interactions between optimisation passes and abstruse details of SIMD instructions semantics and costs. It results in missed optimisation opportunities and subtle bugs, such as miscompiled code, which might remain undiscovered for a long time and manifest themselves in obscure ways across abstraction layers of a software stack.

This project aimed at enabling software to take full advantage of SIMD capabilities of microprocessor designs, without modifying the compiler. In particular, we integrate unbounded superoptimization with auto-vectorisation. This approach reduces the engineering effort needed to tune a production compiler for new SIMD architectures and improves compiler reliability without compromising the performance of generated code. We believe that this approach will lead to fundamental advances in SMT solvers and industrial-strength optimising compilers targeting SIMD architectures.

The work done in this project has had the following impact:

- Initial results were presented, by invitation, at Intel's Compiler, Architecture and Tools Conference (CATC).
- Postdoctoral research assistant, Julian Nagele, who joined in January 2018, has been working on a robust prototype implementation and experiments with SIMD instructions. Julian is engaged with the LLVM community and obtained valuable early-stage feedback from developers at EuroLLVM 2018.
- The work on this project has led directly to the award to Dr Yorsh of ERC Starting Grant. Initial results obtained under VeTSS funding demonstrated feasibility of the proposed ERC plan and the work under ERC will build on the infrastructure and experimental results obtained under VeTSS funding.
- The quantitative trading firm Jane Street expressed interest in incorporating techniques developed under this grant into the compiler for OCaml.
- Amazon invited Dr Yorsh to join as Amazon Scholar to work with Amazon Video on tools for improving correctness and performance of their code.



Structure of the preliminary prototype

RELATED GRANTS. Dr Greta Yorsh, ERC Starting Grant, £1.25M, 2018-2022.

EASTEND: EFFICIENT AUTOMATIC SECURITY TESTING FOR DYNAMIC LANGUAGES

JOHANNES KINDER

- Dynamic languages like JavaScript and Python are immensely popular
- Dynamic types and non-standard semantics make security bugs difficult to spot
- EASTEND focused on automated security testing for dynamic languages, in particular JavaScript.
- EASTEND improves the applicability of dynamic symbolic execution for JavaScript code and develops a flexible specification and testing methodology for security properties

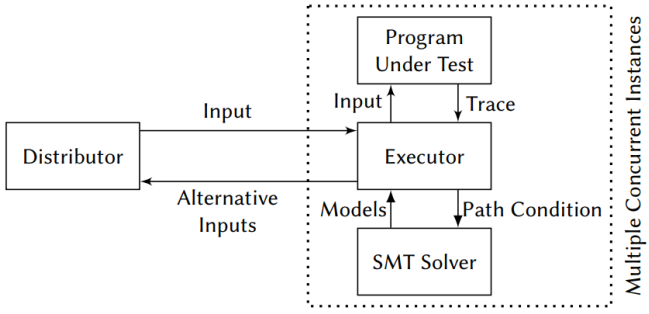
EASTEND is based on the hypothesis that inherently dynamic languages are best served by a dynamic approach to verification that points to errors in the code without restricting the freedom of the developer. It uses test generation via dynamic symbolic execution (DSE) to systematically cover paths through programs and check security properties along those paths. The two main research objectives of EASTEND were: improving the applicability of dynamic symbolic execution (DSE) for real-world JavaScript code (RO1); and developing a flexible specification and testing methodology for security properties that goes beyond simple assertion checking (RO2).

Regular expressions (REs) limit applicability of DSE to testing code security in practical client- and server-side web applications, as modern solvers cannot reason about real-world REs as they are used by developers. We developed an encoding of complex REs and with a refinement scheme that soundly translates REs into the subset supported by state-of-the-art solvers. We implemented our approach in our DSE engine for JavaScript, ExpoSE [1], and evaluated it on 1,131 Node.js packages, demonstrating that the encoding is effective and can increase line coverage by up to 30%. The increased coverage demonstrates that more parts of the program can be reached, increasing the analysis surface for detecting bugs and vulnerabilities, e.g., using the specification and testing methods developed as part of RO2.

We have developed a methodology for specification-based testing of cryptographic applications based on type-like tags attached to runtime values that we call “Security Annotations” (SAs) [2]. We have developed explicit SAs for the widely-used JavaScript library Crypto.JS, which implements common cryptographic algorithms and primitives. These will allow developers using Crypto.JS to automatically inject our annotations into their testing environment at runtime without any expert knowledge required. By using DSE with ExpoSE on a program using an appropriately annotated API, developers will be able automatically detect cryptographic bugs without additional annotation requirements.

PUBLICATIONS. [1] B. Loring, D. Mitchell, J. Kinder. “ExpoSE: Practical Symbolic Execution of Standalone JavaScript”. In Proc. Int. Symp. on Model Checking of Software (SPIN), pp. 196–199, ACM, 2017. [2] D. Mitchell, L. T. van Binsbergen, B. Loring, and J. Kinder. “Checking Cryptographic API Usage with Composable Annotations”. In ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM), 2018. [3] D. Mitchell, J. Kinder. A Formal Model for Checking Cryptographic API Usage in JavaScript. ESORICS (1) 2019: 341-360. [3] B. Loring, D. Mitchell, and J. Kinder. Sound Regular Expression Semantics for Dynamic Symbolic Execution of JavaScript. In Proc. ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI), pp. 425–438, ACM, 2019.

IMPACT STATEMENT. “We have started using ExpoSE as a key component of a research project on privacy-preserving proxy servers. To the best of my knowledge, it is the only existing tool for dynamic symbolic execution of modern real-world JavaScript code.”
– Prof. James Mickens, Harvard University –



Parallel testing architecture of ExpoSE

AUTOMATED REASONING WITH FINE-GRAINED CONCURRENT COLLECTIONS

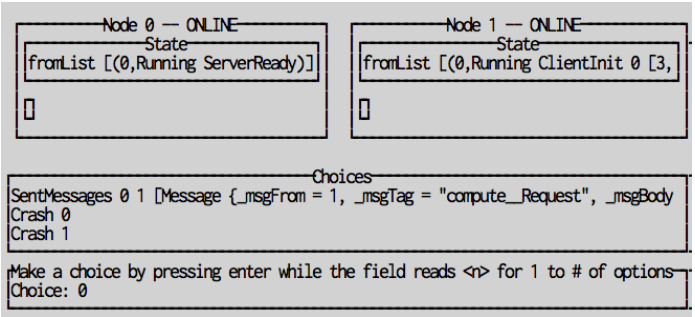
ILYA SERGEY NIKOS GOROGIANNIS

- A domain-specific language (DSL) for concurrent implementations of distributed protocols.
- Prototype DSL implementations of consensus protocols: Two-Phase Commit, Paxos, Multi-Paxos.
- An extension of Disel, a higher-order separation logic for distributed systems to handle concurrent per-node implementations of distributed protocols.

As per the original proposal, the funding has been used to host Kristoffer Just Andersen as a visiting student at the CS department of UCL, where he has worked under our supervision on the applications of techniques for logic-based reasoning about concurrency to the verification of distributed systems with internal multi-threaded parallelism. The project thus naturally evolved from the initially proposed research, elaborating and extending it for the distributed setting. The artefacts produced to date include the runnable prototype (in Haskell) as well as a (partially) mechanised logical development for the verification of multithreaded distributed programs. During Andersen's stay at UCL, Sergey and Andersen developed a domain-specific language for specifying, implementing, randomised testing and visual debugging of distributed protocols.

We have developed *Distributed Protocol Combinators* (DPC), a declarative programming framework that aims to bridge the gap between specifications and runnable implementations of distributed systems, as well as facilitate their modelling, testing, and execution. DPC builds on the ideas from the state-of-the art logics for compositional systems verification. DPC contributes with a novel family of program-level primitives, which allows construction of larger distributed systems from smaller components, streamlining the usage of the most common asynchronous message-passing communication patterns, and providing machinery for testing and user-friendly dynamic verification of systems. The approach has been implemented in a form of a reusable Haskell library, as well as a tool for visual debugging of asynchronous systems.

Declarative programming over distributed protocols is possible and, we believe, can lead to new insights, such as better understanding on how to structure systems implementations. Even though there are several known limitations to the design of DPC due to the chosen linguistic foundations (i.e., Haskell), we consider our approach beneficial and illuminating for the purposes of prototyping, exploration, and teaching distributed system design. In the future, we are going to explore the opportunities, opened by DPC, for randomised protocol testing and lightweight verification with refinement types.



Visual debugging of asynchronous systems using DPC

PUBLICATIONS. [1] K. J. A. Andersen, I. Sergey, "Distributed Protocol Combinators", PADL'19. [2] N. Polikarpova, I. Sergey. "Structuring the Synthesis of Heap-Manipulating Programs", POPL'19.

RELATED GRANTS. Dr Ilya Sergey, EPSRC Grant "Program Logics for Compositional Specification and Verification of Distributed Systems", 01/2017-11/2018, £101,009. Dr. Ilya Sergey, Google Faculty Research Award, "Distributed System Optimisations as Network Semantics Transformations", 2018.

Mechanised Assume-Guarantee Reasoning for Control Law Diagrams via Circus

University of York

Jim Woodcock Simon Foster

- Theoretical reasoning framework for discrete-time part of control-law block diagrams (such as Simulink), based on mathematical semantics of diagrams and capable of dealing with large state spaces
- Contract-based compositional reasoning using refinement for verification of large systems
- Support for reasoning about diagrams with algebraic loops, ignored by most other verification approaches
- Verification of a subsystem of an industrial aircraft cabin-pressure control application

Control-law diagrams are used in industry to model complex engineering systems, such as the many components of modern aircrafts. These systems must be built to the very highest standards possible, and their control laws must be verified to ensure that they behave as required. Our project proposes a general methodology based on mathematical descriptions of diagrams. It is expressive enough both to capture the full range of behaviours required and to be used with other engineering techniques and their own diagrams and notations. Our techniques scale up to tackle verification of large-scale systems. In this VeTSS-funded project, we developed a theoretical reasoning framework for discrete-time blocks of control-law diagrams. As well as giving a mathematical meaning to Simulink (an industry-standard diagrammatic notation for depicting control laws), our framework links to Modelica (another industry standard notation) for multi-model descriptions. Our verification technique relies on computer programs that automatically follow human patterns of reasoning.

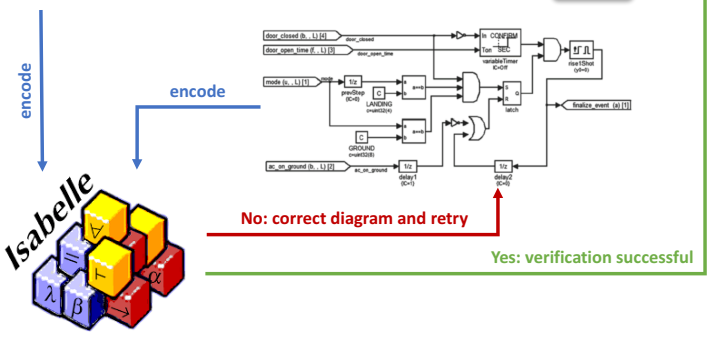
We used our framework to verify the control laws for a subsystem used in aircrafts that controls the cabin pressure after landing. Specifically, the cabin-pressure system must keep working until the aircraft has made a successful landing and the cabin doors have been open for a minimum amount of time. The subsystem is made by Honeywell and we worked with colleagues at D-RisQ. Our technique revealed a vulnerable block that should be improved. The outcomes of this project include a theory to reason about block diagrams using mathematical contracts, mechanisation of the theory in the Isabelle theorem prover, as well as the verification of the cabin-pressure control subsystem. A technical report is available online at http://eprints.whiterose.ac.uk/129640/.

Publications. K. Ye, S. Foster, J. Woodcock. "Compositional Assume-Guarantee Reasoning of Control-Law Diagrams using UTP", From Astrophysics to Unconventional Computation, pp. 215-254, Springer International Publishing, 2020.

Related grants. Dr Simon Foster, EPSRC UKRI Innovation Fellowship: "CyPhyAssure: Compositional Safety Assurance for Cyber-Physical Systems", £562,549, 06/2018–05/2021, with project partners ClearSy and D-RisQ.

Impact Statement. "Simulink is a language highly applied by industry in the development of safety-critical embedded, real-time, and cyber-physical systems, where the establishment of accessible verification support can have substantial impact. This VeTSS project has made a crucial step forward in this area by provision of theorem proving technology in Isabelle/UTP, validated by its application to a real-world aircraft cabin-pressure control application from our company."
– Colin O'Halloran, CEO, D-RisQ –

Requirement 1	A finalize event will be broadcast after the aircraft door has been open continuously for `door_open_time` seconds while the aircraft is on the ground after a successful landing	Verified
Requirement 2	The finalize event is broadcast only while the aircraft is on the ground	Verified
Requirement 3	The finalize event will not occur during flight	Verified
Requirement 4	The finalize event will not be enabled while the aircraft door is closed	Verified



Workflow: from textual representation to formal verification

VeTSS 2018/2019

A Foundation for Testing and Verifying C++ Transactions

John Wickerson
Imperial College London

Session-type-based Verification Framework for Message-passing in Go

Nobuko Yoshida
Imperial College London

Specification and Verification of C++ Data-structure Libraries

Mark Batty
University of Kent

Trustworthy Software for Nuclear Arms Control

Andy King
University of Kent

Building Verified Applications in CakeML

Scott Owens
University of Kent

Operating System Components as Verified Libraries

Tom Ridge
University of Leicester

Formal Verification of Quantum Security Protocols using Coq

Raja Nagarajan
Middlesex University London

Towards Optimised Taint Analysis

Daniel Kroening
Oxford University

SuperVectorizer (Phase II)

Greta Yorsh
Queen Mary University of London

Automated Black-box Verification of Networking Systems

Alexandra Silva
University College London

A Foundation for Testing and Verifying C++ Transactions

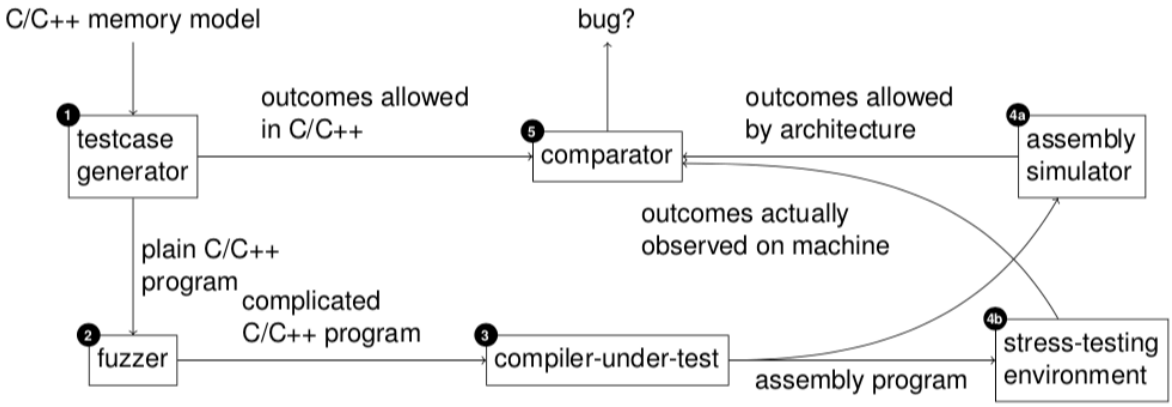
Imperial College London

John Wickerson

- Concurrent programming is hard, but transactional memory promises to make it simpler.
- Transactional memory lets programmers make a group of instructions execute 'instantaneously' by marking them as a 'transaction'.
- However, transactional memory is notoriously hard to specify and implement correctly in compilers.
- The aim of this project is to build a framework for testing whether or not compilers are implementing this form of concurrency correctly.

The scope of this project has broadened considerably compared to the original workplan. The original focus was on testing the compilation of transactional C/C++ concurrency. However, we soon realised that the proposed approach could be applied to test the compilation of any C/C++ concurrency, in a way that has never been tried before, but which has the potential to be a hugely valuable technique for improving compiler reliability. This generalisation has the potential to greatly amplify the impact of our work.

We are developing a tool that uses a combination of techniques (automatic testcase generation, semantics-preserving code mutation, and exhaustive simulation) to search for bugs in the way mainstream compilers translate concurrent C/C++ (including atomic operations). The tool is currently at the prototype stage, and we hope soon to start using it 'in anger' to find bugs. The diagram illustrating the structure of the tool is shown below.



Publications. Our prototype tool has been presented at a scientific meeting at the UCL Computer Science department (November 2018) and at the S-REPLS programming languages workshop (February 2019). An article about its design is in preparation, and will be submitted to a leading conference in the field.

Related grants. This project is also being partially funded by the EPSRC Programme Grant "IRIS: Interface Reasoning for Interacting Systems" (£6.1M, 2018–2023).

Impact Statement. "The programmability of concurrent systems, especially under weak-memory models, is an important challenge for Arm. This is an active area of interest to Arm, and we are delighted to see work that looks at a fuller formalisation of C++ transactional memory."
– Nathan Chong, formerly Principal Researcher at Arm, now Principal Engineer at Amazon Web Services –

SESSION-TYPE BASED VERIFICATION FRAMEWORK FOR MESSAGE-PASSING IN GO

Imperial College London

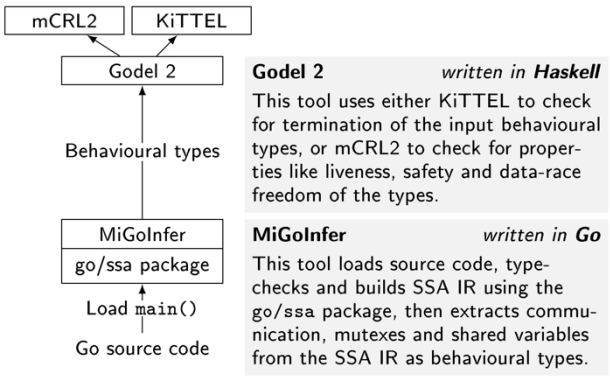
NOBUKO YOSHIDA

- Concurrent programs introduce a specific class of bugs relating to memory safety and deadlocks.
- Detection of concurrency bugs in Go and other concurrent languages often relies on runtime analysing with fuzzing techniques, making it unreliable and allowing bugs to make it to the production software.
- Static detection frameworks allow to reduce the fraction of those bugs that make it to production stages, by consistently analysing a session types model of the source code against properties based on behavioural and memory models of the target language.
- Coupling both techniques allows for an increase in robustness for the development chain of trusted software, ensuring a lesser number of critical bugs end up in production.

Go is a concurrent programming language developed in recent years by Google. It is used by various projects, including cloud service providers like Twitter and Dropbox, open source projects including Docker, Kubernetes and CoreOS. This increasing popularity is reflected on Go's position in several programming language's rankings, including IEEE Spectrum Top Programming Languages (from place 20 in 2014 to 9 in 2017), StackOverflow's most loved and most wanted languages (5th and 3rd place respectively, in 2018), and Github's top growing languages (where it ranked 7th in 2018). Go's most appreciated features are notably its concurrency features, including channel-based message-passing and lightweight thread creation features. These features, however, make programmers struggle with bugs such as communication deadlocks, message mismatches or memory safety issues.

This project builds on the foundations laid by works on detection of channel-based concurrency issues, and brings them further by proving the theoretical base of these works and extending it greatly. Our recent work tackles both channel-based concurrency issues (including deadlocks and safe channel usage) and shared memory-based issued, especially revolving on the correct usage of mutual exclusion locks and data race detection. We use Go's official memory model detailed in the documentation of the language, extracting from it a happens-before relation that is used to define how a data race can be statically detected.

We also formally prove the equivalence between properties of our abstracted types and properties of the source language, defining precisely what conditions programs need to meet so they can be correctly analysed by our framework. This framework is then implemented in a tool, Godel 2, the workflow of which is described in the figure on the right. It uses the mCRL2 model checker and the KiTTEL termination checker to verify the properties we extract from the code against the model behavioural-types we infer from program source code.



PUBLICATIONS. D. Castro, R. Hu, S. Jongmans, N. Ng and N. Yoshida, Distributed Programming Using Role Parametric Session Types in Go, 46th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL), 2019.

RELATED GRANTS. Nobuko Yoshida, PI, EPSRC Established Career Fellowship: "POST: Protocols, Observabilities and Session Types", 04/2020-03/2025, £1.46M; JSPS Invitation Fellowship for Research in Japan, £7,000, 07-08/2019.

IMPACT. The approach used in this project can be extended to a toolchain, Scribble, a protocol description language based on Multiparty Session Types, that is used to design and verify protocols and their implementations. Scribble is used by teams and projects in companies such as Red Hat and Estafet to generate deadlock-free microservices in Go (http://estafet.com/scribble/).

Specification and Verification of C++ Data-Structure Libraries

Mark Batty

University of Kent

- Concurrency in the C++ language is ill-specified in the current International Standards Organisation (ISO) definition: it allows values to be conjured out of thin air (OOTA).
- Working towards verification of C++ code, we fixed this problem with a proposed change to the standard called Modular Relaxed Dependencies (MRD).
- We presented our proposal to the ISO, and they voted unanimously to pursue it.
- We developed a refinement relation that can be used to verify C++ code.
- Further flaws recently uncovered in the standard must be rectified before full verification of data structure libraries is possible.

C and C++ semantics: an ongoing academic/industrial effort. The memory behaviour of modern systems is extremely subtle. Processor vendors avoid the cost of fully hiding micro-architectural details, such as buffering and speculation, by permitting unintuitive program executions. Compiler optimisations alter accesses to memory to similar effect. The end result is a system with relaxed memory behaviour: behaviour that deviates from sequential consistency (SC), where concurrent memory accesses are simply interleaved.

Relaxed memory breaks intuitions about system behaviour leading to bugs in language specifications, deployed processors, compilers, and vendor endorsed programming idioms – it is clear that current engineering practice is severely lacking.

We have an ongoing academic/industrial partnership with the International Standards Organisation (ISO) that has exposed fundamental flaws in the way we specify programming languages. We have shown that the state-of-the-art definitions of C and C++ concurrency are broken -- a problem that stems from a tension between performance and the strength of ordering guarantees provided to programmers. Compilers optimise away some syntactic dependencies, but these programming patterns are an idiomatic way to provide ordering in machine code, and if they are left in place, they serve as a cheap source of ordering at the language level. The language definition must specify which dependencies are left in place: too many, and useful optimisations will be forbidden, as in Java Hotspot; too few, and the semantics of the language permits bizarre behaviour, as in C++. It is provably impossible to strike this balance in C++ by making only minor changes to the concurrency design, so a different approach was necessary. To reason about code, we must fix these problems.

As part of this VeTSS project, we developed Modular Relaxed Dependencies, a model for C++ concurrency which is recognised by the ISO as the best solution to the pernicious OOTA problem. We went on to develop a refinement relation for this model that allows one to verify the correctness of C++ code. Most data structures use pointers, and as part of this work, we started to examine further newly-recognised flaws in the specification of pointers in C++.

Publications. M. Batty, S. Cooksey, S. Owens, A. Paradis, M. Paviotti, and D. Wright. Modular relaxed dependencies: A New Approach to the Out-of-Thin-Air Problem. To appear, ESOP 2020.

Impact Statement. The ISO unanimously passed the following motion endorsing our semantics: "*OOTA is a major problem for C++, modular relaxed dependencies is the best path forward we have seen, and we wish to continue to pursue this direction.*"

TRUSTWORTHY SOFTWARE FOR NUCLEAR ARMS CONTROL

ANDY KING

University of Kent

- We have developed new analysis techniques that are able to correctly handle machine arithmetic for integers of various width.
- This is crucial for reasoning about how paths can be taken, and cannot be taken, through binary programs.
- Our approach allows the model checker itself to be verified by using a classic bit-vector solver, which does not need to support interpolation.

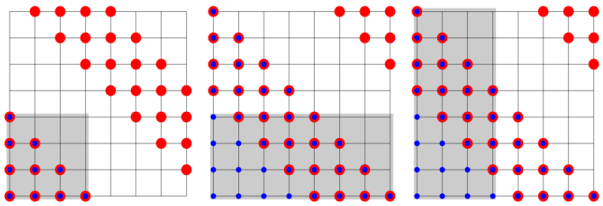
The project focuses on the problem of how to automatically compare two AVR binaries. Using one given binary as the reference semantics, the problem is to determine whether the other binary has the same behaviour (even if it syntactically different), or whether it has been tampered with. We use interpolation-based model-checking to search for a path in one binary which does not exist in the other.

We have developed a new interpolation method for bit-vector formulae by leveraging on existing interpolation techniques for linear integer arithmetic, and integrated this method into the Impact interpolation algorithm, demonstrating how it improves on interpolation techniques which do not reason about the wrap-around nature of machine arithmetic. Interpolation is used to relax a sequence of symbolic formulae which represent a path through a program to give a more general sequence that describes, not just one path, but many.

We have also shown how interpolant methods can be extended beyond systems of linear constraints, and also how to take an interpolant which is a system of linear inequalities and always encode it as a compact bit-vector formula. We have improved an existing interpolation algorithm for bit-vectors by adapting computer algebra techniques to work over modulo arithmetic rather than the field of real numbers. Paradoxically, modulo arithmetic makes it easier, not harder, to automatically reason about the behaviour of the program.

Our analysis work is built on atop a tool-chain for decompiling AVR binaries which is, in turn, build on the QEMU toolkit for emulating various architectures. While undertaking our project, we have made contributions to the AVR support for QEMU, both making it both more robust and extending its functionality.

This research is of key importance to AWE's work on treaty verification in an arms control context. It will pro-vide tools and techniques to verify the authenticity of monitoring equipment that could be deployed for future arms control treaties.



Integer solutions to the inequality $x + y - 4 \leq 3$. In linear integer arithmetic (LIA), these are the same as the solutions to $x + y \leq 7$, but its modulo solutions (in this case, mod 8) are easier to represent as bit-vector formulae. This gives way to converting LIA interpolants into bit-vector interpolants.

PUBLICATIONS. [1] Mind the Gap: Bit-vector Interpolation recast over Linear Integer Arithmetic, T. Okudono and A. King, International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'20); (2) Polynomial Analysis of Modulo Arithmetic, T. Seed, C. Coppins, A. King and N. Evans (under review); [3] Improved AVR support for QEMU, S. Harris, E. Robbins (signed off by M. Rolnik), https://lists.sr.ht/~philmd/qemu/, 2019.

RELATED GRANTS. Atomic Weapons Establishment. Follow-on funding, 1st December 2019-March 30th 2020, £20K.

IMPACT STATEMENT. "As a professional reverse engineer and binary analysis tool developer with 16 years of experience, I believe that the future of binary analysis lies in the further adoption and advancement of techniques in mathematical program analysis. In my estimation, the research group at the University of Kent is one of the only remaining groups – certainly the most prolific one – that is regularly attempting to tackle the theoretical issues plaguing the scalability of binary analysis. Not only are they "attempting" to tackle these issues, they have published a number of deep and fascinating papers, which offered truly novel contributions to binary analysis."
– Rolf Rolles, Möbius Strip Reverse Engineering –

BUILDING VERIFIED APPLICATIONS IN CAKEML

SCOTT OWENS

University of Kent

- CakeML is a functional programming language and an ecosystem of associated proofs and tools, including a formally verified compiler to various processor architectures
- CakeML lacks support for verifying libraries that use unsafe features, e.g., array accesses w/o bounds checks
- The RustBelt project (Dreyer) uses the Iris framework to reason about unsafe features of Mozilla's Rust language
- In our previous VeTSS project, we determined that it was not feasible to use Iris as-is for CakeML. Thus, we have refocussed on foundational research, attempting to apply Iris-like ideas to a CakeML-style lambda calculus. This is a 3-year project supporting a PhD student, the results of which will be applied to CakeML proper.

CakeML is a dialect of the ML family of programming languages, designed to play a central role in trustworthy software systems. The CakeML project is an ongoing collaboration between S. Owens (Kent, UK), M. Myreen (Chalmers, Sweden), and J. Pohjola and M. Norrish (Data61, Australia). The project's main accomplishment is the first fully verified compiler for a practical, functional programming language.

The RustBelt project aims to put the safety of Mozilla's Rust programming language on a firm semantic foundation. The standard libraries of Rust make widespread internal use of *unsafe* blocks, which enable them to opt out of the type system when necessary. The hope is that such unsafe code is properly encapsulated, preserving language-level safety guarantees from Rust's type system. However, subtle significant bugs with such code have already been discovered by RustBelt.

Our previous VeTSS project explored the way in which fundamental mathematical insights from RustBelt could be incorporated into CakeML. It established that the Iris technology can, in principle, solve the problems for CakeML, but that the style of formalisation used in Coq for Iris was not feasible in CakeML.

The PhD student on the project, Hrutvik Kanabar, has built a model of System F in HOL4 in the style of CakeML. He has established an Iris-style semantic model of CakeML types using a logical relation allowing verified code that uses unsafe features to smoothly co-exist with unverified, but type-safe, code. He has also identified several more application areas for the technology, including the development of a verified pathway for executing code extracted from the Coq proof assistant, which currently makes ample use of OCaml's unsafe type casts.

RELATED GRANTS. Dr Scott Owens, EPSRC Grant: "Trustworthy Refactoring", 09/2016-03/2020, £728,766.

PUBLICATIONS. H. Férée, J. Å. Pohjola, R. Kumar, S. Owens, M. O. Myreen, and S. Ho "Program Verification in the Presence of I/O Semantics, verified library routines, and verified applications", VSTTE 2018.

IMPACT STATEMENT. "CakeML is an important strategic research project for Data61. The 'strategic' tag means that we support it as a future-looking endeavour that does not necessarily have a short-term payoff, nor necessarily receives external funding. Instead, we pursue such projects because of our belief that they have high potential for future impact. That said, some of our work on CakeML does receive external funding in the form of money from a multi-million dollar DARPA-funded research project (CASE: https://www.darpa.mil/program/cyber-assured-systems-engineering and http://ts.data61.csiro.au/projects/TS/CASE/). This project sees CakeML being integrated into high-assurance systems development, in collaboration with staff at Collins Aerospace. CakeML's ongoing development makes it a stronger and stronger component (better performing and more featureful) in existing and future systems of this sort. As we work with research collaborators around the world and continue to produce CakeML-related publications in the academic press, we also continue to demonstrate our commitment to the CakeML project as a vehicle for pure research."
– Michael Norrish, Principal Researcher, Data61, Australia –

Operating Systems Components as Verified Libraries

UNIVERSITY OF LEICESTER

Tom Ridge

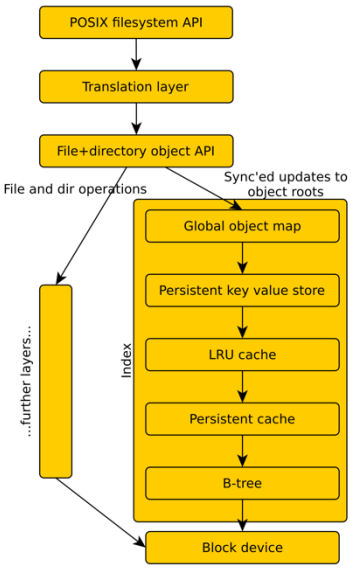
- Core operating system functionality has been verified in projects such as L4.verified, for the seL4 microkernel.
- However, two other major components should also be verified: the network stack and the file system.
- A few verified file systems already exist, but their performance is slow compared to traditional file systems such as ext4 and ZFS. Moreover, they lack important features of modern file systems, such as file system snapshots.
- We aim to develop a verified file system "ImpFS", constructed from small, well-defined components. ImpFS should match or exceed existing file systems, both in terms of performance and features. It will be available both as a desktop file system and as a library of components suitable for use in other software, and even in library-based unikernels such as MirageOS.

In previous work, we developed SibylFS, a formal model of POSIX and real-world file systems. The formal specification was usable directly as a test oracle, to check conformance of existing file systems. We now aim to actually implement a verified file system.

Building a file system is hard. Building a high-performance file system with advanced features is extremely difficult: the BTRFS file system has been in development by a team of engineers for around 10 years and is still not considered stable enough for production use. Thus, we should expect that building a verified high-performance file system will be extremely challenging.

The most important components that we have developed so far are:

- A high-performance novel B-tree-like data-structure with both Copy-on-Write and Mutate-in-place semantics. This is formalised in Isabelle/HOL and extracted to OCaml for execution.
- A persistent cache, offering transactional-log-like functionality.
- A persistent key-value store.



The Design of ImpFS

These components are freely available online from http://www.tom-ridge.com/filesystems.html. All the components are implemented in a purely-functional style, which is a pre-requisite for easy verification. The performance of the components is extremely good. For example, the key-value store matches the performance of the well-regarded LMDB key-value store.

Recently, we have finalised the design for our ImpFS file system, and are in the process of implementing the design, prior to benchmarking. We are confident that the file system will out-perform existing file systems in many areas. More importantly, the components will serve as the basis for many further interesting file system designs.

The VeTSS funding has been critical for securing research time for the author. Most of the work has involved high-level design and low-level component implementation. In addition, a lot of effort has been put into performance engineering of the components. This work is lengthy and would not have been possible without the financial support provided by VeTSS.

Publications. [1] A B-tree library for OCaml. T. Ridge. ICFP 2017, OCaml'17 workshop; [2] Towards verified file systems. A. Giugliano, 2018. PhD thesis; [3] A Key-Value store for OCaml. T. Ridge. ICFP 2019, OCaml'19 workshop.

Impact Statement. "Together with a PhD student, I have been working on a verified copy-on-write filesystem. In the course of our work, we have relied heavily on Dr. Tom Ridge's ImpFS work. The availability of an existing verified copy-on-write B-Tree implementation has been a great help to our work. We continue to use it heavily as reference point for not only our implementation, but also for an example of formally specifying this crucial class of data structures."
– Colin S. Gordon, Drexel University, Pennsylvania –

Formal Verification of Quantum Security Protocols using Coq

Jaap Boender

Richard Bornat

Florian Kammüler

Middlesex University London

Rajagopal Nagarajan

- Quantum computing and communication systems are increasingly becoming practical and are likely to revolutionise modern technology
- Formal methods have been extremely valuable in ensuring correctness as well as security of classical systems and are widely used in industry
- The aim of this project has been to use the proof assistant Coq to verify quantum communication and cryptographic protocols
- Qtpi, our implementation of the quantum process calculus CQP, allows for rapid prototyping

The aim of this project is to use the proof assistant Coq to verify quantum communication and cryptographic protocols. In earlier work, we had formalised qubits and quantum operations in Coq. During this project, we implemented a Quantum IO monad in Coq for the specification of the protocols. In addition to quantum gates and measurement, the monad gives us a lightweight process calculus which supports sequencing of operations and keeping of state. We have proved this monad has the necessary properties. The process simulation function that gives the QIO monad its semantics has also been written. We have been proving properties of simple quantum protocols.

As an example, we show the formalisation of the main theorem in Coq that proves that the quantum teleportation protocol actually transmits Alice's qubit ϕ to Bob. We do not show the formalisation of Alice and Bob here, but the aim is to show that the combination of Alice's and Bob's functions results in a triple of qubits whose last element is the same as ϕ. The theorem states that, for each of the four possible outcomes for q which is an instance of ϕ, a suitable z exists.

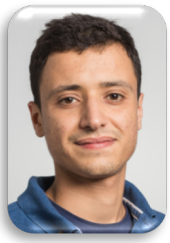
```
Theorem teleportation:
forall phi: qubit 1, forall q: qubit 3,
forall pr: IR, forall qp: List.In (pr, q)‘(Alice phi),
 exists z: qubit 2,
   ‘Bob(q, Alice_out phi q (Alice_case phi q pr qp)) {=} ‘(z {o} phi).
```

In preparation for the Coq verification, teleportation and many other protocols were specified and analysed using Microsoft's Q# and our own symbolic evaluator, Qtpi, which has been developed within the timeframe of this project. Qtpi is an implementation of the quantum process calculus CQP. It is more suited to modelling distributed computation than Q#. It also uses symbolic rather than numeric quantum calculation. Programs are checked statically, before they run, to ensure that they obey real-world restrictions on the use of qubits (e.g. no cloning, no sharing). Qtpi should be of independent interest as a quantum programming language implementation and is available from GitHub (https://github.com/mdxtoc/qtpi).

Given the complexity and novelty of this project, considerable progress has been made in about a year. Quantum computation and communication are becoming increasingly industrially relevant, with companies such as IBM, Microsoft, Intel and Google taking the lead in building quantum computers. We expect that our project will be able to make an impact in the near future.

Publications. [1] R. Bornat, J.Boender, F. Kammueller, G. Poly and R. Nagarajan, "Describing and Simulating Concurrent Quantum Systems", Tool Demonstration Paper, In TACAS '20: 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2020. An extended version of this paper has been submitted for publication elsewhere and is under review. Also, the poster based on the above work was presented at the Verified Software Workshop at the Newton Institute in Cambridge, September 24-25, 2019; the 19th Asian Quantum Information Science Conference (AQIS '19), Seoul, South Korea, August 19–23, 2019; and the International Conference on Quantum Computing (ICoQC '18), Paris, November 26–30, 2018.

TOWARDS OPTIMISED TAINT ANALYSIS



UNIVERSITY OF OXFORD

DANIEL KROENING JOHN GALEA

- Generic Taint Analysis is a flexible technique that enables the enforcement of different taint policies via the same underlying taint tracking system.
- However, generic taint analysis incurs severe performance overheads.
- We propose the Taint Rabbit, an optimized generic taint engine capable of analysing x86 binary applications. Its enhanced performance is based on optimizations that we have investigated and relate to fast path generation and vectorization. Overall, the work done acts as a foundation for further research on security vulnerabilities.
- Results show that with our optimizations, the Taint Rabbit performs faster than other existing generic trackers.

Dynamic taint analysis is a pivotal technique in software security that enables the tracking of interesting/suspicious data as it flows during execution. With the essential funding provided by VETSS, we have researched approaches for optimizing an expensive but generic variant of the analysis. Crucially, the analysis supports various user-defined policies via the same underlying taint tracking system. The research is carried out in an effort towards our long-term goal of automatically detecting and analysing software vulnerabilities and reasoning over their exploitation.

The main performance bottleneck of taint analysis stems from the execution of taint propagation code that is intensively instrumented into the target application at instruction granularity. Unlike specialised bitwise tainting, a generic taint tracker cannot be optimised for a specific taint policy. Instead, it must perform elaborate propagation in order to be versatile. At a high-level, we adopt two strategies to address the performance issue. First, we aggressively elide the execution of propagation routines whenever possible. This is achieved by generating fast paths that result in basic blocks being instrumented based on frequent taint contexts identified at runtime. Second, we directly optimize the code that is responsible for actually conducting taint propagation. In particular, we leverage vectorization so that all taint information pertaining to source operands of a given instruction are processed simultaneously.

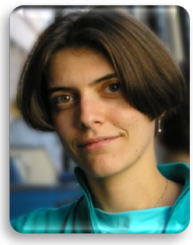
Our research has led to the development of the Taint Rabbit, a novel generic taint tracker that uses our proposed techniques. We evaluated our approach on a number of real-world applications including Apache, PHP, and bzip2, as well as on CPU-intensive benchmarks such as SpecCPU 2017. Results indicate that the Taint Rabbit is the fastest generic taint engine amongst those we assessed. Furthermore, to demonstrate the flexibility of the Taint Rabbit, we also developed several taint-based applications using our versatile system despite their dependence on different taint propagation policies. In particular, we considered Use-After-Free debugging, control-flow hijack detection, and vulnerability discovery through fuzzing. Overall, VETSS has given us the opportunity to engage in imperative research which resulted in generic taint tracking to scale better for binaries than the current state-of-the-art. The Taint Rabbit serves as a vital stepping-stone to automatically analyse and understand security-critical software vulnerabilities.

PUBLICATIONS. A research paper is under submission for publication at a high-ranking security conference. Additional papers are also currently being written.

RELATED GRANTS. Our research is partially funded by EPSRC (EP/P00881X/1) and the ENDEAVOUR Scholarship Scheme (2015-2019).

IMPACT. The Taint Rabbit and all tools built upon it will be made open-source upon publication. We have also made several contributions to DynamoRIO, the open-source DBI system that the Taint Rabbit uses. In this regard: "John has had significant impact on the open-source DynamoRIO project: he has contributed numerous fixes and features to the code base; he has joined the small set of core developers who voluntarily help maintain the continuous integration testing and other infrastructure; he has influenced design decisions for new features by other developers; and he has helped to build the community around this project." – *Derek Bruening, Software Engineer, Google* –

SUPERVECTORIZER (PHASE II)



GRETA YORSH

- Optimising compilers for Single-Instruction-Multiple-Data (SIMD) architectures rely on sophisticated program analyses and transformations
- Correctness hard to prove due to interaction between optimisation passes and SIMD semantics/costs
- Supervectorizer: integration of *unbounded superoptimization* with *auto-vectorisation* enables software to take full advantage of SIMD capabilities of existing and new microprocessor designs
- Potential for fundamental advances in SMT solvers and industrial-strength SIMD optimising compilers

The original aim of the project was to apply unbounded superoptimization to the problem of generating SIMD code. This approach results in faster code than what can be generated using traditional compiler optimization technique called vectorization. With this approach, the problem is encoded in first-order constraints and solved using an SMT solver which has to be extended with special heuristics.

As a prerequisite, the PI started the theoretical development of a framework for symbolic cost models of modern compute architecture, targeting ARM8 as the first experimental platform. The RA supported by this grant, Julian Nagele, worked on the improvement of the initial prototype to make it more reconfigurable and extensible, and evaluation on small, but important benchmarks.

Unfortunately, the PI had substantial health problems, and could not continue to be involved in the work. Without the support of the PI, the RA was not able to carry out the work on SIMD code generation, which required expert knowledge of vectorization, ARM architecture, and SMT solvers.

However, the RA realised the potential of applying this technology to smart contracts. He took a leading role on this research, identifying the direction of blockchain, and brought the work to completion. The PI provided high-level guidance, but was not actively involved in this work. The RA independently established a collaboration at UCL with a group interested in start contracts verification, which motivated this work on optimization. The paper has been accepted for publication at LOPSTR'19. The reviewers recognised the importance of the application domain of smart contracts, novelty of superoptimization in this context, the extensive specification and benchmarking produced by the authors. The prototype and benchmarks are available as open-source (https://github.com/juliannagele/ebso). The RA and the PhD student are preparing a journal version of the paper. Recently, there has also been interest in this prototype from industry (https://www.embecosm.com).

Upon completion of this project, the PI has gone on to work at the global proprietary trading firm Jane Street, whereas the RA has gone on to work at Bank of America.

PUBLICATIONS. Julian Nagele, Maria A. Schett. Blockchain Superoptimizer. 29th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2019).

RELATED GRANTS. Dr Greta Yorsh, ERC Starting Grant, £1.25M, 2018-2022.

AUTOMATED BLACK-BOX VERIFICATION OF NETWORKING SYSTEMS

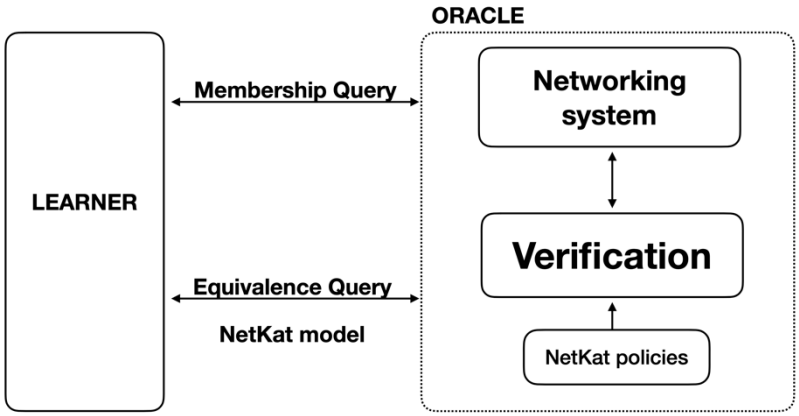
ALEXANDRA SILVA MATTEO SAMMARTINO

- Modern technology relies on complex and safety-critical networking systems
- Automated verification techniques can detect unintended network behaviour and security vulnerabilities
- Our approach incrementally learns a state-based model from a network by observing its behaviour
- Long-term objectives include achieving scalability of learning and verification of network programs

As complex networking systems, such as Software Defined Networks, Cloud Computing and the Internet of Things, become more and more popular, automated verification tools capable of assessing the security and reliability of such systems are in high demand.

Classical approaches to verification tasks require the existence of a model of the system of interest, able to express all its relevant behaviour. Unfortunately, in reality such a model rarely exists, as networking systems may be extremely heterogeneous and parts may lack a formal specification, or the manual construction of a model is simply unfeasible.

Our approach aims to address this issue by automatically inferring a model of a network in a black-box fashion, by observing its behaviour. The goal is to provide a modular framework for learning and verifying networking systems, based on the NetKat language, which provides a compositional programming abstraction of networking behaviour. The envisioned framework is shown in the diagram: the learner interacts with the system via queries, aiming to gather observations (membership query) and to check the correctness of the model (equivalence query); the oracle includes a verification component, which compares the current model against NetKat policies, allowing one to check for specific properties incrementally, as the model is being learnt.



The main objectives for this project were taming the complexity of networks and achieving scalability of learning and verification. We have achieved this by investigating efficient fragments of the NetKat language, and associated learning and verification techniques.

PUBLICATIONS. [1] Learning weighted automata over principal ideal domains. Gerco van Heerdt, Clemens Kupke, Jurriaan Rot and Alexandra Silva. FOSSACS 2020. To Appear. Another paper to be submitted to a top conference is currently in preparation.

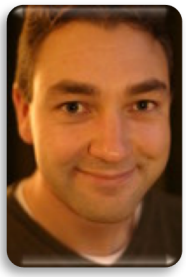
RELATED GRANTS. EPSRC Standard Grant "Verification of Hardware Concurrency via Model Learning (CLeVer)" (EP/S028641/1), £693K.

IMPACT STATEMENT. "The completeness, fidelity, and trustworthiness of models is an important challenge for Arm. Arm is highly interested in the development of techniques that offer the potential to make the design of these models more automatic - both tools that provide a design aid for human designers, and tools that automate the modelling process altogether."
– Dominic Mulligan, Staff Research Engineer, Arm Research –

VeTSS 2019/2020

Higher-Order Program Invariants and Higher-Order Constrained Horn Clauses

Steven Ramsay
University of Bristol



Verifying performance impacts of micro-architecture vulnerability mitigations

David Aspinall
University of Edinburgh

Mechanising the Theory of Build Systems

James McKinna
University of Edinburgh

Reliable High-Level Synthesis

John Wickerson
Imperial College London

Fluid Session Types: End-To-End Verification of Communication Protocols

Nobuko Yoshida
Imperial College London

Generalised static checking for type and bounds errors

Stephen Kell
University of Kent

Formal Verification of Information Flow Security for Relational Databases

Andrei Popescu
Middlesex University London

Automated Testing of Adversarial Safety for Deep Neural Networks

Youcheng Sun
Queen's University Belfast

Predictive Monitoring of Cyber-Physical Systems with Machine Learning Models

Nicola Paoletti
Royal Holloway Univ. of London

Persistent Safety and Security

Brijesh Dongol
University of Surrey

VeTSS

Research Institute in Verified Trustworthy Software Systems

UK's second research institute in cyber-security

Contact us

Petar Maksimović

Academic Program Manager

Teresa Carbajo García

Administrative Program Manager

Research Institute in Verified Trustworthy Software Systems

Department of Computing, Imperial College London
South Kensington Campus, London SW7 2AZ
United Kingdom

Phone: +44 (0)20 759 43140

E-mail: VeTSS@imperial.ac.uk

EPSRC
Engineering and Physical Sciences Research Council

National Cyber Security Centre
a part of GCHQ

Imperial College London