Verification of Hardware **C**oncurrency via Model **Le**arning (**CLeVer**)

Summary

Our society is increasingly reliant on digital devices that are capable of performing several tasks at the same time: tablets, smartphones, our personal computers, they all contain processors capable of executing multiple instructions concurrently. Bugs in these processors, such as the recently discovered Meltdown and Spectre[1], can seriously compromise the security and safety of their users. As the complexity of these systems increases, there is a pressing need to automate the assessment of their correctness, especially with respect to concurrency-related aspects.

Formal verification provides highly effective techniques to automatically check important properties of systems. It relies on a machine-readable abstraction of the actual system—a *model*. A key strength of formal verification is that, when the model is accurate enough, it is able to find bugs that would be hard to find and reproduce using traditional testing alone. However, these models are usually built by humans, and as such can be error-prone and inaccurate. Moreover, sophisticated concurrent behaviours such as weak-memory concurrency make modelling even more challenging.

> **The overall goal of this project is to develop a verification framework for hardware systems that uses artificial intelligence to automatically build and verify better models. Ultimately, this will lead to lower production costs and more reliable hardware.**

This is particularly important for companies that design hardware units destined to large-scale production, such as ARM: a bug in the design may lead to millions of faulty units being manufactured.

The novel idea behind this project is to rely on the *model learning* paradigm, originally proposed in AI, to automatically build a model of a running system in a *black-box fashion*—from a series of observations of the behaviour of the system. This approach can be very effective in taming complexity and achieving scalability: by treating the whole system, or some of its components, as black-boxes, one can fine-tune the level of detail of the model and focus the analysis on specific features.

In recent years model learning has started to be successfully applied in a variety of academic and industrial contexts. However, the models developed using this approach are all inherently *sequential*. The verification of *concurrent* behaviour in hardware systems is an unexplored, challenging, and potentially rewarding application that this project proposes to explore.

[1] https://www.theguardian.com/technology/2018/jan/04/meltdown-spectre-worst-cpu-bugs-ever-found-affect-computers-intel-processors-security-flaw

Track Record

Academic Investigators

Prof. Alexandra Silva (PI) is Professor of Algebra, Semantics, and Computation at UCL.Her research focuses on the modular development of specification languages and algorithms for models of computations. Large part of her work is developed from the unifying perspective offered by coalgebra, a mathematical framework established in the last decades. Silva's work has been recognised by an ERC Starting Grant in 2015 (∼ €1.3m), the Presburger Award in 2017, which is given to a young scientist for outstanding contributions in theoretical Computer Science, and the Needham Award in 2018, made to a distinguished research contribution in computer science by a UK-based researcher. She has been PI on 3 research projects, granted by the Dutch and Portuguese research councils. Silva has been keynote speaker in many international conferences, including CONCUR and FSCD, and she is editor of Logical Methods in Computer Science, J. Logic and Algebraic Methods, and Mathematical Logic Quarterly. Silva's most recent work concerns the generalisation of model learning algorithms for various models and their use in verification. This line of research is particularly relevant for this project.

Dr. Matteo Sammartino (Researcher co-I) is Research Associate and Teaching Fellow at UCL. His research focuses on formal methods for concurrent systems, and on foundational and applied aspects of model learning. He carried out his PhD and early postdoctoral research at the University of Pisa, under the supervision of Prof. Ugo Montanari—world-leading scientist in formal methods for concurrency. This allowed him to gain significant expertise and research experience in this field. In his PhD thesis he introduced a modelling framework for emerging networking architectures, e.g., *Software Defined Networks*, focusing on challenging aspects such as concurrency and operations on infinite data domains [8, 9, 10]. He then developed novel classes of automata able to succinctly represent the semantics of formalisms for concurrency, such as process calculi and Petri nets [3, 2]. More recently, he has worked on modelling and verification of concurrency in Software Defined Networks via actor models [1]. At UCL he became interested in model learning. His research efforts led to publications at POPL [7] and CSL [11] (see next paragraph for details). In particular, he has been intrigued by the possibility of combining model learning and formal methods for concurrency, which is the key idea behind this project. He has been recently awarded ∼£98k to supervise a PhD studentship on a related project, namely verification of networking systems via model learning, granted by the UK Research Institute in Verified

Trustworthy Software Systems. He is co-organiser of the annual Learning and Automata workshop co-located with LICS in 2018, within the Federated Logic Conference (FLoC).

Relevant Recent Work by the Investigators

Recent work by the PI and the RCo-I aimed at providing a unifying perspective on model learning. These efforts resulted in the *Categorical Automata Learning Framework* (CALF) [11] where learning algorithms for a broad range of automata models can be uniformly designed, proven correct and optimised. CALF also provides foundational connections between learning, testing and state-space minimisation algorithms, which are important in practical scenarios. CALF will be used as foundational infrastructure for this project. Building upon this framework, in [12] they developed general optimisations that improve scalability of learning algorithms. In [7] they developed a learning algorithm for *nominal automata*, which are infinite-state – but finitely-representable – automata capable of expressing operations on data from infinite domains. In [5, 6] the PI et al. solved a long-standing question about completeness of *Concurrent Kleene Algebras* (CKA)– a formalism to reason about concurrent systems in a *truly concurrent* fashion. In particular, they introduced an automaton model for CKA with explicit concurrency, which enables practical verification using CKA. CKA and its models will be a crucial part of this project.

Host Organisation

UCL is consistently ranked in the top 20 universities in the world. In REF 2014 the Computer Science Department was ranked in first place. The *Programming Principles, Logic and Verification Group*, of which the PI and the RCo-I are members, has outstanding connections with cutting-edge industry, including Amazon, Facebook and ARM.

Academic Collaborators

Alastair Donaldson is Reader at Imperial College London, where he leads the Multicore Programming Group. His research group is well known for contributions to automated reasoning, verification and testing in the context of many-core systems. He was recipient of the 2017 Needham Award in recognition of this work. He is also director of GraphicsFuzz, a spin-out company from Imperial based on this line of work, recently acquired by Google. Donaldson will contribute expertise on the analysis and programming of multi-core architectures. He will also provide advice on how to maximise practical impact of the work. His Letter of Support is attached.

Industrial Partner

ARM is the world leading semiconductor IP company, with more than 6,000 employees, and a vast ecosystem of more than 1,000 partners. ARM powers 90% of mobile devices, reaching 70% of the global population. ARM will be involved in a close collaboration for the whole duration of the project, providing case-studies and feedback on project results. **Nathan Chong** (Principal Researcher at ARM) will be our main collaborator and liaison with ARM. He has relevant expertise in verification of multi-core systems [4] and in knowledge transfer from academia to industry. His Letter of Support on behalf of ARM is attached to this proposal.

The Research Team

The investigators are well-placed to lead this project, given their combined expertise and research excellence in model learning, formal methods for concurrency, and verification. The collaborators will contribute complementary expertise on verification of multi-core architectures and on industrial applications. The team will also include:

- **a PDRA**, who will be recruited specifically for the project, with expertise in verification and testing, with a particular focus on hardware systems;
- **a PhD student**: the PI will commit to the project one of the studentships from her start-up package. The student will work on both foundational and applied aspects of model learning.

References

[1] E. Albert, M. Gómez-Zamalloa, A. Rubio, M. Sammartino, and A. Silva. Sdn-actors: Modeling and verification of SDN programs. In *FM*, pages 550–567, 2018.

[2] R. Bruni, U. Montanari, and M. Sammartino. A coalgebraic semantics for causality in petri nets. *J. Log. Algebr. Meth. Program.*, 84(6):853–883, 2015.

[3] R. Bruni, U. Montanari, and M. Sammartino. Revisiting causality, coalgebraically. *Acta Informatica*, 52(1):5–33, 2015.

[4] N. Chong, T. Sorensen, and J. Wickerson. The semantics of transactions and weak memory in x86, power, arm, and C++. In *PLDI*, pages 211–225, 2018.

[5] T. Kappé, P. Brunet, B. Luttik, A. Silva, and F. Zanasi. Brzozowski goes concurrent - A kleene theorem for pomset languages. In *CONCUR*, pages 25:1–25:16, 2017.

[6] T. Kappé, P. Brunet, A. Silva, and F. Zanasi. Concurrent kleene algebra: Free model and completeness. In *ESOP*, pages 856–882, 2018.

[7] J. Moerman, M. Sammartino, A. Silva, B. Klin, and M. Szynwelski. Learning nominal automata. In *POPL*, pages 613–625, 2017.

[8] U. Montanari and M. Sammartino. Network conscious π-calculus: A concurrent semantics. In *MFPS*, volume 286, pages 291–306, 2012.

[9] U. Montanari and M. Sammartino. A network-conscious π-calculus and its coalgebraic semantics. *Theor. Comput. Sci.*, 546:188–224, 2014.

[10] U. Montanari and M. Sammartino. Network-conscious π-calculus - A model of pastry. In *LSFA*, volume 312 of *ENTCS*, pages 3–17, 2015.

[11] G. van Heerdt, M. Sammartino, and A. Silva. CALF: categorical automata learning framework. In *CSL*, pages 29:1–29:24, 2017.

[12] G. van Heerdt, M. Sammartino, and A. Silva. Learning automata with side-effects. *CoRR*, abs/1704.08055, 2017.

Proposed Research and its Context

Digital devices increasingly rely on multi-threaded computation, with sophisticated concurrent behaviour becoming prevalent at any scale. Bugs due to poor design and verification can seriously compromise the safety and security of a growing number of people.

In this project we aim to demonstrate that model learning can be used to achieve effective automated modelling and verification of concurrent behaviour in hardware systems, increasing their reliability and security.

Not only is the project proposed timely, but also *now* is the right moment for this research to happen, because the key ingredients forming the basis of the project have been developed by the investigators to a point that makes this project feasible.

The project proposes a highly novel combination of techniques and expertise that will have a transformative effect on the hardware design process, because it will allow for the automation of time-consuming and safety-critical debugging activities that are often performed manually.

We start by introducing the key topics for this project, and the advances this project will provide. We then set out the overall aims and the specific objectives for the project. Finally, we describe our programme and methodology.

Model Learning

The model learning paradigm was introduced by Dana Angluin in the seminal paper [17]. It is shown that a deterministic finite automaton (DFA) representation of an unknown regular language can be learnt under the assumption of an *oracle* that can answer two kinds of queries:

- *membership queries*, asking whether a certain word belongs to the target language;
- *equivalence queries*, asking whether a certain *hypothesis* automaton accepts the target language. If it does not, the oracle provides a *counterexample* that distinguishes the hypothesis automaton and the target one.

Angluin's L* algorithm operates by incrementally building hypothesis DFAs, based on the oracle answers, and terminates when an equivalence query is answered positively. L* has three key features: *accuracy* – the exact DFA is guaranteed to be learned; *succinctness* – the learned DFA is *minimal*; *efficiency* – the number of needed queries is low (*polynomial* in the model size).

In [45], Peled et al. observed that model learning can be used to learn models of real-world systems. The oracle is an abstraction of the target system: a membership query amounts to observing the output generated by the system in response to the given sequence of inputs. However, exact answers to equivalence queries are usually not available. In the same paper important connections with conformance testing and model-checking were made, which enabled relaxing the assumption of an exact oracle. The integration of learning, testing, and model-checking led to techniques for the automated discovery of bugs, and to a variety of applications in academic and industrial contexts [33, 32, 30, 49, 26, 25]. Vaandrager has recently written a comprehensive overview of the widespread use of model learning in verification [52].

As the application domains become more sophisticated, algorithms capable of learning more expressive models are needed. Algorithms have been provided to learn other types of automata [18, 15, 44, 22, 24, 7, 31], including Mealy machines, I/O automata, and several classes of automata for large alphabets.

Learning models of concurrency for hardware verification – the application domain of this project – is an unexplored research venue. As related work in different contexts, we can mention [34, 48, 41], where models of concurrent business processes are synthesised from static data, such as logs. These approaches are not based on model learning. In software engineering, L* has been extended to to learn collections of automata, modelling a distribute system, from message sequence charts representing the system's requirements [23].

In this project we will advance the state-of-the-art of model learning by providing learning techniques for models of concurrency capturing key features of hardware systems. Not all models are suitable for this purpose. We now introduce the specific class of models we shall use.

Models of Concurrency

Modelling concurrency is a traditional problem in Computer Science, and a number of formalisms have been proposed: Petri nets, process calculi, actor models, and others.

Recently, Hoare et al. in [35] introduced Concurrent Kleene Algebras (CKAs) as a unifying theory of concurrency, generalising existing formalisms [36]. CKAs extend Kleene Algebras (KAs), i.e., the algebras for regular expressions, with a concurrency operator $r_1 \parallel r_2$, meaning that r_1 and r_2 can be executed concurrently. Just like KAs and regular expressions allow reasoning about sequential computations, CKAs allow reasoning about concurrent ones. For instance, in [37] they are applied to weak-memory concurrency.

In [5, 6], the PI at al. introduced a new automaton model of concurrency – *pomset automata* – and showed that CKA expressions correspond to pomset automata much as regular expressions correspond to standard finite automata. This is a crucial step to-

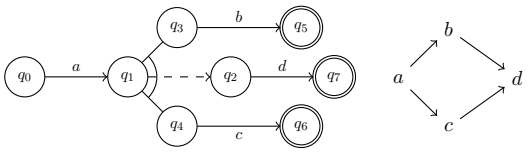

Figure 1: Example of pomset automaton (left) and a pomset in its language (right).

wards the use of CKAs in automated verification.

Pomset automata are automata accepting languages of concurrent executions, represented as particular partial orders called pomsets, where the order enforces sequentiality. An example of pomset automaton is depicted in Figure 1 on the left. In this automaton, one can go from q_1 to q_3 and q_4 *at the same time*. If the final states q_5 and q_6 are reached by these parallel threads, then b and c are read in parallel, the two threads terminate, and the execution resumes from q_2. This mechanism is represented by a special *fork* transition from q_1 to q_2. This automaton accepts the pomset on the right, where first a is read, then b and c concurrently, and then d.

Pomset automata are the model of choice for this project, for which we will develop novel model learning techniques. We expect pomset automata to be particularly advantageous for modelling concurrent hardware behaviour, because of their succinct representation of concurrency. In fact, modelling concurrency via sequential automata – i.e., via *interleaving* – may lead to an exponential state-space blow-up.

The fact that pomset automata are a *natural extension* of ordinary finite automata – and as such they enjoy properties similar to those that made the original L* possible – are strong indicators of the feasibility of the proposed research. Nonetheless, learning them poses a number of challenges, which we plan to address in this project, for instance: *all existing model learning algorithms operate on linear words, whereas pomsets are significantly more complex. How to extend oracles and queries to pomsets?*

Verification of Hardware Concurrency

A key challenge faced by processors vendors such as ARM is to verify that the implementation of memory operations conforms to their specifications. Due to concurrency, implementations are becoming increasingly complex, with such features as speculation and reordering, in the quest for performance, and hence the task of guaranteeing conformance is becoming ever more challenging. It is a crucial task too, because bugs are far from rare: recent work to automate the verification task has led to discoveries of conformance bugs in a future ARM architecture extended with transactional memory [4] and by Lustig et al. [51] of bugs in the RISC-V processor.

In this project we will develop a novel verification framework for concurrent hardware behaviour, which combines learning, testing and model-checking in a coherent whole. The framework will allow verifying the *actual system*, rather than a manually-crafted formal model, which can be inaccurate and error-prone. Also, it will allow the verification to happen *incrementally*, as the model is being learnt, with a potential increase in scalability. Recent work [40] has demonstrated that learning-based approaches can be effective and scalable in safety-critical industrial contexts.

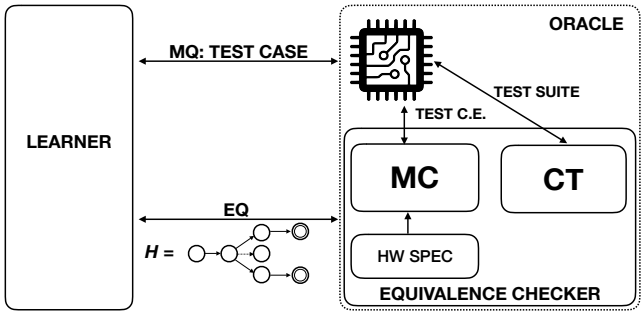

Figure 2: The proposed framework.

The framework we propose is depicted in Figure 2. Recall that, in model learning, the learner incrementally builds an automaton model via queries to an oracle. The oracle will be implemented as an abstraction of the target hardware system: a Membership Query amounts to executing a test case on the system, and Equivalence Queries are implemented via a combination of Model-Checking and Conformance Testing, which will respectively check whether: ❶ the model meets the hardware specification; ❷ the model correctly represents the system's behaviour.

More specifically, the equivalence checker will work as follows. Given a candidate pomset automaton model H, the equivalence checker will first run the model-checker on H to check for ❶. If model-checking succeeds, a conformance test suite will be automatically generated to compare H against the system (point ❷), and the results will be used to refine H. If model-checking fails, generating a counterexample c, a test case will be produced from c and will be executed on the system to see if c corresponds to an actual bug. If so, *a bug in the system has been found*. If not, H is refined.

We note that, due to its black-box nature, our approach is not intended to produce fully detailed models of multi-core architectures, such as the one presented in [46]. In fact, those models require substantial "white-box" knowledge and manual effort. Our approach is somewhat complementary: by treating the whole system, or some of its components, as black boxes, one can fine-tune the level of detail of the learned model and focus the analysis on specific features. This can lead to an increase in scalability of the verification process.

We envisage that this approach will be most beneficial to use at the prototyping stage of the design cycle. In fact, bugs in the prototype logic are harder to find, and debugging is a time-consuming activity at this stage, requiring substantial manual effort (on average, hardware design engineers spend 47% of their time on verification [13]).

We emphasise that our approach will be *architecture-agnostic*, in the sense that will allow for modelling and analysing a variety of concurrent behaviours within the same framework.

Aims and Objectives

The general aims of this project are:

- to develop a novel automated verification framework for hardware concurrency based on model learning;
- to apply it to industrial verification tasks.

In order to achieve our general aims, we set out the following concrete objectives:

O_1: Investigate classes of pomset automata, amenable for learning and verification, capturing two essential features of hardware systems: concurrency and operations on memory.

O_2: Develop efficient techniques and tools to learn the models of **O_1**.

O_3: Develop testing and model-checking techniques for the models of **O_1**, to be integrated with the learning techniques of **O_2** in order to obtain tools that can automatically learn and at the same time verify models of hardware systems.

O_4: Evaluate the effectiveness of our techniques in industrial verification tasks.

Programme and Methodology

The project will be organised into 4 interconnected work packages (see Figure 3). WP1 and WP2 have a more foundational nature, and will investigate learning techniques for increasingly expressive models for concurrency, tackling both **O_1** and **O_2**. These techniques will then be used by WP3 to tackle **O_3**. WP4 (objective **O_4**) will take results of all the other WPs and will apply them in real-world scenarios, and will provide feedback to those WPs.

In order to mitigate risks due to the highly novel nature of this project, our methodology will rely on a gradual approach, where features are added in stages to both models and learning techniques. This is reflected in the tasks into which we organise each WP (see below). Furthermore, research results will be continuously put to the test in practical applications. Feedback from our industrial partner will be used to inform and focus our investigation. Also, our methodology will be *principled*: the *Categorical Automata Learning Framework*, developed by the PI, the RCo-I et al. [11], will be used as theoretical framework to guide the investigation.

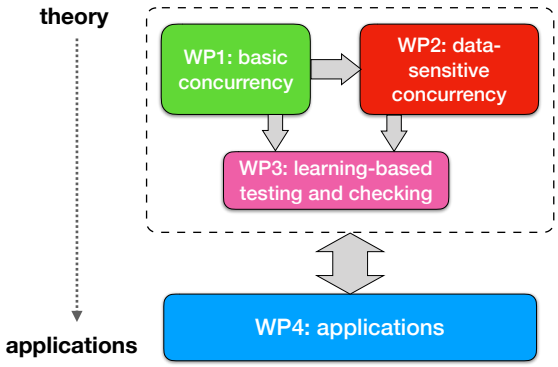

Figure 3: Work packages and their high level interactions.

WP1: learning basic models of concurrency. In this WP we will develop a learning algorithm – extending L* – for the basic pomset automaton model, capable of representing actions happening sequentially or concurrently (see, e.g., Figure 1).

This task is likely to be challenging, and will require identifying learnable sub-classes of those automata. The theoretical underpinnings for this investigation will be provided by the CALF framework, which will be suitably extended with concurrency.

Several optimisations have been proposed for the original L* algorithm, ranging from optimal data structures to state-space reduction techniques [21, 19, 20, 38]. These optimisations are essential to improve scalability in real-world applications. We will develop suitable extensions of these optimisations.

Another intriguing research question is how to optimise the learning process using already available data. For instance, in the context of ARM, logs of executions produced by simulations may already be available. We will study how this information can be incorporated in our learning algorithms in order to learn models in an approximate/incremental way.

Summarising, we will do the following:

T1.1: Develop a learning algorithm for pomset automata.

T1.2: Develop optimisations for this algorithm.

Output: novel, efficient learning algorithm for pomset automata, and prototype implementation.

Roles: The PI will lead on this WP, with major contributions from the RCo-I and from the PhD student. See the the Work Plan document for further details.

WP2: learning data-sensitive models of concurrency. The basic models of concurrency of WP1 have limited expressivity, as they are only able to express concurrent control flow. Modern concurrent systems, such as multi-core processors, require more sophisticated models capable of expressing operations on memory and constraints on these operations. For instance, some architectures allow operations on memory to be re-ordered for optimisation purposes; to verify the correct implementation of these oper-

ations, it is common to specify which outcomes in memory are allowed or forbidden.

In order to capture these aspects, in this WP we will extend the models and techniques of WP1 with:

- **Memory locations**, in the style of *register/nominal automata*. These are automata where each state has finite local memory. Therefore they are suitable to model hardware operating on registers. We will build on the previous work [7] by the PI and the RCo-I, which introduced model learning algorithms for nominal automata.
- **Tests/predicates**. *Kleene Algebras with Tests* (KATs) [42] extend Kleene Algebras with boolean tests. We will develop an analogous extension for pomset automata – capable, e.g, of expressing constraints on the content of memory. We also plan to explore connections with *symbolic automata*, allowing for arbitrary predicates on decidable theories. Their learnability has been recently studied in [31].

As in WP1, the CALF framework will be used to study the theoretical aspects of learning techniques for these models, and will help identifying suitable restrictions whenever needed.

Summarising, we will do the following:

T2.1: Investigate data-sensitive extensions of the models of WP1.

T2.2: Develop learning algorithms for these models.

T2.3: Develop optimisations for these algorithms.

Output: new classes of pomset automata and efficient learning algorithms for them; prototype implementation.

Roles: The RCo-I will lead on this WP. He and the PhD student will make major contributions. The PI will contribute significantly.

WP3: Learning-based testing and model-checking. In this WP we will investigate conformance testing and model-checking techniques for the models of WP1 and WP2, and their integration with learning in order to achieve the framework of Figure 2.

For conformance testing, we will develop techniques to automatically derive test suites from the automata of WP1 and WP2. Being these automata natural extensions of finite automata, the first step will be extending testing techniques commonly used to implement equivalence queries for the basic $L^\star$ algorithm, such as the W-method [28]. The challenge lies in the fact that tests in our setting have a pomset structure. This task will be tackled with the support of the general, foundational connections between model learning and conformance testing established in [11]. We expect structural properties of learned automata, such as minimality, to allow for a better selection of test suites, e.g., minimal ones or ones where tests are ordered in a particular way.

For model-checking, we will investigate techniques to check pomset automata against formal specifications. We will initially explore the use of popular model-checkers, such as SPIN and NuSMV. The limitation of these tools is the representation of concurrency via interleaving, which is inadequate for analysing highly parallel executions. Our ultimate goal for this task is to develop truly concurrent model-checking techniques, allowing a pomset automaton model to be checked against a concurrent specification, in the form of another automaton or of a CKA expression. The first step in this direction is [39], providing a decision procedure for equivalence of pomset automata.

Summarising, we will do the following:

T3.1: Develop conformance testing techniques for the automata of WP1/2.

T3.2: Develop model-checking techniques for the automata of WP1/2.

T3.3: Integrate testing and model-checking with model learning from WP1/2.

Output: novel testing and model-checking techniques; learning-based integrated framework; prototype implementation.

Roles: The PI will lead on this WP, with major contributions from the PDRA, and significant ones from the RCo-I and the PhD student.

WP4: Applications. The purpose of this WP is two-fold: apply results of WP1-3 to industrial case-studies; and provide feedback in order to inform the design choices for those WPs. These activities will be performed in close collaboration with ARM.

We will start off with a preliminary investigation phase. We will identify appropriate verification tasks, and we will familiarise with tools and techniques already used at ARM, which could be used to achieve the integrated framework proposed by this project. In preliminary discussions, promising connections emerged with other black-box verification and testing techniques currently employed by ARM [50, 4, 16, 54].

We will then tailor the models and techniques of the other WPs on the basis of the chosen verification tasks. This will require implementing concrete oracles, capable of interacting with physical hardware (or simulators). In order to do this, we will investigate how to translate pomset-shaped abstract tests into concrete tests, to be run on hardware. The most promising direction is to use *litmus tests* as concrete tests. Litmus tests consist of a concurrent program and constraints on the final memory state. They are closely related to the kind of behaviour and constraints our models aim to express, and have significant tool support on ARM architectures. In particular, we will work on the integration of the tool

suite `diy` by Alglave et al. [16] – for generating and running litmus tests – into our framework. In order to specialise the model-checking component of our framework, we will explore the use of model-checkers and specification languages commonly used in industrial contexts, such as SystemVerilog. *An intriguing research question is whether formal specifications for our model-checking techniques can be automatically derived from natural language specifications*, e.g., reference manuals, which are often translated manually to formal models by engineers. In [47] Verilog code is automatically synthesised from ARM reference manuals. We plan to investigate wether this technique can be adapted to our setting.

Finally, we will experimentally evaluate the effectiveness of our approach, and we will compare our results with existing approaches. Initial evaluation will be conducted using the `herd` tool by Alglave et al. [16], which allows simulating the behaviour of popular multi-core architectures. We will identify metrics for evaluating our techniques. One of the metrics will be the number of bugs detected. Another possible metric is *coverage*, i.e., the percentage of the system's behaviour that is actually tested. In [53] evidence is given that model-learning-based testing can achieve better functional coverage than other testing approaches.

Summarising, we will do the following:

T4.1: Identify suitable verification tasks.
T4.2: Specialise techniques and tools of WP1-3.
T4.3: Conduct experimental evaluation.
Output: specialised techniques and tools; final paper with the overall project results and data.
Roles: The PDRA will lead on and substantially contribute to this WP. The PhD student will make significant contributions. The PI and the RCo-I will both be involved to liaise with WP1-3.

Project Management

The project will be led by the investigators. There will be weekly meetings to discuss research ideas. Additional meetings will be arranged if needed. We will also meet every two months with our industrial partner ARM.

We will establish a **lightweight advisory board** with expertise in model learning and hardware verification. The board will meet every six months to review progress and offer advice. We will seek to involve world-leading scientists in the UK and abroad – e.g., Alastair Donaldson, Prof. Glynn Winskel, Prof. Frits Vaandrager – with whom the investigators have established good professional relationships.

National Importance

Verification is "widely acknowledged as one of the primary bottlenecks in System-on-Chip design" [14], and it is likely to be even more so in the future. In fact, according to the Wilson Research Group Functional Verification Study by Siemens [13], 48% of a typical hardware design project was spent on verification in 2016, 5% more than in 2012. In the same study we find data on the human effort: design engineers spend 47% of their time on verification.

The proposed research aims at providing effective tools – based on AI – to support engineers in their verification efforts, automating time-consuming and safety-critical tasks. Ultimately, it will lead to an increase in productivity and to a reduction in the design costs, helping the development of emerging hardware design companies. This is in perfect alignment with the Artificial Intelligence Grand Challenge set out in the UK Industrial Strategy.

The contribution to the UK economic success can be significant: electronic design is a key UK industry, with more than 150 independent electronic system design houses. Overall, the UK in 2014 had 40% share of Europe's electronic design industry (source: UK Department of International Trade).

Digital devices are nowadays at the heart of our society and our economy. Our project will help produce safer and more secure devices, by reducing the risk of vulnerabilities. Therefore our project falls into the **Digital Economy**, **Engineering** and **Information and Communication Technologies** EPSRC research themes. The project lies at the intersection of a number of EPSRC research areas: **artificial intelligence technologies**, **microelectronic design**, **theoretical Computer Science** and **verification and correctness**. It is also pertains the **pervasive and ubiquitous computing** area (a *grow* area), and the **Trust, identity, privacy and security** and **The internet of things for a service economy** priorities. Our work will also contribute to the **Safe and Secure ICT** Cross-ICT priority.

Academic Impact

Our project lies at at the intersection of several research areas in Computer Science, mainly: formal methods, concurrency theory, artificial intelligence, theoretical computer science, and hardware verification. It will provide significant advances in each of them, stimulating further research, and opportunities for cross-fertilisation.

In particular, this project will push the boundaries of model learning, starting the exploration of a new application domain—verification of hardware concurrency. In order to do so, it will introduce novel formalisms and models for concurrency, and associated verification techniques, which will benefit the communities interested in formal methods for concurrency. New theoretical results will be produced by further developing our framework CALF. This framework will provide researchers with a general method-

ology for designing and optimising new model learning techniques. Finally, researchers in hardware verification will benefit from a novel learning-based approach, which has the the potential to open up new research directions.

Model learning is traditionally connected to other fields, including bio-informatics, computational linguistics and pattern recognition [29]. Our project has the potential to impact them as well, but specific connections would require significant investment in bridging the languages used in the different fields.

References

[13] https://blogs.mentor.com/verificationhorizons/blog/2016/08/08/prologue-the-2016-wilson-research-group-functional-verification-study/.

[14] https://www.arm.com/files/pdf/System_Validation_at_ARM_Enabling_our_partners_to_build_better_systems.pdf.

[15] F. Aarts and F. W. Vaandrager. Learning I/O automata. In *CONCUR*, pages 71–85, 2010.

[16] J. Alglave, L. Maranget, and M. Tautschnig. Herding cats: Modelling, simulation, testing, and data mining for weak memory. *ACM Trans. Program. Lang. Syst.*, 36(2):7:1–7:74, 2014.

[17] D. Angluin. Learning regular sets from queries and counterexamples. *Inform. Comput.*, 75:87–106, 1987.

[18] D. Angluin and M. Csürös. Learning markov chains with variable memory length from noisy output. In *COLT*, pages 298–308, 1997.

[19] D. Angluin, S. Eisenstat, and D. Fisman. Learning regular languages via alternating automata. In *IJCAI*, pages 3308–3314, 2015.

[20] S. Berndt, M. Liśkiewicz, M. Lutter, and R. Reischuk. Learning residual alternating automata. In *AAAI*, pages 1749–1755, 2017.

[21] B. Bollig, P. Habermehl, C. Kern, and M. Leucker. Angluin-style learning of NFA. In *IJCAI*, volume 9, pages 1004–1009, 2009.

[22] B. Bollig, P. Habermehl, M. Leucker, and B. Monmege. A fresh approach to learning register automata. In *DLT*, volume 7907, pages 118–130, 2013.

[23] B. Bollig, J. Katoen, C. Kern, and M. Leucker. Learning communicating automata from mscs. *IEEE Trans. Software Eng.*, 36(3):390–408, 2010.

[24] S. Cassel, F. Howar, B. Jonsson, and B. Steffen. Active learning for extended finite state machines. *FAC*, 28(2):233–263, 2016.

[25] M. Chapman, H. Chockler, P. Kesseli, D. Kroening, O. Strichman, and M. Tautschnig. Learning the language of error. In *ATVA*, volume 9364, pages 114–130. 2015.

[26] C. Y. Cho, D. Babić, E. C. R. Shin, and D. Song. Inference and analysis of formal models of botnet command and control protocols. In *CCS*, pages 426–439. ACM, 2010.

[27] N. Chong, T. Sorensen, and J. Wickerson. The semantics of transactions and weak memory in x86, power, arm, and C++. In *PLDI*, pages 211–225, 2018.

[28] T. S. Chow. Testing software design modeled by finite-state machines. *IEEE Trans. Software Eng.*, 4:178–187, 1978.

[29] C. de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010.

[30] J. de Ruiter and E. Poll. Protocol state fuzzing of TLS implementations. In *USENIX Security*, pages 193–206, 2015.

[31] S. Drews and L. D'Antoni. Learning symbolic automata. In *TACAS*, pages 173–189, 2017.

[32] L. Feng, S. Lundmark, K. Meinke, F. Niu, M. A. Sindhu, and P. Y. H. Wong. Case studies in learning-based testing. In *ICTSS*, pages 164–179, 2013.

[33] A. Groce, D. A. Peled, and M. Yannakakis. Adaptive model checking. *Logic Journal of the IGPL*, 14(5):729–744, 2006.

[34] J. Herbst. A machine learning approach to workflow management. In *ECML*, pages 183–194, 2000.

[35] C. A. R. Hoare, B. Möller, G. Struth, and I. Wehrman. Concurrent kleene algebra. In *CONCUR*, pages 399–414, 2009.

[36] T. Hoare and S. van Staden. The laws of programming unify process calculi. *Sci. Comput. Program.*, 85:102–114, 2014.

[37] A. Horn and D. Kroening. On partial order semantics for sat/smt-based symbolic encodings of weak memory concurrency. In *FORTE*, 2015.

[38] M. Isberner, F. Howar, and B. Steffen. The TTT algorithm: A redundancy-free approach to active automata learning. In *RV*, pages 307–322, 2014.

[39] T. Kappé, P. Brunet, B. Luttik, A. Silva, and F. Zanasi. Equivalence checking for weak bi-kleene algebra. 2018. Submitted for publication.

[40] H. Khosrowjerdi, K. Meinke, and A. Rasmusson. Learning-based testing for safety critical automotive applications. In *IMBSA*, pages 197–211, 2017.

[41] E. Kindler, V. A. Rubin, and W. Schäfer. Process mining and petri net synthesis. In *BPM*, pages 105–116, 2006.

[42] D. Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997.

[43] J. Moerman, M. Sammartino, A. Silva, B. Klin, and M. Szynwelski. Learning nominal automata. In *POPL*, pages 613–625, 2017.

[44] O. Niese. *An integrated approach to testing complex systems*. PhD thesis, Universität Dortmund, 2003.

[45] D. A. Peled, M. Y. Vardi, and M. Yannakakis. Black box checking. *Journal of Automata, Languages and Combinatorics*, 7(2):225–246, 2002.

[46] C. Pulte, S. Flur, W. Deacon, J. French, S. Sarkar, and P. Sewell. Simplifying ARM concurrency: multicopy-atomic axiomatic and operational models for armv8. *PACMPL*, 2(POPL):19:1–19:29, 2018.

[47] A. Reid. Trustworthy specifications of arm® v8-a and v8-m system level architecture. In *FMCAS*, pages 161–168, 2016.

[48] G. Schimm. Mining exact models of concurrent workflows. *Computers in Industry*, 53(3):265–281, 2004.

[49] M. Schuts, J. Hooman, and F. Vaandrager. Refactoring of legacy software using model learning and equivalence checking: an industrial experience report. In *IFM*, volume 9681, pages 311–325, 2016.

[50] D. Stewart, D. Gilday, D. Nevill, and T. Roberts. Processor memory system verification using dogrel: a language for specifying end-to-end properties. In *DIFTS*, 2014.

[51] C. Trippel, Y. A. Manerkar, D. Lustig, M. Pellauer, and M. Martonosi. Tricheck: Memory model verification at the trisection of software, hardware, and ISA. In *ASPLOS*, pages 119–133, 2017.

[52] F. W. Vaandrager. Model learning. *Commun. ACM*, 60(2):86–95, 2017.

[53] N. Walkinshaw, K. Bogdanov, J. Derrick, and J. Paris. Increasing functional coverage by inductive testing: A case study. In *ICTSS*, pages 126–141, 2010.

[54] J. Wickerson, M. Batty, T. Sorensen, and G. A. Constantinides. Automatically comparing memory consistency models. In *POPL*, pages 190–204, 2017.