Informatics PhD projects at King's College London, AY 24-25 — Systems (SE, programming, autonomous systems, robotics, ...)

The PhD project proposals listed below will be considered for 2024/25 studentships available in the Department of Informatics to start 1 October 2024 or later during the 2024/25 academic year. Please note that this list is not inclusive and potential applicants can alternatively identify and contact appropriate supervisors outlining their background and research interests or proposing their own project ideas.

The PhD projects are listed in two groups. In the first group are the projects with allocated studentships: each project in this group has one allocated studentship. The remaining studentships will be considered for the projects listed in the second group. The number of those remaining studentships is smaller than the number of the projects in the second group. The allocation of studentships will be based on the merits of individual applications. Applications for PhD studies in the Department of Informatics, for all listed projects as well as for other projects agreed with supervisors, are also welcome from students applying for other funding (within other studentship schemes) and from self-funded students. See also this list of funding opportunities available at King's for post-graduate research in Computer Science.

- Scholarship Allocated
- Scholarship Not Allocated

Scholarship Allocated

(Back to Top)

- Infrastructure for productive multi-language programming
- Visual live programming in scientific computing and similar domains
- Studying induced demand in software performance
- Reliability and verification of software for scientific and ML computing
- Improving Understandability of Automatically Generated Test Cases using Text-to-Text Transformer Models
- Estimating the ground truth of LLMs in softare engineering Tasks

Infrastructure for productive multi-language programming

Supervisor: Stephen Kell

Areas: Systems (SE, programming, autonomous systems, robotics, ...)

(Back to Scholarship Allocated)

Project Description

It is usually difficult to combine code written in different languages, especially one or more higher-level languages involving a garbage collector. Typically, programmers must write onerous 'binding' or 'foreign function interfacing' (FFI) code, mapping between C and some higher-level language implementation, but the resulting maintenance burden is high and the effort involved does not scale across many languages, implementations and codebases. This project will exploit recent work adding run-time type information to native code, which changes the game between high-level language implementations and low-level code, making FFI logic plausibly unnecessary. A proof-of-concept version of such an FFIless system has already been created as a CPython module offering seamless interfacing between Python and C, published at the VMIL workshop in 2019. The PhD will address the challenges of a 'full-blown' approach, addressing one or more of the following issues: performance, garbage collection, support for tools such as debuggers and profilers, and support for additional/multiple languages. The project requires strong systems programming skills.

Visual live programming in scientific computing and similar domains

Supervisor: Stephen Kell

Areas: Systems (SE, programming, autonomous systems, robotics, ...), Visualisation, Data science, Human-centred computing, Computing Applications

(Back to Scholarship Allocated)

Project Description

Currently, working interactively with data means either using a pre-built application offering a fixed interface, which is visual and interactive but offers limited programmability, or using custom workflows built by programming/scripting or command-line wizardry, which are flexible but technically demanding and far less visual and interactive. Computational notebooks like Jupyter are in some senses a third way, being somewhat visual, and have proven approachable by those seeking to learn programming 'on the job'. However, they currently suffer many usability and reproducibility issues, and still present a 'walled garden' environment with poor integration into the surrounding system. This PhD is about ways to combine the interactivity of applications and the flexibility of command lines, possibly by designing a notebook system that works differently than Jupyter et al. We observe that crude operating system (OS) interfaces are the bottleneck to interoperable, visual programming, since they lack a rich data model on which to build visualisation as a system-wide service; this is what leads to smaller-scope walled-garden approaches. Recent work adding run-time type information to native code has shown that such limitations can be overcome without defining an entirely new platform. This project will pursue similar approaches encompassing file data and graphical user interface elements. The objective is to demonstrate a graphical workspace that is highly compatible and interoperable, dealing in files of existing formats, but can support working visually and programmatically via a palette of small, composable, user-tailorable graphical tools. Target audiences include computational scientists, data scientists, digital artists and the like. The project requires systems programming skills and an interest in human--computer interaction topics.

Studying induced demand in software performance

Supervisor: Stephen Kell

Areas: Systems (SE, programming, autonomous systems, robotics, ...), Human-centred computing

(Back to Scholarship Allocated)

Project Description

Tradition has it that computing resources are scarce and therefore that efficiency is a prime concern of programs, especially in infrastructure programs such as compilers. However, in the modern world this is no longer true: computing resources are plentiful and powerful, and software's functionality rarely challenges the limits of the hardware. Despite this, users continue to experience software that is slow, and continue to replace hardware with newer hardware in order to 'keep up' -- especially in the era of continuously updated web-based software. It has long been observed that everyday software is getting slower (e.g. the famous Wirth's law). One theory to explain this is 'induced demand', where greater capacity changes habits of programmers and users in ways that effectively 'soak up' the extra capacity and, often, worsen the apparent infrastructure shortfall. (One classic text on induced demand is Hart & Spivak's "The Elephant in the Bedroom", 1993). This project will study the phenomenon of induced demand in commodity software stacks. It will most likely consist of developing novel profiling tools to study the evolution of the performance of commodity software, and of case studies that pinpoint technical decisions or changes which explain the loss of performance. One possible angle is to study open-source desktop software over the period from the mid-1990s to the present; one tool-building tactic would be to exploit how a single Linux kernel can host user software environments spanning a large interval of time.

Reliability and verification of software for scientific and ML computing

Supervisor: Dr Karine Even-Mendoza, Dr Hana Chockler, Dr Hector Menendez Benito (1st, 2nd and 3rd).

Areas: Machine Learning (ML), Artificial Intelligence (AI), Systems (SE, programming, autonomous systems, robotics, ...), Foundations of computing, Computing Applications

(Back to Scholarship Allocated)

Project Description

Project Description. The long-standing challenge of ensuring the reliability of machine learning implementations (ML) and programming language (PL) libraries, particularly in floating-point and arithmetic computation, has been a focus of attention within programming languages, formal methods, and AI research communities. Historically, researchers have often tended to confine the scope of their research to small-scale problems rather than real-world computer systems. However, modern applications increasingly rely on mathematical computations, such as Alexa voice recognition (using discrete Fourier transform) and autonomous cars. This heightened dependence on accurate computation amplifies concerns about the impact of inaccuracies on system reliability. This project addresses low-level implementations heavily reliant on floating-point computations—a crucial but insufficiently tested area. The precision of these computations significantly influences the reliability of ML and PL libraries and compilers. The project aims to enhance the testing of software systems, specifically focusing on the reliability of software using ML and PL libraries in their algorithms. To achieve this, the student will: - Develop methods to assess the quality of test cases, applying them when attempting to detect miscompilations (silent errors during translation into machine code) and logical bugs in floating-point optimizations and library implementations. - Investigate approaches for testing software libraries and their compilers meaningfully in the context of mathematical and numerical procedures. - Explore fault localization approaches and other techniques to pinpoint detected bugs, ensuring clarity in distinguishing actual bugs from potential issues related to the testing mechanism. The student will employ static and dynamic code analysis, code generation for testing, and testing strategies (like differential testing). After designing a system for meaningful testing of ML and PL libraries, the student will extensively evaluate its bug detection capabilities. The emphasis is on ensuring the identified issues are genuine bugs rather than stemming from the testing methodology. The student will actively engage with the software engineering community, reporting any bugs uncovered during the evaluation process. The above will include investigations of novel ways to design tests and testing campaigns

and deal better with coverage of specific functionalities in the compilers and their PL and ML libraries. Context. Compilers and their software libraries, widely used complex programs, are the bridge between software (written in English-look-alike programming language) and machine code (consisting of 0s and 1s). They give us the means to write complex and sophisticated yet efficient algorithms in healthcare, finance, transportation (and more) using mathematical, ML and AI components, empowering today's engineers and relieving them of conceptual high-level tasks. Consequently, compiler bugs broadly impact software, and library defects affect ML and AI trustworthiness. C standard libraries give us the power to compute values of the sinuous function in just one line, and ML libraries allow us to run reinforcement learning with several lines of code. However, ensuring correct translation is complex, as it involves reasoning about the program code's connection to its machine code translation. One of the most expensive yet neglected errors is associated with the floating-point data types: essential types representing numerical data in software, particularly vital in ML and AI implementations; these, in many cases, led to significant financial losses and jeopardised lives. Yet, testing support is often insufficient, commonly limited to the detection of logical faults in lines of code written by the user or crashes when executing machine code because of the testing mathematical code complexity.

References

[1] K. Even-Mendoza, A. Sharma, A. F. Donaldson, and C. Cadar. 2023. GrayC: Greybox Fuzzing of Compilers and Analysers for C. ISSTA 2023: 1219—1231. https://doi.org/10.1145/3597926.3598130

[2] K. Even-Mendoza, C. Cadar, and A. F. Donaldson, CSMITHEDGE: more effective compiler testing by handling undefined behaviour less conservatively. Empir Software Eng 27, 129 (2022). https://doi.org/10.1007/s10664-022-10146-1.

[3] MLighter is an ongoing project with a webpage: http://mlighter.freedevelop.org

[4] K. Even-Mendoza, A. E. J. Hyvarinen, H. Chockler and N. Sharygina: Lattice-based SMT for Program Verification. MEMOCODE 2019 : 16:1-16:11.

[5] H. Chockler, K. Even, and E. Yahav. Finding rare numerical stability errors in concurrent computations. ISSTA, 2013, pages 12—22. (alphabetic order)

[6] J. M. Zhang, M. Harman, L. Ma and Y. Liu, "Machine Learning Testing: Survey, Landscapes and Horizons", in IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 1-36, 1 Jan. 2022, doi: 10.1109/TSE.2019.2962027.

[7] J. Chen, J. Patra, M. Pradel, Y. Xiong, H. Zhang, D. Hao, and L. Zhang. 2020. A Survey of Compiler Testing. ACM Comput. Surv. 53, 1, Article 4 (January 2021), 36 pages. https://doi.org/10.1145/3363562

[8] X. Yang, Y. Chen, E. Eide, and J. Regehr. 2011. Finding and understanding bugs in C compilers. SIGPLAN Not. 46, 6 (June 2011), 283—294. https://doi.org/10.1145/1993316.1993532

[9] A. F. Donaldson, H. Evrard, and P. Thomson. 2020. Putting randomized compiler testing into production (experience report). In Proceedings of the 34th European Conference on Object-Oriented Programming (ECOOP 2020). Schloss Dagstuhl-Leibniz-Zentrum fur Informatik. https://drops.dagstuhl.de/opus/volltexte/2020/13179/

[10] V. Livinskii, D. Babokin, and J. Regehr. 2020. Random testing for C and C++ compilers with YARPGen. Proc. ACM Program. Lang. 4, OOPSLA, Article 196 (November 2020), 25 pages. https://doi.org/10.1145/3428264

[11] C. Murphy, K. Shen, and G. Kaiser. 2009. Automatic system testing of programs without test oracles. In Proceedings of the eighteenth international symposium on Software Testing and Analysis (ISSTA '09). Association for Computing Machinery, New York, NY, USA, 189—200. https://doi.org/10.1145/1572272.1572295

[12] A Google self-driving car caused a crash for the first time. 2016. https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report

[13] SGD: commonly used during the training process to update the weights of the neural network based on the gradients of the loss function with respect to the weights. https://keras.io/api/optimizers/sgd/

Improving Understandability of Automatically Generated Test Cases using Text-to-Text Transformer Models

Supervisor: Gunel Jahangirova

Areas: Systems (SE, programming, autonomous systems, robotics, ...), Artificial Intelligence (AI), Natural Language Processing (NLP), Machine Learning (ML)

(Back to Scholarship Allocated)

Project Description

The costs associated with software testing activities make their full automation an important research topic. The existing automated test case generation tools (ATGTs) have made significant progress in achieving high coverage, high fault detection rate and input diversity. However, the research in software testing is still far from fulfilling its dream of full automation because multiple studies demonstrate that developers find automatically generated test cases hard to read and understand. This project proposes three directions to tackle the problem of the understandability of automatically generated test cases. The first direction is based on the insight that developer-written test suites capture the information about what testing the given class looks like when performed by the developer and therefore contains features that make the test cases more understandable. We aim to extract the available understandability-related information from developer-written test suites and transfer it into the automatically generated test cases. Our second direction aims to make the understandability of the test case part of the test case generation process such that it favours the test cases with higher understandability. For this, we want to collect a large dataset with human-annotated understandability scores and train a learning model that can predict the understandability score for a candidate test case. The last direction aims to take advantage of the increasing success of text-to-text transformer models. We plan to collect a large dataset of pairs of automatically generated and developer-written test cases that test similar behaviour and train a transformer model that takes an automatically generated test case and transforms it into a version that looks like developer-written. The expected results from the project are in two directions. The first one is the deepened comprehension of the understandability problem. The second one is the set of automated software testing tools that will produce an output that is more understandable by the developers leading to wider adoption of such tools in industrial settings. Moreover, we plan to conduct large studies involving human participants to evaluate the understandability, which will hopefully provide the software engineering research community with examples of well-designed studies evaluating the qualitative properties of test cases.

References

Related Work:

1. E. Daka, J. M. Rojas, and G. Fraser, “Generating unit tests with descriptive names or: Would you name your children thing1 and thing2?” in Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2017, pp. 57–67.
2. G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, “Does automated white-box test generation really help software testers?” in Proceedings of the 2013 International Symposium on Software Testing and Analysis, 2013, pp. 291–301.
3. J. M. Rojas, G. Fraser, and A. Arcuri, “Automated unit test generation during software development: A controlled experiment and think-aloud observations,” in Proceedings of the 2015 international symposium on software testing and analysis, 2015, pp. 338–349.
4. E. Daka, J. Campos, G. Fraser, J. Dorn, and W. Weimer, “Modeling readability to improve unit tests,” in Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015, pp. 107–118.

Estimating the ground truth of LLMs in softare engineering Tasks

Supervisor: Jie M. Zhang

Areas: Artificial Intelligence (AI), Machine Learning (ML), Natural Language Processing (NLP), Systems (SE, programming, autonomous systems, robotics, ...)

(Back to Scholarship Allocated)

Project Description

When using LLMs for software engineering tasks such as code generation, it is important to understand how reliable the generated outputs are. Most of the time the ground truth is unknown. Thus, it is important to estimate the confidence and accuracy of LLMs so as to improve their usability and help users judge whether to adopt the provided solutions. This proposal aims to explore different methods to estimate the confidence of LLMs in generating solutions, in particular to software engineering-related tasks.

References

https://arxiv.org/pdf/2310.03533.pdf
https://openreview.net/forum?id=gjeQKFxFpZ

Scholarship Not Allocated

(Back to Top)

- Requirements formalisation using machine learning
- Trustworthy digital twins and simulations
- Ensuring Trustworthy AI through Verification and Validation in ML Implementations: Compilers and Libraries via Generative Approaches.
- Combining Symbolic and Non-symbolic machine learning for program translation
- Understanding Software Security: Unveiling Vulnerabilities through Binary-based Testing Strategies
- Enhancing Safety in Robotics by Tackling Blind-Spots and Bias in AI Models
- Verification of Autonomous Agents in Uncertain Environments
- Reliable Learning for Safe Autonomy with Conformal Prediction

Requirements formalisation using machine learning

Supervisor: Kevin Lano

Areas: Systems (SE, programming, autonomous systems, robotics, ...), Machine Learning (ML), Natural Language Processing (NLP)

(Back to Scholarship Not Allocated)

Project Description

Formalisation of natural language requirements in software modelling languages such as UML is an essential activity in software development. Various heuristic and machine learning approaches have been applied to this problem over the last 10 years. This research proposal will investigate the application of deep learning approaches and in particular large language models (LLMs) to the formalisation of software requirements.

References

"On the assessment of ChatGPT for modeling tasks"
https://link.springer.com/article/10.1007/s10270-023-01105-5

Trustworthy digital twins and simulations

Supervisor: Steffen Zschaler

Areas: Artificial Intelligence (AI), Systems (SE, programming, autonomous systems, robotics, ...)

(Back to Scholarship Not Allocated)

Project Description

Digital twins are digital representations of real-world entities that are continually updated and can affect changes in the real world. They are used as decision-support systems or even to make decisions autonomously. It is, therefore paramount that we can trust them and have a clear and explicit understanding of when they are applicable or not. However, the scope of validity and rationale underpinning a digital twin is often not explicitly documented or is only documented in natural-language text, making evaluating and maintaining any such arguments difficult. In this project, you will explore how trust in digital twins can be better supported through developing structured, explicitly represented arguments and how these arguments can be semi-automatically validated and maintained over time. I work with stakeholders in a range of domains, providing opportunities for research in different contexts, potentially as separate PhD projects.

Ensuring Trustworthy AI through Verification and Validation in ML Implementations: Compilers and Libraries via Generative Approaches.

Supervisor: Karine Even-Mendoza, Hector Menendez Benito

Areas: Machine Learning (ML), Systems (SE, programming, autonomous systems, robotics, ...), Artificial Intelligence (AI), Computing Applications

(Back to Scholarship Not Allocated)

Project Description

Project Description. The issue of machine learning trust is a pressing concern that has brought together multiple communities to tackle it. With the increasing use of tools such as ChatGPT and the identification of fairness issues, detecting security concerns and ensuring the reliability of machine learning is paramount to its continued development. This project addresses low-level implementation in machine learning, an often-overlooked area, but one that profoundly impacts the reliability of libraries and languages, including TensorFlow, Keras, PyTorch, Python, and R. Knowledge in programming languages and compilers such as CPython and C, as well as familiarity with ML libraries in Python and R, are essential for this project. Project. The project's main idea is to generate and diversify test cases for testing machine learning implementations for each level of abstraction from the top language to the low-level libraries. The student will be employing diverse testing techniques, like LLM for generating test cases, focusing on aspects like numerical validity, security, and fairness to be able to test these aspects more thoroughly, and a "Godel Test" variant: a method that parametrises input generators for programs and controls the parameters to create testing strategies. Among the testing strategies, we will apply multiple test suite generation strategies, such as focused testing (i.e. testing new software's components, which are common in the traditional machine learning libraries), vulnerability unmasking, and differential testing techniques. The student will design a system based on search strategies that will try to guide the algorithms to exhibit the possible branches of the machine learning code and its compilers. For that, we will extend the testing framework of the MLighter tool, a holistic tool for evaluating the security, reliability and performance of machine learning, to deal with these specific problems using generative approaches (including LLM). The student will then extensively evaluate the system's capabilities, focusing on its ability to test deeper parts of the ML code and potentially communicating with the software engineering community to report any exposed vulnerabilities and logical bugs discovered during the evaluation process. The above will include investigations of novel ways to design tests and testing campaigns using LLM and better deal with coverage of specific functionality in the ML code and the test oracle problem. Context. To the best of our

knowledge, while there are a few works related to Python compiler fuzzing (and none for R compiler fuzzing), we have recently seen a substantial volume of research focused on testing ML libraries. With the introduction of LLM and the growing interest in ChatGPT-related research, there is an increased need to expand and enhance testing methodologies in these areas, including a growing emphasis on fuzzing ML libraries. None of these works suggested a holistic way of dealing with the reliability of machine learning libraries with the compilers generating their executable binaries. Considering the potential points of failure: it can occur in any of the following components, or a combination thereof: (1) the Python or R compiler, (2) the ML library implemented in an optimising compiler like C, and (3) the optimising compiler itself (e.g., C). The project consists of two parts: ML libraries testing and the lowest level of testing, which is compiler testing.

References

[1] MLighter is an on-going project with a webpage: http://mlighter.freedevelop.org

[2] H. D. Menendez, "Measuring Machine Learning Robustness in front of Static and Dynamic Adversaries*," 2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI), Macao, China, 2022, pp. 174-181, doi: 10.1109/ICTAI56018.2022.00033.

[3] K. Even-Mendoza, A. Sharma, A. F. Donaldson, and C. Cadar. 2023. GrayC: Greybox Fuzzing of Compilers and Analysers for C. In Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2023). Association for Computing Machinery, New York, NY, USA, 1219—1231. https://doi.org/10.1145/3597926.3598130

[4] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz and S. Yoo, "The Oracle Problem in Software Testing: A Survey," in IEEE Transactions on Software Engineering, vol. 41, no. 5, pp. 507-525, 1 May 2015, doi: 10.1109/TSE.2014.2372785.

[5] A. Wei, Y. Deng, C. Yang, and L. Zhang. 2022. Free lunch for testing: fuzzing deep-learning libraries from open source. In Proceedings of the 44th International Conference on Software Engineering (ICSE '22). Association for Computing Machinery, New York, NY, USA, 995—1007. https://doi.org/10.1145/3510003.3510041

[6] O. Bastani, R. Sharma, A. Aiken, and P. Liang. 2017. Synthesizing program input grammars. SIGPLAN Not. 52, 6 (June 2017), 95—110. https://doi.org/10.1145/3140587.3062349

[7] CompCert: Leroy, X. (2021). The CompCert C verified compiler: Documentation and user's manual (Doctoral dissertation, Inria).

[8] S. Poulding and R. Feldt. 2014. Generating structured test data with specific properties using nested Monte-Carlo search. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14). Association for Computing Machinery, New York, NY, USA, 1279—1286. https://doi.org/10.1145/2576768.2598339

[8] J. Chen, J. Patra, M. Pradel, Y. Xiong, H. Zhang, D. Hao, and L. Zhang. 2020. A Survey of Compiler Testing. ACM Comput. Surv. 53, 1, Article 4 (January 2021), 36 pages. https://doi.org/10.1145/3363562

[9] A. Dakhama, K. Even-Mendoza, W.B. Langdon, H. Menendez, and J. Petke (2023). SearchGEM5: Towards Reliable gem5 with Search Based Software Testing and Large Language Models. Symposium on Search Based Software Engineering (SSBSE). https://tinyurl.com/2u2aeb4r

[10] J. M. Zhang, M. Harman, L. Ma and Y. Liu, "Machine Learning Testing: Survey, Landscapes and Horizons", in IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 1-36, 1 Jan. 2022, doi: 10.1109/TSE.2019.2962027.

[11] The Ken Thompson Heck: https://wiki.c2.com/?TheKenThompsonHack

Combining Symbolic and Non-symbolic machine learning for program translation

Supervisor: Kevin Lano

Areas: Systems (SE, programming, autonomous systems, robotics, ...), Machine Learning (ML)

(Back to Scholarship Not Allocated)

Project Description

Machine learning approaches such as LLMs have been applied to the problem of program translation: translating programs from one language such as Java to another (e.g., Python). These approaches have limited accuracy and reliability. This project will investigate improvements to ML program translation by combining precise translation rules with non-symbolic ML in order to produce more effective translation approaches.

References

Lano, K and Xue, Q., "Code generation by example using symbolic machine learning", SNCS, 2023. https://link.springer.com/article/10.1007/s42979-022-01573-4.

Understanding Software Security: Unveiling Vulnerabilities through Binary-based Testing Strategies

Supervisor: Dr Karine Even-Mendoza, Dr Hector Menendez Benito

Areas: Machine Learning (ML), Foundations of computing, Cybersecurity, Systems (SE, programming, autonomous systems, robotics, ...), Computing Applications

(Back to Scholarship Not Allocated)

Project Description

Software ecosystems rely on the way operating systems distribute resources. By creating the address space, the process space, the threads and the security tokens of the running program, the system provides an execution context that changes depending on the kernel version or even the compiler used to execute the programs. The integration of a program into a systematic environment that evolves depending on kernel and compilation version might not imply security vulnerabilities, but in the presence of crashes, the exploitability of the system will directly depend on how it deals with resources, as obfuscations proved [2]. Under these conditions, there are a few strategies that can provide some light on ways to identify these vulnerabilities. The first one is to employ semantic equivalent transformations to the software and study the behavioural changes in the system. The second is to study the decompilation of the final PE or EFL file and investigate how it changes under different compilation options. The third is to employ various testing strategies, such as differential testing, to analyse how the environment is changing the execution, often tracked through profiling strategies. These three strategies will define the three parts of the thesis. Part 1: Process Resources. The student will work by extending the previous work on the security of obfuscations [2]. The extension will focus on the way the heap and the stack are affected in terms of the address space and the managed resources. With this information, the student will better understand the exploitability of specific parts of the system and work on potential mitigations that can support the system's security. Part 2: Compiler's configurations. Compilers optimise code by adding transformations that reduce the way the process collects and manipulates resources. It is also affected by the scheduler. The student will catalogue the effect of optimisations in software, especially bugs, and how they change their nature and exploitability when the system is more vulnerable. Based on these principles, the student will extrapolate the previous knowledge on exploitability to the compilers' context. Part 3: Testing improvements. Based on the previous discoveries about how the system interacts with processes and how compilers and contexts change this, this last part of the thesis focuses on changing the ways testing is applied in the

context of vulnerabilities with the aim of making it more focused to unmask the risk that the context and the compiler can associate with the execution of the files.

References

[1] A. Dakhama, K. Even-Mendoza, W.B. Langdon, H. Menendez, and J. Petke (2023). SearchGEM5: Towards Reliable gem5 with Search Based Software Testing and Large Language Models. Symposium on Search Based Software Engineering (SSBSE). https://tinyurl.com/2u2aeb4r

[2] H. D. Menendez and G. Suarez-Tangil. 2022. ObfSec: Measuring the security of obfuscations from a testing perspective. Expert Syst. Appl. 210, C (Dec 2022). https://doi.org/10.1016/j.eswa.2022.118298

[3] K. Even-Mendoza, C. Cadar, and A. F. Donaldson. 2022. Csmithedge: more effective compiler testing by handling undefined behaviour less conservatively. Empirical Softw. Engg., 27, 6, (Nov. 2022), 35 pages. doi: 10.1007/s10664-022-10146-1.

[4] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, H. Wang. Large language models for software engineering: A systematic literature review. arXiv preprint arXiv:2308.10620. 2023 Aug 21

[5] V. Le, M. Afshari, and Z. Su. 2014. Compiler validation via equivalence modulo inputs. In PLDI '14. ACM, New York, NY, USA, 216—226. https://doi.org/10.1145/2594291.2594334

[6] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz and S. Yoo, "The Oracle Problem in Software Testing: A Survey," in IEEE Transactions on Software Engineering, vol. 41, no. 5, pp. 507-525, 1 May 2015, doi: 10.1109/TSE.2014.2372785.

[7] X. Yang, Y. Chen, E. Eide, and J. Regehr. 2011. Finding and understanding bugs in c compilers. In Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '11). Association for Computing Machinery, San Jose, California, USA, 283—294. isbn: 9781450306638. doi: 10.1145/199349 8.1993532.

[8] J. Regehr. Finding Bugs in C and C++ Compilers using YARPGen. SIGPLAN. PL Perspectives. https://blog.sigplan.org/2021/01/14/finding-bugs-in-c-and-c-compilers-using-yarpgen/

[9] AFL Michal Zalewski, "Technical "whitepaper" for afl-fuzz," https://lcamtuf.coredump.cx/afl/technical_details.txt

Enhancing Safety in Robotics by Tackling Blind-Spots and Bias in AI Models

Supervisor: Gerard Canal (1st) and Hector Menendez (2nd)

Areas: Machine Learning (ML), Artificial Intelligence (AI), Systems (SE, programming, autonomous systems, robotics, ...)

(Back to Scholarship Not Allocated)

Project Description

The current revolution of artificial intelligence (AI) is becoming more prominent and its potential is still to be unleashed. In the context of robotics, AI can provide support to multiple scenarios, among them, industry, education and healthcare. It is important to know how these systems can work on these contexts but it is imperative that they can treat people respectfully and equally. There are significant efforts in this direction that focus on the context of fairness and explainability. Several AI models normally employed in robotics, such as computer vision models, have been tested to discover that they still contain blind-spots in their detection capabilities, several of them affecting specifically protected groups, such as children or citizens with disabilities. Even if the models are becoming more explainable these days, the consequences of these blind-spots in their explanations and especially the actions of the robots in the real world still requires deeper studies. This is particularly important due to the safety issues that this may impose, which is specially critical in assistive scenarios where a robot helps a user from a vulnerable group perform activities of daily living. This thesis aims to address these issues by: 1) Identifying use cases where the sensitiveness of fairness issues might have a strong repercussion in the behaviour of the robots, with a special emphasis on when this results in unsafe situations for the user recipient of the assistance. This will consist of collecting different examples for the literature that the student can have access and implementing them with the robots that we have available in the department such as the PAL Robotics' TIAGo or models of smart cars. It will also potentially employ digital twins to create a simulation environment for more complex robots. 2) Create strategies to identify blind-spots. Based on the previous work of adversarial machine learning where blind-spots are normally identified as misclassifications or mis-actions that a robot will execute, this part of the thesis will work on identifying and designing adversarial scenarios that will make the system misbehave. The scenario design will consider potential sensory alterations that the robot will face, especially connected with environment conditions. With this information, the thesis will aim to explain the scenario and the specific conditions that led to the misclassification. This will support redesigning the learning process and will serve for standardising benchmark testing conditions. 3) Based on the previous adversarial scenarios and the specific transformations that led the

system to make erroneous decisions, this last part will provide explanations about the system limitations, with an aim to enhance the safety of the system. It will focus on: 1) generalising from the adversarial scenarios to create explanations and 2) inverse the pipeline and create adversarial conditions from specific explanations. These adversarial conditions will be focused on fairness. Besides this last part will put a strong effort on evaluating explanatory systems for robotics under adversarial conditions.

References

Canal, G., Torras, C., & Alenya, G. (2021). Are preferences useful for better assistance? a physically assistive robotics user study. ACM Transactions on Human-Robot Interaction (THRI), 10(4), 1-19.

Vila Abad, M., Canal, G., & Alenya, G. (2018). Towards safety in physically assistive robots: eating assistance. In: Proceedings of the 2018 IROS Workshop on Robots for Assisted Living

Wachowiak, L., Celiktutan, O., Coles, A., & Canal, G. (2023, June). A Survey of Evaluation Methods and Metrics for Explanations in Human—Robot Interaction (HRI). In ICRA2023 Workshop on Explainable Robotics.

Menendez, H. D., Bhattacharya, S., Clark, D., & Barr, E. T. (2019). The arms race: Adversarial search defeats entropy used to detect malware. Expert Systems with Applications, 118, 246-260.

Calleja, A., Martin, A., Menendez, H. D., Tapiador, J., & Clark, D. (2018). Picking on the family: Disrupting android malware triage by forcing misclassification. Expert Systems with Applications, 95, 113-126.

Menendez, H. D. (2022, October). Measuring Machine Learning Robustness in front of Static and Dynamic Adversaries. In 2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 174-181). IEEE.

Verification of Autonomous Agents in Uncertain Environments

Supervisor: Nicola Paoletti

Areas: Machine Learning (ML), Artificial Intelligence (AI), Systems (SE, programming, autonomous systems, robotics, ...)

(Back to Scholarship Not Allocated)

Project Description

With the widespread deployment of autonomous agents, such as autonomous cars and robots and the increasing focus on AI safety, this project aims to investigate the safety of neuro-symbolic agents. The field of neuro-symbolic systems is an exciting area of research that combines the power of machine learning with the rigour of symbolic reasoning. Neural systems have shown great promise in a wide range of applications, from robotics and autonomous systems to natural language processing and decision-making. However, verifying the correctness of these systems remains a significant challenge. While neural networks are excellent at learning patterns in data, they can be difficult to interpret and analyse. On the other hand, symbolic reasoning is highly transparent and understandable, but it can be challenging to scale up to complex non-linear and high-dimensional systems. In this project, we are interested in the analysis of multi-agent neuro-symbolic systems (NSS), which are systems comprising multiple agents interacting with each other and with the environment. The behaviour of such agents is determined by a combination of physical dynamics, such as laws of motion, and machine learning components, which are used, for instance, for perception and control. This kind of systems is relevant in many applications, such as multi-agent (deep) reinforcement learning [1], swarm robotics, and traffic management. We aim to develop verification algorithms for multi-agent NSSs, to provide formal guarantees about the satisfaction of some requirements of interest (reach-avoid, multi-stage tasks, or other kinds of temporal properties). Formal reasoning about these systems is, however, computationally challenging, owing to the presence of (complex) neural network models, multiple agents, uncertain (non-deterministic or probabilistic) environments, and sequential decision-making over multiple time steps. Considerable progress has been made in the verification of one-step reachability for neural networks (i.e., input-output specifications), including probabilistic deep models, using techniques like bound propagation [2,3], constraint solving [4,5], and abstract interpretation [6]. These techniques have been recently extended to the verification of single-agent sequential decision-making [7-9]. However, the multi-agent case remains a largely unexplored research area, with the exception of [10-12]. This project will focus on developing new methods to verify the behaviour of multi-agent NSSs under uncertain environments, where uncertainty can be reasoned about in a probabilistic or non-deterministic fashion. We envision that the solution methods will build on and improve existing verification

techniques for single-agent systems, possibly investigating suitable abstractions for dimensionality reduction as well as the combination with data-driven methods like [13] to obtain probabilistic guarantees in the most complex cases where purely symbolic approaches fail. The research project will contribute to the development of trustworthy and reliable multi-agent systems, which can have a significant impact on many applications. The proposed techniques will be evaluated in standard multi-agent RL benchmarks like [14] and different real-world scenarios coming from the REXASI-PRO EU project [15], which will focus on safe navigation of autonomous wheelchairs in crowded environments for people with reduced mobility.

References

[1] Hernandez-Leal, Pablo, Bilal Kartal, and Matthew E. Taylor. "A survey and critique of multiagent deep reinforcement learning." Autonomous Agents and Multi-Agent Systems 33, no. 6 (2019): 750-797.

[2] Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., ... & Kohli, P. (2018). On the effectiveness of interval bound propagation for training verifiably robust models. arXiv preprint arXiv:1810.12715.

[3] Wicker, Matthew, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. "Probabilistic safety for Bayesian neural networks." In Conference on uncertainty in artificial intelligence, pp. 1198-1207. PMLR, 2020.

[4] Katz, Guy, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah et al. "The marabou framework for verification and analysis of deep neural networks." In CAV 2019, pp. 443-452.

[5] Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., & Misener, R. (2020, April). Efficient verification of relu-based neural networks via dependency analysis. In AAAI Conference on Artificial Intelligence, AAAI.

[6] Singh, G., Gehr, T., Puschel, M., & Vechev, M. (2019). An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages, POPL.

[7] Lopez, Diego Manzanas, et al. "NNV 2.0: the neural network verification tool." International Conference on Computer Aided Verification. Cham: Springer Nature Switzerland, 2023.

[8] Wicker, Matthew, et al. "Probabilistic Reach-Avoid for Bayesian Neural Networks." arXiv preprint arXiv:2310.01951 (2023).

[9] Hosseini, M. & Lomuscio, M., (2023) Bounded and Unbounded Verification of RNN-based Agents in Non-deterministic Environments. In AAMAS 2023.

[10] Akintunde, Michael E., Elena Botoeva, Panagiotis Kouvaros, and Alessio Lomuscio. "Verifying strategic abilities of neural-symbolic multi-agent systems." In Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, vol. 17, no. 1, pp. 22-32. 2020.

[11] Mqirmi, P. E., Belardinelli, F., & Leon, B. G. (2021). An Abstraction-based Method to Check Multi-Agent Deep Reinforcement-Learning Behaviors. In The International Conference on Autonomous Agents and Multiagent Systems, AAMAS.

[12] Yan, Rui, Gabriel Santos, Gethin Norman, David Parker, and Marta Kwiatkowska.

"Strategy synthesis for zero-sum neuro-symbolic concurrent stochastic games." arXiv preprint arXiv:2202.06255 (2022).
[13] Bortolussi, Luca, Francesca Cairoli, and Nicola Paoletti. "Conformal Quantitative Predictive Monitoring of STL Requirements for Stochastic Processes." In 26th ACM International Conference on Hybrid Systems: Computation and Control. 2023.
[14] Mordatch, Igor, and Pieter Abbeel. "Emergence of grounded compositional language in multi-agent populations." In Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1. 2018.
[15] REliable & eXplAinable Swarm Intelligence for People with Reduced mObility (REXASI-PRO), https://cordis.europa.eu/project/id/101070028.

Reliable Learning for Safe Autonomy with Conformal Prediction

Supervisor: Nicola Paoletti

Areas: Artificial Intelligence (AI), Machine Learning (ML), Systems (SE, programming, autonomous systems, robotics, ...)

(Back to Scholarship Not Allocated)

Project Description

For their high expressive power and accuracy, machine learning (ML) models are now found in countless application domains. These include autonomous and cyber-physical systems found in high-risk and safety-critical domains, such as healthcare and automotive. These systems nowadays integrate multiple ML components for e.g., sensing, end-to-end control, predictive monitoring, anomaly detection. Hence, data-driven analysis has become necessary in this context, one where rigourous model-driven techniques like model checking have been the go-to solution for years. In this project you will develop data-driven analysis techniques for autonomous systems based on conformal prediction (CP) [1,2], an increasingly popular approach to provide guarantees on the generalization error of ML models: it can be applied on top of any supervised learning model and it provides so-called prediction regions (instead of single-point predictions) guaranteed to contain the (unknown) ground truth with given probability. Crucially, these coverage guarantees are finite-sample (as opposed to asymptotic) and do not rely on any parametric or distributional assumptions. Our group has a track record of developing CP-based methods for predictive monitoring of autonomous and cyber-physical systems [3-6]. With this project, you will contribute to this endeavour working on challenge problems including off-policy prediction [7,8], data-driven optimization, causal inference [9,10], robust inference under distribution shifts [11,12] and uncertain distributions [13,14]. The proposed techniques will be evaluated in standard relevant benchmarks and different real-world scenarios coming from the REXASI-PRO EU project [15], which focuses on safe navigation of autonomous wheelchairs in crowded environments for people with reduced mobility.

References

[1] Vovk, Vladimir, Alexander Gammerman, and Glenn Shafer. Algorithmic learning in a random world. Vol. 29. New York: Springer, 2005.
[2] Angelopoulos, Anastasios N., and Stephen Bates. "A gentle introduction to conformal prediction and distribution-free uncertainty quantification." arXiv preprint arXiv:2107.07511 (2021).
[3] Cairoli, Francesca, Nicola Paoletti, and Luca Bortolussi. "Conformal quantitative predictive monitoring of STL requirements for stochastic processes." Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control. 2023.

[4] Cairoli, Francesca, Luca Bortolussi, and Nicola Paoletti. "Learning-Based Approaches to Predictive Monitoring with Conformal Statistical Guarantees." International Conference on Runtime Verification. Cham: Springer Nature Switzerland, 2023.
[5] Bortolussi, Luca, et al. "Neural predictive monitoring and a comparison of frequentist and Bayesian approaches." International Journal on Software Tools for Technology Transfer 23.4 (2021): 615-640.
[6] Cairoli, Francesca, Luca Bortolussi, and Nicola Paoletti. "Neural predictive monitoring under partial observability." Runtime Verification: 21st International Conference, RV 2021, Virtual Event, October 11—14, 2021, Proceedings 21. Springer International Publishing, 2021.
[7] Russo, Alessio, Daniele Foffano, and Alexandre Proutiere. "Conformal Off-Policy Evaluation in Markov Decision Processes." 62nd IEEE Conference on Decision and Control, Dec. 13-15, 2023, Singapore. IEEE, 2023.
[8] Taufiq, Muhammad Faaiz, et al. "Conformal off-policy prediction in contextual bandits." Advances in Neural Information Processing Systems 35 (2022): 31512-31524.
[9] Lei, L., & Candes, E. J. (2021). Conformal inference of counterfactuals and individual treatment effects. Journal of the Royal Statistical Society Series B: Statistical Methodology, 83(5), 911-938.
[10] Chernozhukov, V., Wuthrich, K., & Zhu, Y. (2021). An exact and robust conformal inference method for counterfactual and synthetic controls. Journal of the American Statistical Association, 116(536), 1849-1864.
[11] Barber, R. F., Candes, E. J., Ramdas, A., & Tibshirani, R. J. (2023). Conformal prediction beyond exchangeability. The Annals of Statistics, 51(2), 816-845.
[12] Gibbs, Isaac, and Emmanuel Candes. "Adaptive conformal inference under distribution shift." Advances in Neural Information Processing Systems 34 (2021): 1660-1672.
[13] Cauchois, M., Gupta, S., Ali, A., & Duchi, J. C. (2020). Robust validation: Confident predictions even when distributions shift. arXiv preprint arXiv:2008.04267.
[14] Gendler, A., Weng, T. W., Daniel, L., & Romano, Y. (2021, October). Adversarially robust conformal prediction. In International Conference on Learning Representations.
[15] REliable & eXplAinable Swarm Intelligence for People with Reduced mObility (REXASI-PRO), https://rexasi-pro.spindoxlabs.com/.