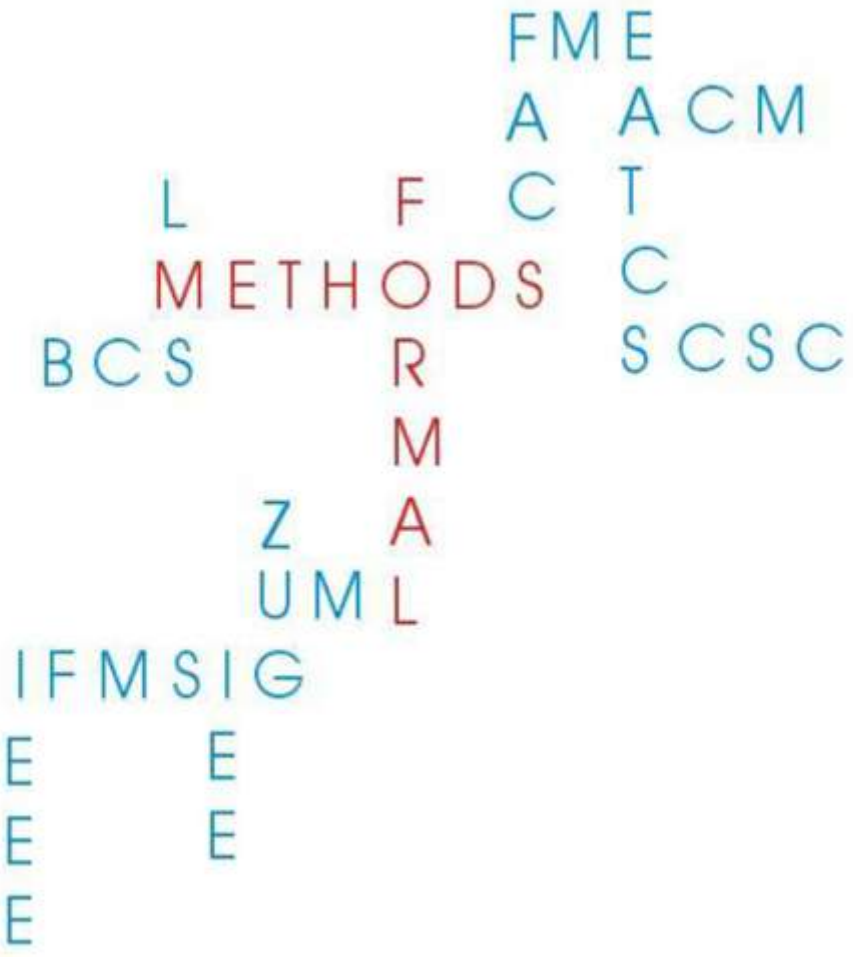
Issue 2026-2
July 2026

FACS FACTS



Formal Aspects of Computing Science Specialist Group

The Newsletter of the Formal Aspects of Computing Science (FACS) Specialist Group

ISSN 0950-1231

About FACS FACTS

FACS FACTS (ISSN: 0950-1231) is the newsletter of the BCS Specialist Group on Formal Aspects of Computing Science (FACS). FACS FACTS is distributed in electronic form to all FACS members.

Submissions to FACS FACTS are always welcome. Please visit the newsletter area of the BCS FACS website for further details at:

https://www.bcs.org/membership/member-communities/facs-formal-aspects-of-computing-science-group/newsletters/

Back issues of FACS FACTS are available for download from:

https://www.bcs.org/membership/member-communities/facs-formal-aspects-of-computing-science-group/newsletters/back-issues-of-facs-facts/

The FACS FACTS Team

Newsletter co-editors:

Tim Denvir **timdenvir@bcs.org**
Brian Monahan **brianqmonahan@googlemail.com**

Editorial team:

Jonathan Bowen, Tim Denvir, Keith Lines, Brian Monahan

Contributors to this issue:

Jonathan Bowen, Tim Denvir, Keith Lines, Li Qin, Zhou Zhilin

BCS-FACS websites:

BCS **https://facs.bcs.org**
LinkedIn **https://www.linkedin.com/groups/2427579/**
Wikipedia **https://en.wikipedia.org/wiki/BCS-FACS**

If you have any questions about BCS-FACS, please send these to Keith Lines at: **klinesfacs@gmail.com**.

View from the Chair

Keith Lines

Carroll Morgan got this year's series of FACS talks off to an excellent start with a presentation on Formal Methods Informally. As Jonathan Bowen notes in his report on the talk, and review of the related book, Carroll has had great success using Hoare logic to help inspire undergraduates to think about code quality. Thank you to Roger Carsley for his great suggestion of approaching Carroll.

I've just had the pleasure of attending the inaugural summer school of the RoboStar centre of excellence in software engineering for robotics, held at the University of York. The attendees covered a wide range of ages, but the majority were, as far as your FACS Chair could tell, post-graduates in the early stages of their PhD research, even from as far as Australia and Brazil. There was also significant presence from industry: about a third of attendees. Also present were the five undergraduate winners of the 2026 Royal Academy of Engineering–Google DeepMind Research Internship (congratulations to them).

Process algebra for refinement (CSP, tock-CSP, Circus, and CyPhyCircus) provides the mathematical basis for the exciting technologies being developed by the RoboStar team. One of the presenters, the leader of the RoboStar branch in France, also noted how tools based on the B-Method are helping ensure the safety of metros worldwide.

In May, Jonathan taught at the Eighth International School and Workshop on Engineering Trustworthy Software Systems (SETSS 2026) at the Southwest University near Chongqing, China. He provides a comprehensive report, including an impressive list of presenters that features names familiar to us all at FACS.

As part of the May workshop, Jonathan gave a presentation on the life and achievements of Tony Hoare, including references to Professor Hoare's family, his many colleagues and the students whose lives and careers he had a key role in shaping. Jonathan also presented this excellent talk at the BCS London offices in early July, soon after returning from Shanghai. The slides for this talk are available on the BCS website. A recording of the talk will be made available on the BCS YouTube channel.

(Photograph by Chris Rees)

Before the talk got underway, I had the pleasure of making a brief presentation to Jonathan on behalf of BCS, thanking him for his ongoing hard work over many years (not least as FACS chair).

Jonathan has indeed had a busy few months, because in early July he also presented at the International Summer School on Trusted Software in Shanghai. The theme of the school was "Formal Verification and Trusted AI". Qin Li and Zhilin Zhou present a report on this school (translated by Google from Chinese and then copy-edited, with permission of the authors).

These prestigious events, with young researchers showing interest and engagement, that underline the role for formal methods in bringing rigour and trust to AI applications, indicate that a bright

future for this topic is there for the taking.

We also enter the archives for a reminder of "proper" computers, with an intriguing short article ostensibly concerning the Manchester Ferranti computer (aka the "Mark 1"). However, the key question considered here concerns how many photographs are there that feature Alan Turing alongside actual computer hardware? The article's conclusions may surprise you!

Tim Denvir provides this issue with some fascinating reflections on his notion of hyper-algebras. Tim presents the ideas in an elegant, conversational style, making the underlying thought processes clear. That is indeed an excellent approach for our (dare I say!) informal newsletter.

With all the interest in matters relating to AI, you may have heard of the Leiden Declaration (**https://leidendeclaration.ai**) by now? The purpose of this declaration is to clearly state what it means for people to conduct responsible AI-related research within a range of primarily STEM subjects, such as mathematics, computer science, and related areas, e.g., robotics. Individuals are invited to *publicly* pledge to conduct their AI-related research in accordance with the Leiden Declaration. We strongly recommend that FACS members give this their fullest consideration!

Sadly, in recent years we have had to say goodbye to great pioneers and practitioners of formal aspects, not least and not only Tony Hoare and Jean-Raymond Abrial. It is important that we celebrate their lives and achievements. To that end, Jonathan has provided a fulsome obituary of Tony Hoare included in this issue. To be clear, what better way is there to achieve that aim than to show how the solid foundations laid down by pioneers such as these are being built upon by new generations of brilliant and enthusiastic researchers?

FACS is finalising details of its programme of events for this year. We'll be presenting joint events with the London Mathematical Society (LMS) and Formal Methods Europe (FME), as well as the annual Peter Landin Semantics Seminar. Details of how to stay up to date are given at the end of this issue. And, of course, please check the BCS Events Calendar for details of BCS events more widely: **https://www.bcs.org/events-calendar/**

And finally, Tim Denvir is stepping down as main editor of *FACS FACTS*, and Brian Monahan is stepping back to largely focus on the technical production of the newsletter. Jonathan Bowen will be the main editor from next issue. We thank Tim (and Brian) for their many years of service, although they will still be around and contributing articles we hope!

Table of Contents

View from the Chair 3

Report from the Inaugural RoboStar Summer School 29th June to 2nd July 2026 6
(Reported by: Keith Lines)

Report on SETSS 2026 School and Workshop, Chongqing, China 11–17 May 2026 11
by Jonathan P. Bowen

The 22nd International Summer School on Trustworthy Software 34
by Qin Li and Zhilin Zhou

The Manchester Ferranti computer and Alan Turing 41
by Jonathan P. Bowen

Hyper-algebras 46
by Tim Denvir

***Formal Methods Informally* by Carroll Morgan A book review and associated talk** 51
(Reviewed by: Jonathan P. Bowen)

An Obituary of Sir Charles Antony Richard Hoare (1934–2026) 59
by Jonathan P. Bowen

Forthcoming Events 66

FACS Committee 67

Report from the Inaugural RoboStar Summer School 29th June to 2nd July 2026

University of York

(**Reported by:** Keith Lines)

Introduction

RoboStar [1] is a centre of excellence in software engineering for robotics, with researchers based at the Universities of York, Sheffield and KCL. It is also an international initiative, including researchers and industrial partners in Brazil, Denmark, France, Germany, and Norway.

FACS committee members Ana Cavalcanti and Alvaro Miyazawa are RoboStar leaders (York, UK). Therefore, your FACS Chair joined around 70 international delegates at the first RoboStar Summer School, held at the University of York during late June / early July. This article is a brief and informal, (yes, informal), summary of proceedings. I'll leave formal matters to RoboStar members for future FACS presentations, which I fully intend to encourage.

There was a wide range of delegates, including early-stage PhD researchers, postdocs, roboticists and safety-critical systems developers. Five undergraduate winners of this year's Royal Academy of Engineering–Google DeepMind Research Internship [2] also attended. Over the following eight weeks under the guidance of RoboStar, they were to design, verify, and assure a robot for nuclear decommissioning.

Term	Description
RoboArch	A textual language for describing a layered robotic architecture. Can be automatically translated to a **RoboChart** model.
RoboCert	A language for expressing properties of **RoboChart** and **RoboSim** models.
RoboChart	A formal, domain-specific visual modelling language for modelling and verification of robotic systems. Based on model-based systems engineering (MBSE) languages such as UML and SysML, it has a formally defined semantics that MBSE languages generally do not have.
RoboScene	A formal domain-specific visual modelling language for modelling and verification of human-robot interaction.
RoboSim	A formal, domain-specific visual modelling language for representing the physical simulation of robotic systems. RoboSim is used purely for simulations, unlike **RoboChart**.
RoboStar	A centre of excellence in software engineering for robotics.
RoboTool	An Eclipse-based development environment that supports **RoboChart**. Downloadable from: `https://robostar.cs.york.ac.uk/robotool/`
RoboWorld	A controlled natural language to capture operational requirements of robots.

Table 1: RoboStar overview

Figure 1: RoboStar Summer School 2026 attendees

Programme

Over four intense but highly rewarding days (there was a lot of material covered), we were initiated into RoboStar, its tools, and some of its applications.

Day 1

Model-based systems engineering (MBSE, [3]) lies at the heart of the RoboStar approach. As was made clear on Day 1, process calculi based on Tony Hoare's pioneering development of CSP, and its variants to deal with hybrid and probabilistic systems, underpin the models. This approach provides mathematical rigour essential for reasoning about robotics and autonomous systems in which safety-criticality is often vital.

I had always thought of MBSE as a quasi-formal method; a considerable improvement on more informal approaches, but lacking the mathematical rigour of formal methods. It was the first time I'd seen such underpinnings applied in this context.

The day began with Ana giving an overview of the RoboStar approach, comparing it with more traditional state machine based development. State machines have an important role in RoboStar, but they are part of a much richer framework. Gradually a diagram was built up showing how the framework is organised into layers: design, simulation, code generation from simulation and deployment. This diagram provided a map to indicate progress over the following days.

And then we were off into **Module 1:** Modelling Control Software: RoboChart. Speakers: Ana Cavalcanti, Ziggy Attala, Simon Foster, Pedro Ribeiro and Fang Yan. In the first module, Ana

presented the core RoboStar notation to model software components and control software as a whole. The approach is diagrammatic, based on statecharts, but distinguished by its coverage of time constructs. RoboTool, which automates the editing, validation, and verification of models, was demonstrated by Pedro and Ziggy. Simon and Fang covered verification.

In just over forty years of attending computer science lectures and presentations, I'd never seen such details covered at such an early stage. FACS members will surely approve.

Module 2: Normative Requirements Engineering for Autonomous Agents. Speaker: Radu Calinescu. Consideration was now given to social, legal, ethical, empathetic and cultural (SLEEC) requirements. There are a lot of standards in development to help address these matters [4], but these standards can be high-level and abstract.

RoboStar provides a SLEEC framework for normative requirements engineering [5]. Formal rules are defined that ensure robotic systems address high-level normative principles such as dignity, autonomy, accountability, justice, and benevolence. These rules can be verified against a RoboChart design. An assistive-care robot and firefighting drone were presented as examples.

Day 2

Module 3: Enhancing Autonomy for Robots. Speakers: Peter Gorm Larsen, Lukas Esterle, James Baxter and Ana Cavalcanti. Here, Lukas and Peter discussed software architectures for autonomous systems, and some impressive industrial applications where a novel architecture operationalised in the RoboStar framework is being used. Ana and James presented that highly automated operationalisation.

Module 4: From RoboChart Designs to RoboSim Simulations to Code Generation. Speakers: Augusto Sampaio and Alvaro Miyazawa. The translation of RoboChart models, for design, to RoboSim models, for simulation, and, later on, into code, was described. The model-based, workflow, i.e., design → simulate → deploy, is a key concept. The RoboSim model must be a refinement of the RoboChart model and assumptions (e.g. there are no deadlocks) and can be used to automatically produce executable code.

Day 3

Module 5: Implementing Models with a Safety Platform. Speaker: Thierry Lecomte representing RoboStar France and CLEARSY [6]. The module began with an overview of safety, software and hardware failures and safety functions and concepts such as safety integrity levels (SIL). The IEC 61508 standard [7] defines SILs to quantify the performance requirements for safety functions, where SIL1 represents the lowest level of risk and SIL4 the highest.

CLEARSY uses tools based on formal methods to develop safety-critical hardware and software that meet the requirements of SIL2 to SIL4. They have developed a safety platform, which includes software tools based on Jean-Raymond Abrial's B-Method. RoboSim can be used to program this platform.

Module 6: RoboSim Physical Models: modelling, simulation and verification of robotic application. Speakers: Alvaro Miyazawa, Simon Foster, and Jim Woodcock. RoboSim was explored

in further detail, describing the relationship between d-models (data models, services provided by the robotic platform, such as sensors and actuators without going into details of the platform itself), p-models (physical models, the physical details of the robotic platform such as structure and dynamic behaviours), and s-models (scenario models, the environment in which the robot operates).

Semantics, both static to verify that a robotic system is physically plausible and dynamic to verify the system's behaviour were described. The variants of CSP described above are used to provide dynamic semantics.

Day 4

Module 7: Capturing Operational Requirements and Testing. Speakers: Ana Cavalcanti, James Baxter, Felix Brüning, and Alvaro Miyazawa. In the first part of this module, RoboWorld was introduced by Ana and demonstrated by James. Next, Felix introduced testing in general, model-based testing, and Verified Systems' tool, RT-Tester. In the final part of the module, Ana returned to discuss model-based testing approaches using RoboChart, and Alvaro considered the integration of RT-Tester with automatically generated tests from RoboChart.

Module 8: Assured Human–Robot Collaboration (HRC): working together safely and efficiently. Speakers: David A. Anisi and Holly Hendry. The first section of the module built on **Module 5**, providing a further example of the use of SILs. An example was presented of safety assurance of an agricultural robot using Probabilistic Model Checking (PMC) [8].

The final section introduced the RoboScene notation for human-computer interaction. There was a practical exercise, using RoboScene to specify interactions in a restaurant between a robot waiter and human customers and kitchen staff.

The Institute for Safe Autonomy

A guided tour of the highly impressive facilities in the nearby Institute for Safe Autonomy (ISA) was included. As noted on its website [9], ISA "works across academic disciplines, and with partners from industry, policy and society, to safely explore how Robotics and Connected Autonomous Systems can benefit people and the planet". Projects undertaken within ISA range from verifying the safety of autonomous vehicles (land, air and sea-based) to using drones and machine learning for monitoring the health of insect populations [10]. There was also a demonstration of 3D-printed robots whose designs were evolved from other robots.

Wrap-up

After the school ended, there was socialising and informal discussions. For me, it had been a thoroughly inspiring return to student days; enhanced by staying on the campus itself.

Developments for future schools could include robots walking or flying around the lecture room (presenting even? Or would that be taking things too far?!). But surely the most important development will be for previous early-career delegates to return as presenters, to speak about their

successful research and its applications. Applications in industry have been discussed as well.

Many thanks to Ana Cavalcanti and Alvaro Miyazawa for their efforts in reviewing and updating this report, for hosting and organising this highly successful inaugural Robostar meeting, and finally to all the excellent presenters, organisers and attendees.

References

[1] RoboStar homepage: **https://robostar.cs.york.ac.uk/**

[2] Google DeepMind Research Ready:
https://raeng.org.uk/programmes-and-prizes/programmes/uk-grants-and-prizes/support-for-research/research-ready/

[3] International Council on Systems Engineering (INCOSE), MBSE Initiative Working Group:
https://www.incose.org/group/mbse-initiative/

[4] AI Standards Hub: **https://aistandardshub.org/**

[5] Ribeiro, P. et al. (2026). The SLEEC Framework for Normative Requirements Engineering. doi:10.1007/978-3-032-26220-2_24

[6] CLEARSY homepage: **https://www.clearsy.com/**

[7] The 61508 Association homepage: **https://61508.org/**

[8] M. Adam et al (2023). Probabilistic Modelling and Safety Assurance of an Agriculture Robot Providing Light-Treatment. doi:10.1109/CASE56687.2023.10260395

[9] Institute for Safe Autonomy homepage: **https://www.york.ac.uk/safe-autonomy/**

[10] Treetop drones and targeted genetics:
https://www.york.ac.uk/news-and-events/news/2026/research/drones-targeted-genetics/

Report on SETSS 2026 School and Workshop, Chongqing, China 11–17 May 2026

Jonathan P. Bowen
London South Bank University, UK &
Southwest University, China

SETSS Overview

The Eighth International School on Engineering Trustworthy Software Systems (SETSS 2026) with an associated workshop took place at the College of Computer and Information Science (CIS), Southwest University (see Figure 2), in Beibei (see Figure 1), a satellite town in the north of Chongqing, China, held during 11–17 May 2026.

Figure 1: A view of Beibei.

From 2014 to 2019, five consecutive annual editions of the International School on Engineering Trustworthy Software Systems (SETSS) took place, with international speakers presenting extended tutorials. These events were popular within the Chinese software engineering community, particularly among young researchers (mainly postgraduates) interested in formal methods for trustworthy software. The 6th edition was suspended due to the COVID pandemic. However, the School resumed in 2024, with the postponed 6th edition taking place during 15–21 May 2024, at Southwest University in Chongqing, China [1, 3]. The 7th edition took place during 17–23 May 2025, at the Institute of Software, Chinese Academy of Sciences (ISCAS) in Beijing, China [2, 4].

The academic programme of SETSS 2026 offered five days of extended tutorial courses (in one to four 90-minute sessions), along with a two-day workshop providing shorter presentations immediately afterwards. The School offered lectures on leading-edge research in methods and tools for computer system engineering, topical talks on the history and trends of computing, and a workshop. SETSS is mainly intended for postgraduate students, but others interested in recent

Figure 2: Jonathan Bowen and the SETSS 2026 banner at Southwest University.

research presented in a didactic style are also welcome. Presenters range from leading pioneers to outstanding younger scholars.

The courses were led by international experts in the field, invited by the Steering Committee of SETSS 2026. The associated workshop provided an opportunity for school participants to exchange research ideas.

In addition to the academic schedule, informal discussions about research, teaching, and potential collaborations were also encouraged. Besides the scientific aspects, participants also had the opportunity to enjoy the culture of Chongqing, including the local very spicy "hot pot" cuisine (see Figure 3) and a nighttime river cruise on the Yangtse in the centre of the city (see Figure 6), surrounded by brightly lit buildings (see Figure 5).

SETSS 2026 School, 11–15 May 2026

A welcome speech was given by Fudu Zhou, Vice President of Southwest University. The SETSS 2026 Spring School was introduced by Zhiming Liu of Southwest University and the lead organizer of the School (see Figure 7). Afterwards, a group photograph (see Figure 8) was taken before the first of the main presentations. The following keynote lectures and extended tutorials were presented by invited speakers at SETSS 2026 (see Figure 10), the latter consisting of one to four 90-minute sessions. The sessions were chaired by Jonathan Bowen, Xinxin Liu, Tianhai Liu, Shmuel Tyszberowicz, and Kim Larsen.

Figure 3: Students at the hot pot evening.

Figure 4: Dinners for speakers and organizers.

Figure 5: Evening river cruise view: Raffles City Chongqing complex of eight buildings in the Yuzhong District, at the junction of the Jialing and Yangtze rivers.

Figure 6: Evening river cruise gathering of old colleagues.
Left: Zhiming Liu, Jonathan Bowen. **Centre:** Zhiming Liu, Kim Larsen, Xinxin Liu. **Right:** Meng Sun, Bernhard Aichernig, Zhiming Liu. JB and ZL were formerly at UNU-IIST (United Nations University International Institute of Software Technology), Macau, and Birmingham City University, UK. They are now colleagues at Southwest University. XL was a PhD student of KL. XL and ZL were fellow MSc students supervised by Zhou Chaochen. BA and ZL were former UNU-IIST faculty staff, and MS was their former student.

Figure 7: Introduction to the SETSS 2026 Spring School.**Left:** Zhiming Liu. **Right:** Fudu Zhou.

Figure 8: The SETSS 2026 group photograph.
Front row: Jie Liu, Peng Yi, Zhiming Liu, Einar Johnsen, Jonathan Bowen, Fudu Zhou, Kim Larsen, Shmuel Tyszberowicz, Emily Yu, Xinxin Liu, Bo Liu.

Figure 9: The audience at the SETSS 2026 Spring School.
Front row: Kim Larsen, Shmuel Tyszberowicz, Jonathan Bowen, Emily Yu.

Moshe Y. Vardi, Rice University, United States

Are AI minds genuine minds?

(Keynote talk.)

Abstract: The question "Are AI minds genuine minds?" invites us to examine the nature of mind itself and whether artificial intelligence meets its defining criteria. A genuine mind is typically associated with consciousness, self-awareness, intentionality, and the capacity to experience mental states such as emotions. Whether AI qualifies as possessing a true mind ultimately depends on how we define the essential qualities of consciousness and intelligence. While this question has already been raised in the 19th century, recent progress in AI requires us to re-examine it deeply.

Figure 11: Zoom talk by Moshe Vardi.

Biography: Moshe Y. Vardi is University Professor and the George Distinguished Service Professor in Computational Engineering at Rice University. His research focuses on the interface of mathematical logic and computation – including database theory, hardware/software design and verification, multi-agent systems, and constraint

Figure 10: Tutorial presenters: Einar Johnsen, Kim Larsen, Emily Yu, Wei Dong, Maiomiao Zhang, Bernhard Aichernig.

satisfaction. He is the recipient of numerous awards, including the ACM SIGACT Goedel Prize, the ACM Kanellakis Award, the ACM SIGMOD Codd Award, the Knuth Prize, the IEEE Computer Society Goode Award, and the EATCS Distinguished Achievements Award. He is the author and co-author of over 800 papers, as well as two books. He is a Guggenheim Fellow as well as fellow of several societies, and a member of several academies, including the US National Academy of Engineering, National Academy of Science, the American Academy of Arts and Science, and the Royal Society of London. He holds ten honorary titles. He is a Senior Editor of the *Communications of the ACM*, the premier publication in computing.

Einar Broch Johnsen, University of Oslo, Norway

Digital Twins: Connecting Models with The Real World

(Four sessions.)

Figure 12: Introduction by Einar Johnsen.

Abstract: Digital twins are model-centric systems able to perform online analysis. These systems have become an important backbone for modern industry. The purpose of the digital twin is typically to help a target system meet requirements in an environment that it does not fully understand or control. The digital twin mirrors its target system in real time by integrating live observations of the target system, such as sensor data, into its knowledge. This allows the twin to assess the precision of its models, to adjust the models to make them more precise, and possibly to replace models or requirements on the fly to adapt to changes in the twinned system. Digital twins can be used for different kinds of analysis: descriptive analysis aims at explaining incidents that have happened, such as safety violations, predictive analysis aims at explaining what we expect will happen in the near future, thereby enabling a feedback loop to make adjustments to the target system, while prescriptive analysis explores hypothetical "what-if" scenarios for longer-term decision making. In this lecture, we introduced central concepts of digital twins, discussed simple examples, and explored the idea of digital twins from a formal methods perspective, including how to design digital twins as self-adaptive model management systems. We discussed how digital twins can be used to enhance the trustworthiness of software systems, and measures to enhance the trustworthiness of the digital twin itself.

Biography: Einar Broch Johnsen is a Professor and head of the research group on Reliable Systems at the Department of Informatics, University of Oslo. His research interests include programming models, semantics and methodology; program specification and modelling; formal methods and associated theory; model-based analysis, testing, and formal logic. With digital

twins, his research interests extend from model-based analysis to model-centric systems; his work on digital twins spans from self-adaptation and sensor-driven model management to programming abstractions and architectures. He is also interested in the socio-technological aspects of digital twins and digital twins of natural systems, including marine ecosystems and pandemic prevention. He is one of the main developers of ABS, a modelling language for asynchronous distributed systems, and SMOL, a formally defined programming language for digital twins.

Kim G. Larsen, Aalborg University, Denmark

Model Checking, Monitoring, Performance Analysis, Synthesis and Learning for Cyber-Physical Systems

(Four sessions.)

Abstract: Timed automata and priced timed automata have emerged as useful formalisms for modelling real-time and energy-aware systems as found in several embedded and cyber-physical systems. During the last 20 years, the real-time model checker UPPAAL has developed a number of methods allowing for efficient verification of hard timing constraints of timed automata. Moreover a number of significant branches exists, e.g., UPPAAL CORA, providing efficient support for optimization, Also the branch UPPAAL SMC, provides a highly scalable new engine supporting (parallel and distributed) statistical model checking of stochastic hybrid automata with significant applications to performance of energy-aware sensor networks as well as evaluation of lock-down measures in Denmark under COVID.

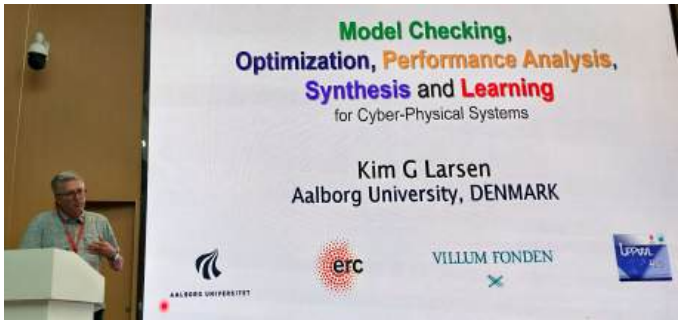


Figure 13: Introduction by Kim Larsen.

More recently the new branch UPPAAL Stratego offers a neuro-symbolic approach to achieve safe and near-optimal decision tree control strategies. The tool combines a dynamic partition-refinement Q-learning algorithm with symbolic methods for synthesizing safety shields that ensures correctness by design. To make synthesis of shields tractable, UPPAAL Stratego are using various abstraction and state-space transformation techniques. We demonstrated superiority of applying the shield before learning (pre-shielding) compared to after (post-shielding). In addition trade-offs between efficiency of strategy representation and degree of optimality subject to safety constraints were discussed, as well as successful on-going applications (water-management, heating systems, and traffic control, swarm robotics). Most recently, methods for shielding and reinforcement learning has been extended to multi-agent systems.

Finally, the lecture covered a recent series of work on developing efficient methods for on-line monitoring the conformance of a real-time systems with respect to logical specifications (e.g.,

MITL). Most of the methods and techniques presented are to be found in the recently released UPPAAL 5.0.

Biography: Kim Guldstrand Larsen is Professor in Computer Science at Aalborg University, Denmark. His field of research includes modelling, validation and verification, performance analysis, and synthesing of real-time, embedded, and cyber-physical systems utilizing and contributing to concurrency theory, model checking and model checking. Kim Guldstrand Larsen is co-founder and main contributor to the tool UPPAAL (`https://uppaal.org`). UPPAAL received the prestigious CAV Award in 2013 as the foremost tool for modelling and verification of real-time systems. Kim Guldstrand Larsen won the ERC Advanced Grant in 2015 and won a Villum Investigator grant in 2021. In 2022 he received the CONCUR Test-of-Time Award 2022.

Kim Guldstrand Larsen is a member of Royal Danish Academy of Sciences and Letters, elected fellow and digital expert (vismand) in the Danish Academy of Technical Sciences and Knight of the Order of the Dannebrog (2007). Moreover he is Honorary Doctor, Uppsala University (1999), Honorary Doctor, École normale supérieure Paris-Saclay, Paris (2007), Foreign Expert of China, Distinguished Professor, Northeastern University (2018). He has published more than 500 peer-reviewed papers and received Thomson Scientific Award as the most cited Danish computer scientist 1990–2004.

Emily Yu, Leiden University, Netherlands

Trustworthy Systems through Automated Reasoning, Certificates, and Runtime Monitoring

(Two sessions.)

Abstract: This series of lectures explored how formal methods can be used to ensure the safety and reliability of modern complex systems, including systems that incorporate machine learning components. Automated reasoning plays a central role in the formal verification of hardware, software, and increasingly AI-based systems. As verification tools become more complex, certification techniques have been developed to provide an additional layer of trust in their results by producing independently checkable evidence of correctness. Firstly, the lectures covered static verification using automated reasoning such as SAT/SMT solving technologies, where certificates serve as mathematical proofs that allow verification results to be independently validated. Secondly, the lectures explored how reinforcement learning can be combined with formal methods to synthesize controllers and neural certificates for systems that are difficult to ad-

Figure 14: Zhiming Liu thanking Emily Yu for her presentation.

Figure 15: Emily Lu in discussion with students.

dress with classical control techniques, while still providing formal guarantees with respect to temporal specifications. Thirdly, the lectures covered runtime monitoring and enforcement techniques, which provide runtime assurance as a complementary approach to static verification by detecting or preventing violations during system execution thereby enabling safe deployment of AI technologies in safety-critical applications.

Biography: Emily Yu is an Assistant Professor in the Leiden Institute of Advanced Computer Science at Leiden University. Her research focuses on system verification and trustworthy AI, with a focus on advancing formal methods as first-order design principles for system correctness and security. She joined Leiden University in October 2025. Prior to that, she was a postdoctoral researcher at the Institute of Science and Technology Austria in Thomas Henzinger's group (2023–2025). She obtained her PhD in Computer Science in 2023 at Johannes Kepler University in Austria, supervised by Armin Biere. She got her bachelor's and master's degree at Imperial College London. Emily Yu's research has been recognized with an ESPRIT fellowship from the Austrian Science Fund.

Wei Dong, National University of Defense Technology, China

Formal and intelligent synthesis for high confidence HCPS software

(Two sessions.)

Abstract: Formal synthesis has always been an important research direction in the field of automatic software generation, yet improving synthesis efficiency remains a major challenge. In recent years, LLM-based code generation has been widely accepted, but how to generate high-quality code has drawn significant attention. For high-confidence Human-Cyber-Physical System (HCPS), its software generation must consider the correctness and safety of interactions between various entities and the external environment, and also needs to improve the quality of the generated code. This course introduced basic concepts of reactive synthesis, and our recent work

on scaling up synthesis and improving efficiency, as well as LLM-based approaches for high-assurance code generation. We considered both formal and intelligent program synthesis to enhance the efficiency of automated development of HCPS software.

Figure 16: Introduction by Wei Dong.

Biography: Wei Dong is the professor in College of Computer Science and Technology, National University of Defense Technology. His research interests include program analysis and verification, program synthesis, and runtime verification. He has authored or coauthored more than 80 papers in conferences and journals, which include FM, OOPSLA, ICSE, FSE, TSE, TOSEM, etc. He has served on more than 20 program committees, and as the Program Co-Chair of several conferences and workshops. He is the vice chair of Technical Committee on Formal Methods of China Computer Federation (CCF TCFM).

Miaomiao Zhang, Tongji University, China

Learning and Verifying Timed Systems

(One session.)

Abstract: Timed systems are widely deployed in industrial systems, especially in safety-critical domains. Formal verification is an important technique for ensuring the reliability of such systems, where model construction is crucial for subsequent analysis. Model learning can infer formal models from system behaviors and has been widely applied in areas such as protocol verification and embedded software analysis. This course presented related work on the modelling and verification of such systems.

Figure 17: Miaomiao Zhang and her PhD student during her presentation.

Biography: Miaomiao Zhang is a Professor at the School of Computer Science and Technology, Tongji University. She received her Ph.D. in Automation from Shanghai Jiao Tong University in 2001, and worked as a postdoc at the Department of Software Science, Radboud University (the Netherlands). Her research focuses on the modelling and verification of timed systems, with publications in CAV, RTSS, FSE, OOPSLA, TACAS, and ACM TECS, among others.

Joseph Sifakis, Verimag & EPFL, France

Bringing AI to Autonomous Systems

(Keynote talk.)

Abstract: Autonomous systems are distributed systems composed of agents, each pursuing its own goals, but which must coordinate to satisfy the overall goals of the system.

The following main points were covered:

1. We analyzed the characteristics of autonomous systems, explaining that they underlie a multifaceted concept of intelligence that cannot be characterized by conversational behavioral tests such as the Turing test.

2. We proposed a development method based on an agent reference architecture that characterizes autonomous behavior as the result of the composition of a set of independent functions. The behavior results from the orchestration of reactive behavior producing actions in response to external stimuli, and proactive behavior aimed at satisfying the agent's needs relating to the success of its mission. The two behaviors coordinate by sharing knowledge contained in a long-term memory.

3. We analyzed how AI can contribute to the creation of AI agents and multi-agent systems, highlighting the need for its seamless integration with traditional software and discussing the current limitations of the state of the art. These limitations particularly concern the use of knowledge stored in long-term memory to semantically control and further improve the accuracy of AI components and adaptation to a constantly changing environment through goal management and planning.

The talk concluded by emphasizing that AI is still in its infancy, and that there is a long way to go to realize the vision of autonomous systems and get as close as possible to human intelligence.

Biography: Professor Joseph Sifakis is Emeritus Research Director at Verimag. He has been a full professor at Ecole Polytechnique Fédérale de Lausanne (EPFL) for the period 2011–2016. He is the founder of the Verimag laboratory in Grenoble, a leading laboratory in the area of safety critical systems that he directed for 13 years.

Joseph Sifakis has made significant and internationally recognized contributions to the design of trustworthy systems in many application areas, including avionics and space systems, telecommunications, and production systems. His current research focuses on autonomous systems, in particular self-driving cars and autonomous telecommunication systems.

In 2007, he received the Turing Award, recognized as the "highest distinction in computer science", for his contribution to the theory and application of model checking, the most widely used system verification technique.

Figure 18: Kim Larsen chairing Joseph Sifakis during his Zoom talk.

Joseph Sifakis is a member of the French Academy of Sciences, the French National Academy of Engineering, Academia Europea, the American Academy of Arts and Sciences, the National Academy of Engineering, the National Academy of Sciences, and the Chinese Academy of Sciences.

Joseph Sifakis is a frequent speaker at international scientific, technical and public forums. He is the author of the book *Understanding and Changing the World*, published in English and Chinese.

DI Dr. Bernhard Aichernig, Johannes Kepler University Linz, Austria

Automata Learning and Testing with AALpy

(Four sessions.)

Figure 19: Bernhard Aichernig and his "Triptych of Formal Methods".

Abstract: This course introduced the fundamentals of automata learning and its close connection to black-box testing, using the open-source tool AALpy[1]. The presented learning-based testing approach combines the active inference of finite-state models with systematic model-based test-case generation.

Participants learned how AALpy can be used to infer and validate deterministic, non-deterministic, and stochastic models. After demonstrating the automated testing of Python classes, more advanced applications were discussed, including protocol fuzzing (e.g., MQTT, Bluetooth) and the extraction of interpretable models from recurrent neural networks.

[1] https://github.com/DES-Lab/AALpy

In addition to active learning algorithms, the course also covered passive learning algorithms that have recently been added to AALpy.

Biography: Bernhard K. Aichernig is a full professor of Formal Methods at Johannes Kepler University Linz, where he leads the Institute of Formal Models and Verification. His research focuses on the foundations of software engineering for dependable and trustworthy systems, with interests in automated falsification, verification, and modelling. Current topics include automata learning, learning-based testing, and the integration of symbolic and subsymbolic AI. He is the author of more than 150 scientific publications. Until April 2025, he was affiliated with Graz University of Technology. From 2002 to 2006, he held a faculty position at the United Nations University in Macao, China. He served on the board of Formal Methods Europe from 2004 to 2016. Prof. Aichernig holds a habilitation in Practical Computer Science and Formal Methods, a doctorate, and a Diplom-Ingenieur degree, all from Graz University of Technology.

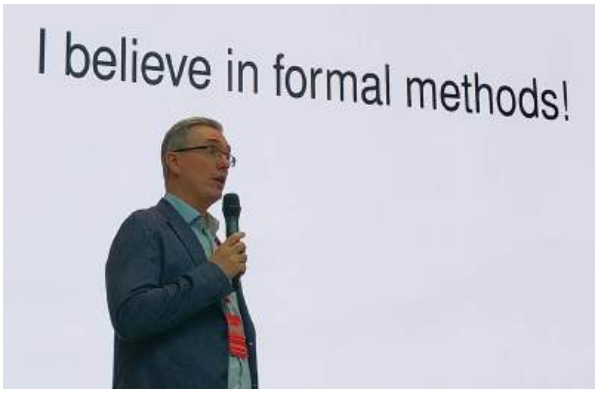


Figure 20: Bernhard Aichernig and his belief in formal methods!

Figure 21: Jonathan Bowen, Einar Johnsen, and Kim Larsen in discussion during a break.

SETSS 2026 Workshop, 16–17 May 2026

The Workshop was organized by Zhenbang Chen of the National University of Defense Technology, China. It was chaired by Tianhai Liu, Xinxin Liu, and Jonathan Bowen. There were eight workshop speakers, mostly from China, who delivered one-hour presentations over a day and a half (five on the first day, three on the second day).

Jonathan Bowen, London South Bank University, UK & Southwest University, China

Tony Hoare: A Life of Logic, Theory, and Practice

Abstract: Prof. Sir Charles Antony Richard (Tony) Hoare (1934–2026) was a giant of computer science, whose work bridged the gap between the abstract elegance of mathematical logic and the practical necessity of reliable software. Best known for the Quicksort algorithm, the development of Hoare logic, and the formalization of Communicating Sequential Processes (CSP), his career was defined by a relentless pursuit of "correctness by construction". This talk chronicled his journey from a student of the Classics at Oxford to the recipient of the ACM A.M. Turing Award, exploring a legacy that helped to transform programming from a craft into a disciplined science.

Figure 22: Jonathan Bowen and Zhiming Liu with busts of Alan Turing and John von Neumann at Southwest University, together with a possible bust of Tony Hoare, in the background.

Biography: Prof. Jonathan P. Bowen, MA Oxon, FBCS, FRSA, is an Emeritus Professor of Computing at London South Bank University, UK, Chair of Museophile Limited, a museum and IT consultancy company that he founded in 2002, and an Adjunct Professor at Southwest University, China. He has been a visiting scholar/professor at a variety of institutions, including the Israel Institute for Advanced Studies (Jerusalem), King's College London, and the Pratt Institute (New York). Previously, he has held academic and research posts at the University of Reading, Oxford University,and Imperial College London. Jonathan's research interests range from computer science, especially software engineering and formal methods, to the history of computing and digital culture. He contributes to Wikipedia on computing-related and cultural topics. His books include *The Turing Guide* (Oxford University Press, 2017) on Alan Turing.

Liqian Chen, National University of Defense Technology, China

Making Template Polyhedral Domain Practical for Real-World Programs

Figure 23: Liqian Chen's presentation.

Abstract: Abstract interpretation provides a generic framework for design static analysis, and has been widely applied in analysis of real-world programs. The scalability and precision of these analyses depend largely on the chosen abstract domain. The template polyhedral domain is well-known for offering a compelling trade-off between scalability and precision. However, designing effective templates currently requires intensive manual effort and domain expertise. This talk presents an approach to automatically generate linear templates for the template polyhedral domain. Experimental results demonstrate that our method can generate highly relevant templates for a variety of standard benchmark sets and complex real-world programs.

Biography: Liqian Chen is a full professor in the College of Computer Science and Technology at the National University of Defense Technology (NUDT), China. His research focuses on program analysis, neural network verification, automated program repair. He has published in venues such as TOSEM, TSE, POPL, OOPSLA, CAV, FSE, ICSE, ASE, ISSTA, and CCS. He has received twice the ACM SIGSOFT Distinguished Paper Award.

He Jiang, Dalian University of Technology, China

Trustworthiness Assurance and Intelligence of Compilers

Figure 24: He Jiang's presentation.

Abstract: Compilers serve as the fundamental infrastructure of the software and information industry. With the evolution of technology, conventional compilers such as GCC and LLVM can no longer meet the diverse emerging demands for trustworthiness, intelligence and other requirements in new application scenarios including aviation, aerospace and robotics. This report first elaborates on the concept of compiler trustworthiness for lightweight compilers and introduces mainstream trustworthy compilers. On this basis, it presents compiler testing methods designed to improve the quality of large-scale compilers. Finally, the report introduces a novel intelligent compiler developed by the research group, which is tailored for autonomous unmanned scenarios.

Biography: He Jiang is the professor and doctoral supervisor at Dalian University of Technology. His current research mainly focuses on intelligent systems and applications, with research achievements widely applied in aerospace and other fields. He has presided over the Key Projects of the National Natural Science Foundation of China, the National Key R&D Program, as well as research projects commissioned by Huawei and aerospace institutions. He has published more than 100 papers in top journals including ACM/IEEE Transactions, as well as prestigious international conferences such as ICSE and FSE. His academic honors include the Citi Teaching Fellowship and the ACM SIGSOFT Distinguished Paper Award (2018, 2023, 2025).

Tianhai Liu, Karlsruhe Institute of Technology (KIT), Germany

Semantic Consistency in Model-Driven Development

Figure 25: Tianhai Liu's presentation.

Abstract: Model-driven development involves multiple heterogeneous artefacts, such as requirements, models, and code, as well as a variety of analyses, including testing, simulation, and formal verification. However, analyses are typically treated as secondary processes, and consistency is often checked only at a syntactic level, making it difficult to ensure meaningful system-level correctness as systems evolve. This talk presents a unified perspective that addresses both challenges. We first introduce the idea of analyses as first-class citizens, explicitly specified by their inputs, assumptions, outputs, and the properties they ensure. This enables modularisation, explicit dependency management, and efficient incremental re-execution when artefacts change. Building on this foundation, we then introduce an observable-centric approach to semantic consistency checking. Observables serve as shared, cross-domain properties that link requirements and models, allowing consistency constraints to be expressed and verified at a semantic level. By integrating these two perspectives, we obtain a framework in which analyses are both structurally explicit and semantically grounded.

Biography: Tianhai Liu is a Postdoctoral Researcher at the Institute for Information Security and Dependability (KASTEL), Karlsruhe Institute of Technology (KIT), Germany. He is also involved in industrial and European horizon research projects (e.g., COSMOS and TRISTAN), where he serves as a project lead on trustworthy and safety-critical software (including ML for cyber-physical systems) and hardware systems (e.g., RISC-V). His research focuses on formal methods and model-driven development for cyber-physical systems, with particular emphasis on consistency management, semantic integration, and incremental verification. He has published in venues such as ICST, FASE, HVC, SETTA, and ENASE. Before joining KIT, he worked at IBM and Technische Universität Darmstadt. He received his PhD from KIT, Master's degree from Technische Universität Dresden, and Bachelor's degree from Chongqing University.

Meng Sun, Peking University, China

Automata-Based Steering of Large Language Models for Diverse Structured Generation

Figure 26: Meng Sun's presentation.

Abstract: Large language models (LLMs) are increasingly tasked with generating structured outputs. While structured generation methods ensure validity, they often lack output diversity, a critical limitation that we confirm in our preliminary study. We propose a novel method to enhance diversity in automaton-based structured generation. Our approach utilizes automata traversal history to steer LLMs towards novel structural patterns. Evaluations show our method significantly improves structural and content diversity while maintaining comparable generation efficiency. Furthermore, we conduct a case study showcasing the effectiveness of our method in generating diverse test cases for testing open-source libraries.

Biography: Meng Sun received his BS degree in Information Science and PhD degree in applied mathematics from Peking University in 1999 and 2005, respectively. He worked as a postdoc researcher at National University of Singapore from 2005 to 2006, and as a scientific staff member at CWI, the Netherlands, from 2006 to 2010. He joined Peking University in 2010 and currently is a full professor at School of Mathematical Sciences in Peking University. His research interests mainly lie in theories of programming, software formal methods, cyber-physical systems, trustworthiness guarantee of AI systems, blockchain and smart contracts. His publications appear in top-tier venues including TSE, TDSC, ICSE, FSE, FM, ICML and AAAI. He has served as PC Co-chair of ICFEM (2024 and 2018), TASE (2023), FACS (2024 and 2009), and PC member of international conferences such as FM and TACAS.

Zhilin Wu, Institute of Software, Chinese Academy of Sciences, China

A Formally Verified Procedure for Width Inference in FIRRTL

Figure 27: Zhilin Wu's presentation.

Abstract: FIRRTL is an intermediate representation language for Register Transfer Level (RTL) hardware designs. In FIRRTL programs, the bit widths of some components may not be given explicitly, thus they must be inferred during compilation. In mainstream FIRRTL compilers such as *firtool*, the width inference is conducted by a compilation pass called InferWidths, which may fail even for simple FIRRTL programs. In this paper, we investigate the width inference problem for FIRRTL programs. We show that if the constraint obtained from a FIRRTL program is satisfiable, there must exist a unique least solution. Based on this result, we propose a complete procedure for solving the width inference problem, which can handle programs while

firtool may fail. We implement the procedure in Rocq and prove its correctness. From the Rocq implementation, we extract an OCaml implementation, which is the first formally verified InferWidths pass. Extensive experiments demonstrate that it can solve more instances than the official InferWidths pass in *firtool* using less time.

Biography: Zhilin Wu is Research professor at Key Laboratory of System Software (Chinese Academy of Sciences), Institute of Software, Chinese Academy of Sciences (ISCAS), China. His main research interests are formal verification of system software and digital circuit designs. He is the core contributor of the string constraint solver OSTRICH, the RISC-V CPU ISA consistency verifier χRVFormal, and BMCFuzz that integrates the bounded model checking and fuzzing for hardware verification. His research work has been published at top-tier international conferences and journals including LICS, POPL, CAV, FM, and DAC.

Jifeng Xuan, Wuhan University, China

Code Transformation and Defect Detection for Heterogeneous Programming Design: Trials and Thoughts

Abstract: Heterogeneous programming design is programming crossing two or more languages. In this talk, we would like to share some trials on code transformation and memory leak detection crossing different languages. The techniques in use are immature. We share some challenges and thoughts in our work.

Figure 28: Jifeng Xuan's presentation.

Biography: Jifeng Xuan, is a professor and a deputy dean at the School of Computer Science, Wuhan University. He is a standing committee member of the Technical Committee of Software Engineering at CCF, an executive committee member of the Technical Committee of Services Computing at CCF, a standing council member of the Wuhan Association of Software Engineering, a director of Wuhan Innovation Centre of Open Source Software Engineering. His research interest is software analysis and testing, including software testing and debugging, software data analysis, and search based software engineering. He serves as the editor of Empirical Software Engineering, PloS One, etc. and as the guest editor of Automated Software Engineering Journal, Science China Information Science, etc. He is the general co-chair of the National Conference of Evolutionary Computation and Learning (ECOLE 2021) and the program co-chair of the National Conference of Software (ChinaSoft 2025). He has won the ACM SIGSOFT Distinguished Paper Award, the IEEE TCSE Distinguished Paper Award, the IEEE ICSEM Distinguished Services Award, and the CCF Distinguished Doctoral Thesis Award.

Lei Bu, Nanjing University, China

From Specification Understanding to Verification: Formal Specifications as a Bridge Between LLMs and Trustworthy Software

Figure 29: Lei Bu's presentation.

Abstract: Formal program specifications provide a precise semantic foundation for understanding, generating, and verifying software. However, current large language models are still primarily evaluated and used through code-level tasks, while their ability to understand program semantics, produce rigorous specifications, and support end-to-end verification remains insufficiently explored. This talk presented a specification-centric perspective that connects these three challenges into a unified story. We first introduced formal specifications as semantic probes for evaluating code comprehension in LLMs, where specification judgement, selection, infilling, and generation tasks are used to examine whether models can capture program behaviours beyond individual execution traces. Building on this understanding-oriented view, we then present SpecGen, an LLM-based approach that generates formal specifications through conversational prompting, verifier feedback, and mutation-based refinement, aiming to produce specifications that accurately describe program behaviours. Finally, we introduced SpecSyn, which extends specification generation toward real-world program verification by decomposing large programs, synthesising segment-level specifications, and refining their semantic strength through non-equivalent program mutations and variant discrimination. Together, these works form a complete pipeline from specification-based semantic understanding, to automated specification generation, and further to verification-oriented specification synthesis, showing how formal specifications can serve as a bridge between LLM-based software intelligence and trustworthy program verification.

Biography: Lei Bu is professor and the vice dean of Software Institute, Nanjing University. He received his bachelor and PH.D. degree in Computer Science from Nanjing University in 2004 and 2010. He has been visiting scholar in institutes like Carnegie Mellon University, Microsoft Research Asia, and University of Texas at Dallas. His main research interests include formal method, model checking, hybrid system, and cyber-physical system. He has published around 80 papers in leading journals and conferences like TCAD, TC, TSE, TOSEM, TDSC, RTSS, CAV, ICSE, DAC, and so on. He has been awarded the CCF-IEEE CS Young Scientist Award, The Outstanding Teachers of Computing in Higher Education Award Program, Zhongchuang Software Talent Award, the NASAC Youth Software Innovation Award, the CCF Youth Talent Development Program, and the MSRA Star Track young faculty program, among others.

Figure 30: Jonathan Bowen illustrating the varied weights of Alan Turing and John von Neumann, with the busts outside the SETSS 2026 venue! ☺ (Photographs by Einar Johnsen.)

Conclusion

All the previous editions of the SETSS School have resulted in Springer LNCS volumes (numbers 9506, 10215, 11174, 11430, 12154, 15584, and 16481), as post-proceedings after the event. It has been agreed with Springer Beijing that the SETSS 2026 proceedings will appear in a similar way in the LNCS Tutorial series, in time for SETSS 2027.

For further information on the SETSS 2026 Spring School, see:

https://www.rise-swu.cn/SETSS2026/

Acknowledgement: Zhiming Liu provided helpful suggestions and corrections for an earlier draft.

References

[1] Bowen, J. P. (2024). Report on the SETSS 2024 Spring School. *FACS FACTS*, **2024**(2):60–71, July. BCS. https://www.bcs.org/media/1wrosrpv/facs-jul24.pdf#page=60.0

[2] Bowen, J. P. (2025). Report on SETSS 2025 workshop and school, Beijing, China, 17–23 May 2025. *FACS FACTS*, **2025**(2):35–45, July. BCS. https://www.bcs.org/media/yd4ocehl/facs-jul25.pdf#page=35.0

[3] Bowen, J. P., Gomes, C., Liu. Z. (eds): *Engineering Trustworthy Software Systems*: 6th International School, SETSS 2024, Chongqing, China, April 15–21, 2024. LNCS, vol. 15584. Springer, Singapore (2025). doi:10.1007/978-981-96-4656-2

[4] Bowen, J. P., Turrini, A. (eds): *Engineering Trustworthy Software Systems*: 7th International School, SETSS 2025, Chongqing, China, May 17–23, 2025. LNCS, vol. 16481. Springer, Singapore (2026). doi:10.1007/978-981-95-8617-2

The 22nd International Summer School on Trustworthy Software

Qin Li and Zhilin Zhou

East China Normal University (ECNU)
Shanghai, China

July, 2026

During 7–10 July 2026, the 22nd East China Normal University International Summer School on Trusted Software was successfully held at the Dishui Lake International Software College. Continuing the theme of "Formal Verification and Trusted AI," this year's event brought together many top international experts in the field. Young teachers and students from numerous universities and research institutions, both domestically and internationally. gathered in Lingang to complete four days of systematic and in-depth specialized lectures and academic exchanges.

Since 2004, the East China Normal University International Summer School on Trustworthy Software has been held for 22 consecutive sessions, providing a regular international academic exchange platform for young researchers and graduate students in the fields of computer science and software engineering, and helping to promote the development of disciplines and the cultivation of young talents in the fields of trustworthy software and formal methods in China.

Figure 1: Opening ceremony scene

This event featured lectures by four internationally renowned scholars: Professor James Woodcock, Fellow of the Royal Academy of Engineering and University of York; Professor Yamine Ait Ameur of the University of Toulouse, France; Alessandro Cimatti, Senior Research Fellow at FBK; and Professor Jonathan Bowen of London South Bank University. The experts delivered systematic lectures on the theme, covering fundamental theory, cutting-edge developments, and engineering practice, encompassing several popular research areas such as infinite-state system analysis, trusted verification of intelligent connected vehicles, Event-B domain modelling, and the interdisciplinary research of formal methods and artificial intelligence.

Figure 2: Alessandro Cimatti giving a lecture

Professor Alessandro Cimatti delivered a course over two days on SMT-based analysis of infinite-state systems, providing a comprehensive explanation of the underlying principles of SMT solutions, complex system modelling approaches, and automated verification techniques. Using examples from safety-critical systems such as industrial control and embedded systems, he outlined the complete technical framework for formal analysis in infinite-state scenarios, offering attendees a novel research approach to highly reliable software verification.

Figure 3: Jonathan Bowen giving a lecture

Professor Jonathan Bowen presented two main thematic sessions. The first traced the development of the integration of formal methods and artificial intelligence, outlining the history of their combination, current technological bottlenecks, and future research directions. The second focused on the academic career of Tony Hoare, a pioneer in computer logic, connecting classic theoretical systems such as program logic, formal foundations, and software verification to solidify the students' underlying theoretical foundation for formal research.

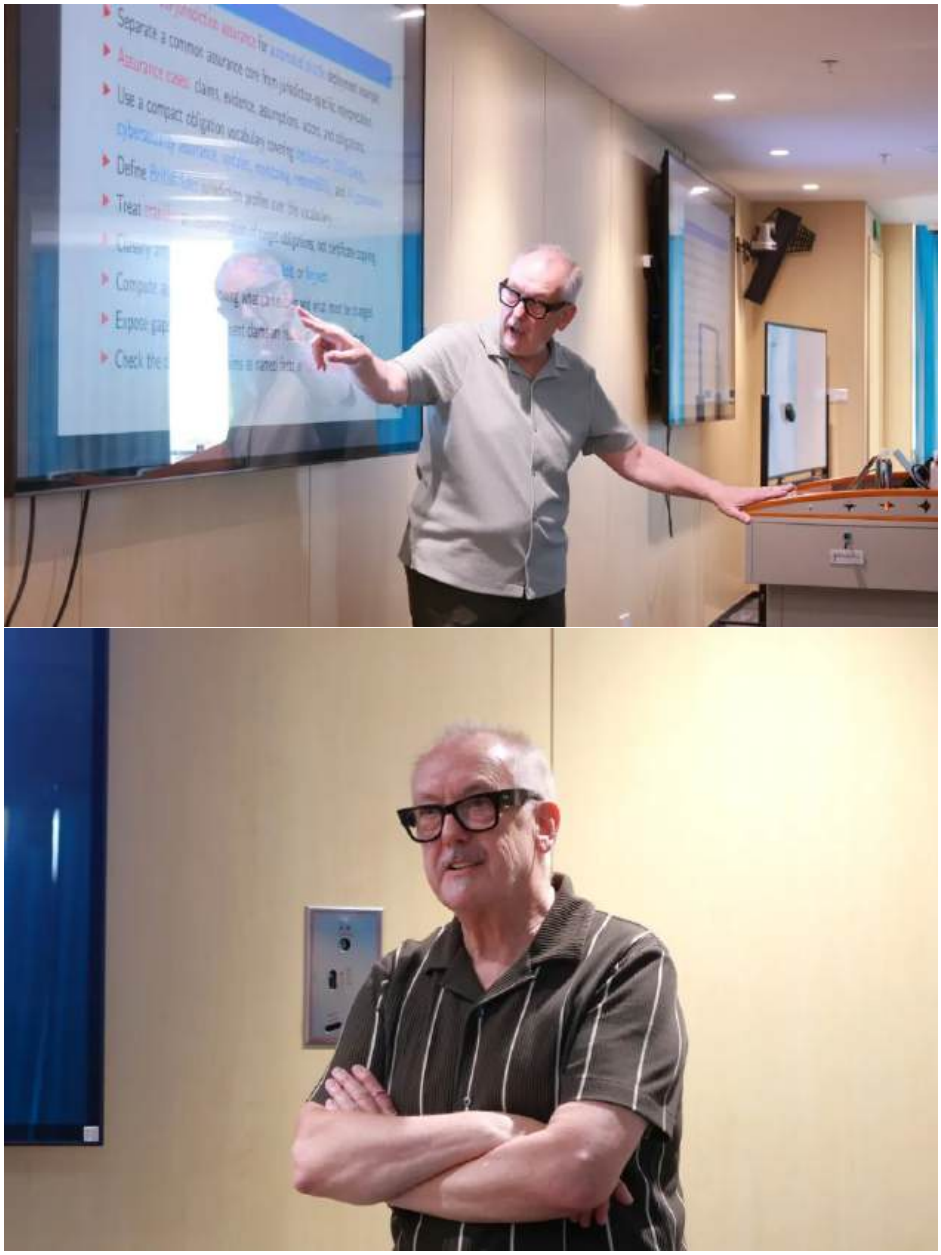


Figure 4: James Woodcock giving a lecture

Professor James Woodcock focused on the modelling and verification of intelligent connected vehicles (ICVs), delivering lectures over two days. He explained automotive-grade formal modelling specifications, risk constraint definitions, and the entire automated verification process, addressing the core safety requirements of autonomous driving scenarios. He also incorporated in-vehicle system engineering examples, bridging the gap between formal theory and the practical application of intelligent transportation technologies.

Figure 5: Yamine Ait Ameur giving a lecture

Professor Yamine Ait Ameur gave lectures over two days on domain-specific formal modelling based on Event-B, clearly distinguishing between descriptive and reduction modelling paradigms. He will fully demonstrate the practical process of building the Event-B toolchain, iterative optimization of the domain model, and verification of system invariants. The technical solution can be widely adapted to various application scenarios such as industrial software and safety-critical control systems.

The daily courses were designed to provide ample opportunities for interaction. After each lecture, a Q&A session was held where participating faculty and students actively asked questions related to their research areas, and the lecturers provided detailed and targeted answers. The tea breaks also served as a relaxed platform for academic communication, where faculty and students freely sat around and exchanged ideas on research challenges such as formal verification of large models, safety assurance for autonomous driving, hands-on practice of Event-B tools, and optimization of SMT algorithms. The atmosphere throughout the seminar was relaxed and engaging.

Figure 6: Summer school scene

This summer school provided young researchers with a direct learning channel to engage with cutting-edge international theories, promoting academic exchange across institutions and countries in the fields of trustworthy AI and highly reliable software. In the future, the Software Engineering Institute at ECNU will continue to bring in overseas experts to teach, focusing on research in trustworthy intelligent software, and building a comprehensive platform for talent development, scientific innovation, industry-education integration, and international exchange to support the development of the discipline of trustworthy software.

The Manchester Ferranti computer and Alan Turing

Jonathan P. Bowen

My daughter is a Reader in Chemistry at the University of Manchester, and my wife was recently searching for a suitable Manchester-based picture for her on eBay. During her searches, she spotted a page from a 1953 "magazine" with a photograph of a "calculating machine" at Manchester. On viewing this, I realised that it was a photograph of the Ferranti Mark 1 machine from the early 1950s, especially comparing it with the classic photograph including Alan Turing, the only known photograph of Turing with a computer (Figure 1; Lavington, 1975, p.17). What is more, the rather dark and ill-defined photograph on eBay included a man with his head half-turned away, who looked as if he could be Turing himself, wearing a similar light jacket to that of Turing in Figure 1, and with short dark hair.

The page was on sale for the price of a cup of coffee, so I thought it was worth a punt to see if the person could really be Alan Turing, and thus a second, till now unknown photograph of Turing with a computer. On receiving the page, it became more obvious that the person was wearing glasses, so it was highly unlikely to be Turing (Figure 2). Instead, the figure may be Frank Sumner (Lavington, 2013). In any case, it seems to be a photograph that is not available online. The publication itself does not seem to be available online either, after specific searches on Google Books and the Internet Archive for phrases in the text, as well as a general Google search.

Further searching on eBay revealed another interesting early 1953 advertisement for Ferranti, including a drawn image of the computer from another angle (Figure 3). This is based on a published photograph (Figure 4; Lavington, 1975, p.16). While my original hope of finding a second photograph of Turing with a computer was dashed, the search has produced two interesting and lesser-known images of the Manchester Ferranti computer.

With the availability of Generative AI (GenAI) facilities such as Google Gemini, low-quality photographs and images can now be significantly improved in quality, while retaining the original image's dimensions. It can even be used to "hallucinate" more speculative images from existing images (Figure 5). All the images in this short article have been post-processed with GenAI to improve their quality. More speculatively, colourisation is also possible and has been improving rapidly in recent years.

In conclusion, I still don't know the publication details for the first image discovered on eBay (Figure 2). The photograph is located on page 170 of a publication, in a section titled "CALCULATING MACHINE", also covering Hollerith punched card sorters, the ENIAC, etc. The caption for the photograph in Figure 2 reads as follows:

> *One of the latest examples of an electronic calculating machine is that completed at Manchester University in 1949, part of which can be seen in this photograph. It contains 4,000 electronic values, which replace the moving parts of smaller calculating machines. In it are also mounted 8 radar cathode ray tubes, which store the numbers used in the machine. This number store compares with the memory of the human brain. The control desk is at the far end.*

If anyone knows this publication, and/or can identify either of the people in Figure 2, do get in touch via jonathan.bowen@lsbu.ac.uk.

Acknowledgements: Thank you to Martin Campbell-Kelly and Dik Leatherdale for helpful suggestions based on an earlier draft.

References

Lavington, S. (1975). *A history of Manchester computers*. NCC Publications, pp. 16–17.
https://archive.org/details/HistoryOfManchesterComputers/page/16/mode/2up

Lavington, S. (2013). Obituary: Professor Frank Sumner. *Resurrection: The Bulletin of the Computer Conservation Society*, **64**, p.32.
https://www.computerconservationsociety.org/resurrection/res64.htm#g

Figure 1: Photograph of the Manchester Ferranti computer in 1952, with Alan Turing on the right (Lavington, 1975, p.17).

Figure 2: Photograph of the Manchester Ferranti computer with open racks from a 1953 publication.

Figure 3: Early 1953 advertisement for Ferranti, including a drawing of the Ferranti computer.

Figure 4: Photograph of the Manchester Ferranti computer (Lavington, 1975, p.16) used as a basis for the drawing in Figure 3.

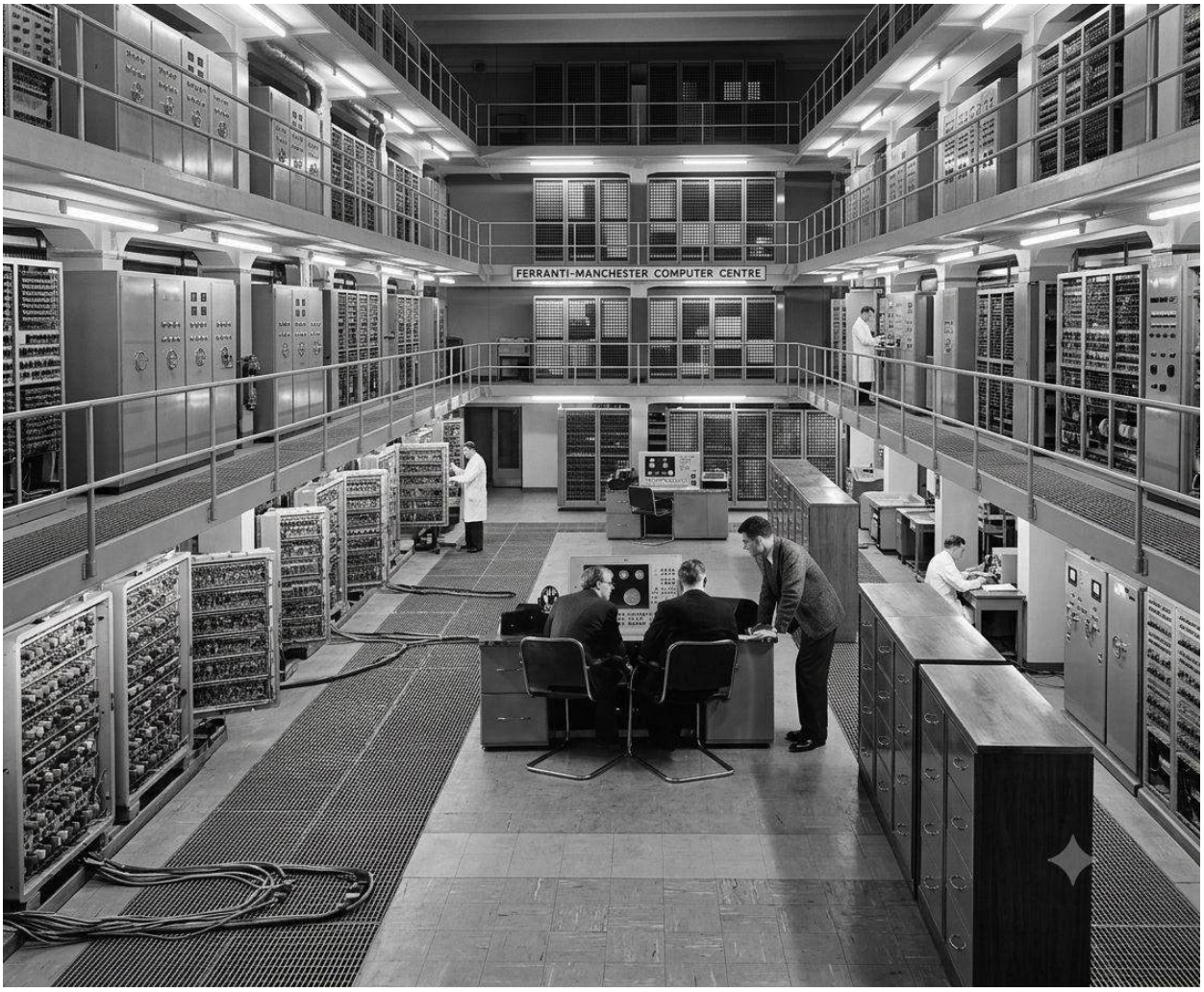


Figure 5: “Hallucinated” GenAI view of the Manchester Ferranti computer, generated from the photographs in Figures 1 and 2.

Hyper-algebras

Tim Denvir

Introduction

I want to present these ideas on hyper-algebras by revealing the thought-processes which led me to my tentative conclusions. I believe this approach is more communicative, rather than presenting a *fait accompli*, which is the way that, for example, proofs of mathematical theorems are usually presented. With that latter tradition, one is left with the thought, "but how did they think of this?", with no way of discerning the originator's cognitive process. So, with the approach I have adopted here, definitions will develop and be revised several times as the article proceeds, and I hope that this will be more revelatory. My conclusions are tentative and intended to invite comments and debate, which will be very welcome.

I have coined the term, Hyper-algebras, more specifically, universal multi-sorted hyper-algebras, to mean universal multi-sorted algebras which incorporate hyper-operations. I have written about hyper-operations before in *FACS FACTS* issue 2023-2 [1]. The positive integers can provide an example: applying successor repeatedly gives addition; repeated addition gives multiplication and repeated multiplication gives exponentiation. There conventionally we stop but it is possible to continue indefinitely, defining a function f_n where addition, multiplication and exponentiation are f_1, f_2, f_3 respectively. Then f_n where $n = 10^6$, for example can be used to define a very large number indeed[1].

That is a very brief summary of the thoughts that have led me up to this point. Why might one want universal hyper-algebras? I believe that they would form a good mathematical basis for the semantics of High-Level Language compilers. A HLL compiler takes as its input the script of a program and as a result of the compilation one can get an enormous variety of types and functions. For example, in Algol68, Ada or Chill one can define an unlimited variety of types, using set-theoretic operators (union, Cartesian product, lists, etc.), based on a few provided types in the language (integer, Boolean, characters etc.). Furthermore, types and functions can be defined recursively, with, in some of these languages, functions themselves being available as types. Here I hope to develop the idea of a hyper-algebra more generally: I believe it can provide the basis of a framework from which the semantics of a HLL such as those quoted, and any future ones which might be devised, at least in terms of types and functions.

[1] One can in theory generate all the natural numbers using the successor function repeatedly on zero, but even if one uses a shorthand like S^n the universe has a limited amount of paper, or indeed elementary particles of matter, to write a *really* big number, so that leads to the need eventually for hyper-operations. See my earlier *FACS* article [2]. Also, my attention has been drawn to a much earlier paper, "Some Rapidly Growing Functions", by Craig Smorynski, relating to very large numbers [7], far larger than any naturally occurring quantities in the physical world, even when those are qualified by probabilities or statistics.

Signatures, Sorts, Operations and Terms

A Universal Algebra is characterised by a Signature[2] A Signature Σ consists of a finite set of sorts, $\mathbf{S}$, a finite set of operations $\mathbf{Op}$, and a finite set of axioms, $\mathbf{Ax}$. Sorts can denote "carrier sets" in a more concrete view of an algebra.

Each $op \in \mathbf{Op}$ consists of a mapping $s^* \to s_r$ where s^* denotes a finite list, possibly empty, of $s_i \in \mathbf{S}$, and where $s_r \in \mathbf{S}$. If the list s^* is empty, the operation is a constant of sort s_r: A (deterministic) mathematical function will give the same result for the same values of its arguments each time it is invoked. So a function with no arguments has to be the same as a constant of whatever type is specified for its result, where the s_r is the result sort and the s^* are the 'argument' sorts of the operation.

Such a Universal Algebra can be interpreted as an algebra of Terms, where each sort denotes a 'carrier set'. A Term t is an expression which involves variables, operations, and constants (but see earlier regarding operations with an empty list of arguments). To create the possibility of terms involving more than operations whose arguments ultimately are merely constants, we need a formal way of denoting variables. To do so, we need an alphabet of characters, $\mathbf{C}$, where each $c \in \mathbf{C}$ is associated with a sort $s \in \mathbf{S}$. One might call this a type assignment, $\mathbf{T_A} : \mathbf{C} \to \mathbf{S}$. We assume that alphabet $\mathbf{C}$ is finite. (I also want to avoid the complexities of scope and overloading, certainly for the time being.)

Thus, so far, we have:

$$\Sigma = \mathbf{S}, \mathbf{Op}, \mathbf{C}, \mathbf{T_A} : \mathbf{C} \to \mathbf{S}$$

We define terms recursively. Any $c \in \mathbf{C}$ is a term. The *type* of such a term c is s where $\mathbf{T_A}(c) = s$. Also, for any operation $op \in \mathbf{Op}$, and terms t, $op(t^*)$ is a term of type s_r where the operator op has type $s^* \to s_r$, and each term t_i in t^* is of type s_i in s^*, i.e., such that $\mathbf{T_A}(t_i) = s_i$ where $\mathbf{T_A}$ is suitably extended. Denote the set of terms thus defined for a given signature $\Sigma$by $\mathbf{T}_\Sigma$.

Axioms

Finally, *Axioms*: the axioms are a finite set of logical expressions, involving Terms. There are various ways of forming logical expressions, involving connectives such as $\leq, <, \iff$, and negation $\neg$, and also universal and existential quantifiers $\forall$ and $\exists$. Means of converting terms into logical values are comparison operators such as $\geq, <, \supseteq, =$. An equational axiom of form $t_1 = t_2$ would therefore be of type $\mathbf{T}_\Sigma \times \mathbf{T}_\Sigma \to \{True, False\}$. In general, an axiom system $\mathbf{Ax}$ consists of a finite set of axioms of general type $\mathbf{T}_\Sigma^+ \to \{True, False\}$.

So, we have now elaborated on the foregoing:

$$\Sigma = \mathbf{S}, \mathbf{Op}, \mathbf{C}, \mathbf{T_A} : \mathbf{C}^+ \to \mathbf{S}^+, \mathbf{T}_\Sigma, \text{Ax: } \mathbf{T}_\Sigma^+ \to \{True, False\}$$

[2] I follow some of the notation of Donald Sannella & Adrzej Tarlecki [3], which is based on Ehrig & Mahr [4], and which in turn again is, I believe, inspired by Paul Cohn's *Universal Algebra* [5].

Where $\mathbf{S}, \mathbf{Op}, \mathbf{C}, \mathbf{T}_\Sigma$ are defined as above.

Hyper-operations

So far, all that is standard. Now, I want to consider hyper-operations in the context of a Universal Algebra as developed above. Our intended task is to enhance the definition of Op to include hyper-operations, thus making Σ the signature of an algebra in a Universal style, with hyper-operations, or therefore, a Universal Hyper-algebra.

The example of a hyper-operation given above, using the positive integers (Natural Numbers) with addition, forms a monoid (a set with an associative binary operation and an identity element). This use of repeated application of the operation does not work for all monoids. Consider the familiar two-valued logic with the $\wedge$ ("and") and T ("True") as identity, or indeed $\vee$ ("or") and F ("False") as identity. As soon as one tries to repeat the operation, the whole collapses: $x \wedge x = x$, and $x \vee x = x$. (The same applies to more general Boolean algebras.) It would be profitable to investigate what properties a monoid needs to have to enable it to generate an indefinite series of hyper-operations in the way that the positive integers can.

However, more generally, and we wish to be as general as reasonably feasible, one might be able to generate a hyper-operation in ways other than repetition of a single initial operation.

A more general hyper-operation will be a function taking as its arguments a list of basic operations and producing an infinite list of new operations. A countably infinite list of something of type FN is of type $\mathbb{N} \to FN$, where $\mathbb{N}$denotes the natural numbers $0, 1, 2, \ldots$. Let us, following some traditions, denote this by $FN^{\mathbb{N}}$. It will also be convenient to include hyper-operations within the definition of operations **Op**, using recursion, rather than having separate definitions of basic and hyper-operations. In that way a hyper-operation may be defined even more generally in terms of both basic operations and other hyper-operations. So the "target" of our generation mechanism will be of type $\mathbf{Op}^{\mathbb{N}}$ and the hyper-operation itself will be of type $\mathbf{Op}^* \to \mathbf{Op}^{\mathbb{N}}$.

Then the recursive definition of the fully general hyper-operation becomes:

$$\mathbf{Op} = (\mathbf{S}^* \to \mathbf{S}) \cup (\mathbf{Op}^* \to \mathbf{Op}^{\mathbb{N}})$$

with

$$\mathbf{T}_\Sigma = \mathbf{C} \cup \mathbf{Op}(\mathbf{T}_\Sigma^*)$$

We could dispense with the concept of **C**, observing that **C** is identical to operations with no arguments, i.e. $\mathbf{C} = \{\, c \in \mathbf{Op} \mid \mathbf{arity}(c) = 0 \,\}$, (see the remark on functions with no arguments above) but this probably does not help with clarity of presentation.

And Finally

Finally, as a signature of a universal multi-sorted hyper-algebra, we have:

$$\Sigma = \mathbf{S}, \mathbf{Op}, \mathbf{C}, \mathbf{T}_\Sigma, \mathbf{Ax}$$

where:

$$\mathbf{Op} = (\mathbf{S}^* \to \mathbf{S}) \cup (\mathbf{Op}^* \to \mathbf{Op}^{\mathbb{N}})$$
$$\mathbf{C} = \{\, c \in \mathbf{Op} \mid \mathbf{arity}(c) = 0 \,\}$$
$$\mathbf{T}_\Sigma = (\mathbf{C} \cup \mathbf{Op}(\mathbf{T}_\Sigma^*) = \mathbf{Op}(\mathbf{T}_\Sigma^*) \quad \textit{Because } C \subset Op((\boldsymbol{T}_\Sigma^*)$$
$$\text{Ax: } \mathbf{T}_\Sigma^+ \to \{True, False\}$$

The mathematically alert reader will notice that **Op** contains its own function space, which is not possible for any valid mathematical set. However, the whole motive for this exercise is to provide a mathematical basis for HLL compilers in a general form. Thus, **Op** is intended to capture the semantics of *computable* functions and operations. The classical means of doing this is to restrict ourselves to Domains rather than fully fledged mathematical sets and functions, through Domain Theory, as originally proposed by Scott [8] and others. **Op** should be viewed in that Domain Theory context.

That, then, I submit, is a Signature for a Universal Multi-Sorted Hyper-Algebra which can form the mathematical basis for semantic definitions of conventional high-level language compilers, with their capability for defining arbitrary, wide-ranging functions and types.

One might object that hyper-operations could themselves be expressed as single recursive operations, or by means of an extra argument denoting the depth of recursion. I would answer that, while this may look feasible for the integer example used for illustration here, it would be much more difficult to demonstrate for the more general cases where hyper-operations may be generated in many more complex ways, indeed, ways that we may not conceive of.

Where Next?

A Signature for an algebra thus essentially comprises a trio: Sorts, which refine into Carrier Sets and Types, Operations and Axioms. We have seen how to extend Operations into Hyper-Operations, leading to a definition of of Hyper-Algebras. Can we go further and define Hyper-Carriers (Hyper-Types) and Hyper-Axioms?

I think not. Already within traditional set-theory notation there are mechanisms (such as infinite unions, $\cup_{i \geq 0} S_i$, for example) for defining infinite families of sets, dispensing with the need for Hyper-Carriers or Hyper-Types. Likewise the arsenal of logical operators, Universal Quantification ($\forall$) coming readily to hand, would seem to make redundant any special mechanism for forming Hyper-Axioms.

Hyper-Algebras, incorporating Hyper-Operations, as we have defined them, seem to be "sufficient unto the day"[3]. But, please, debate is invited: The impact of the context of Domain Theory would benefit from more detailed elaboration.

[3] Recalling the well-known ancient proverb, "Sufficient unto the day is the evil thereof", which can be loosely paraphrased as "enough to consider for now". For the origin, see [6].

Acknowledgements

I am most grateful to comments and advice from other members of the *FACS* editorial team whose assiduous reading of my drafts has been of much technical insight and benefit. Any remaining errors or inadequacies remain the author's, however.

References

[1] Tim Denvir, *A Brief History of Hyperoperations (Marking a Centenary)* FACS FACTS Issue 2023-2, July 2023, **https://www.bcs.org/media/10894/facs-jul23.pdf**

[2] Tim Denvir, *Rambling Thoughts on VLNI (Very Large Numbers Indeed)* FACS FACTS Issue 2012-1, November 2012, **https://www.bcs.org/media/3088/facs-nov12.pdf**

[3] Donald Sannella & Andrzej Tarlecki, *Foundations of Algebraic Specification and Formal Software Development*, Springer 2012.

[4] H. Ehrig & B. Mahr, *Fundamentals of Algebraic Specification 1, EATCS Monographs on Theoretical Computer Science*, Volume 6, Springer-Verlag 1985.

[5] Paul M. Cohn, *Universal Algebra, Mathematics and Its Applications/6*, Revised Edition, D. Reidel 1981.

[6] New Testament, Matthew 6.34, King James' version 1611 and later editions and printings. (The same verse in subsequent versions are scarcely recognisable as the same text.)

[7] Craig Smorynski, *Some Rapidly Growing Functions*, Elsevier Science Publishers B.V., North-Holland, 1985.

[8] Dana Scott, Princeton University, Outline of a Mathematical Theory of Computation, Technical Monograph PRG-2 Oxford University Computing Laboratory Programming Research Group.

Formal Methods Informally by Carroll Morgan
A book review and associated talk

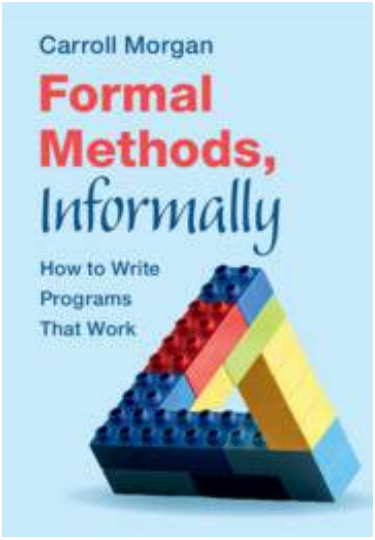
July 2026

(**Reviewed by:** Jonathan P. Bowen)

Book Review

Here we review a book by Carroll Morgan [7] assocated with a BCS-FACS talk.

Formal Methods, Informally: How to Write Programs That Work



For over half a century, the field of formal methods has suffered from a serious public relations problem. To the average software engineer, it evokes images of dense predicate calculus, terrifying mathematical proofs, and automated verification tools that require a PhD to operate. While these rigid techniques are essential for flight-control software or operating system kernels, they rarely make it into the daily workflow of the mainstream programmer.

Carroll Morgan, a leading researcher in the field of formal programming methodology and the author of the book *Programming from Specifications* [6], seeks to correct this disconnect. In his new book *Formal Methods, Informally* [7], Morgan aims to strip away the intimidating mathematics of formal logic, leaving behind its most practical and useful core: a structured way of thinking about code.

Instead of forcing readers to write proofs, Morgan teaches them how to reason about programs using intuitive, everyday concepts. The text is split into four primary parts, starting with "Everyday programs", which functions as a standalone guide to improving logical programming habits. Rather than overloading the reader with syntax, the book revisits classic algorithmic building blocks – such as searching, sorting, complexity ($O(n)$ notation), data structures, and concurrency – through a new viewpoint. The strategy of the book relies on a few core principles concerning understanding, learning, and pragmatism.

Comments on "what is true here" are recommended for understanding in programs. Morgan argues that the greatest tool of a programmer is not as a "linter", performing mental static analysis, but instead including simple and precise comments, in the form of assertions, even if done informally. By training readers always to determine the state of the data at each line of code, he teaches Hoare-style invariants and preconditions without ever invoking daunting academic jargon.

An incremental learning approach is adopted. The book deliberately uses basic and lightweight code examples. Morgan handles cognitive load carefully; he does not try to teach a complex algorithm and a new way of thinking at the same time. For example, he presents the reasoning for simple loops first, developing the necessary intellectual ability gradually.

Pragmatism is embraced with regard to social and design issues. In a refreshing approach for a textbook motivated by formal logic, Morgan acknowledges that bugs are not just mathematical errors; instead, they are often the result of flawed human communication. The book covers how informal reasoning helps developers divide a large project among team members and cleanly combine it back together again.

By way of overview, the book is divided into four main parts: I) *Everyday programs*, including loops, invariants, variants, assignments, and conditionals; II) *Data structures and their encapsulation*, with more on invariants, data types, and case studies; III) *Concurrency – and how to check it*, including the Owicki-Gries method [8] for proving the correctness of concurrent programs, Peterson's algorithm for mutual exclusion, and garbage collection; IV) *Machine-assisted program checking, and testing*. The appendices include "drill" exercises (ideally to be undertaken without consulting the main book text, to bed in important skills), rules for checking programs, data refinement conditions, logical identities, an illustration of heap behavior, some issues specific to the Python programming language, and selected answers to exercises. There is also a bibliography with many classic formal methods publications and an index. The book's associated website (**`https://www.cambridge.org/Morgan`**) provided online access to additional material, such as instructor resources including answers to exercises, which can be unlocked on request by validated teachers.

The book's strengths include the following. It has a pedagogical appoach: Morgan's writing style is witty, accessible, and always grounded. Many years of teaching experience are obvious from the book's content. It feels less like a rigid lecture and more like a friendly discussion with a knowledgeable and patient mentor. It also has a flexible approach: you do not need to adopt a new language or toolset to use this book. Whether you code in Python, C, or Java, you can apply the "informal" reasoning patterns to your next programming project. Finally, the book is practical: with over 200 exercises (many including complete solutions directly in the text), the book serves as a highly practical workbook.

Despite its strengths, the book does have some limitations. Firstly, it is not a programming language primer: if you are a complete beginner who does not yet understand standard imperative programming control flow (such as conditional statements or loops), this book is not the place to start. It requires at least a basic familiarity with writing programs. Secondly, purists may not approve of the book's approach: hardline computer science theorists might miss the absolute mathematical rigour exhibited in Morgan's earlier work [6]. However, criticizing the book for lacking explicit formal logic misses its stated objective concerning informality.

There is no mention of "artificial intelligence" (AI) in the index. It will be interesting to see if the ideas in this book can be integrated with current developing approaches such as "vibe coding", allowing program generation through natural language prompts. Whether this will lead to "vibe proving" as well, only time will tell. The book concentrates on the human understanding of programs in their production and it remains to be seen if AI can aid in this. Perhaps this will be the

subject of a future book.

The target readership for this book is undergraduate computer science students, self-taught programmers looking to improve their skills, and experienced veteran developers seeking to better understand why their intuition works. It may not be best as the first course on programming in a degree programme, but would make an excellent subsequent course, perhaps in the second year of a computer science degree. Many professional programmers would also benefit from reading and absorbing the material in the book. The hardback version of the book (ISBN 978-1-009-42099-0) is relatively expensive at £90, no doubt mainly aimed at libraries, but the paperbook version (ISBN 978-1-009-42102-7) is more affordable for individuals at £40.

In summary, *Formal Methods, Informally* is a helpful addition to the canon of computer science education with respect to programming. It successfully uses some of the most valuable aspects of formal verification, but makes them accessible to the "average programmer", providing a helpful pragmatic approach for everyday use by programmers, even if they do nnot have a formal methods background. For those that wish to write cleaner program code, reduce the time spent debugging, and learn how to mentally trace a loop without being over-formal, this book deserves a place on the bookshelf. It is an excellent reminder that writing correct programs is less about syntax and much more about clarity of thought.

Talk Information

This section is an augmented version of the information provided about the talk on the BCS-FACS website.

Why might we teach "Formal Methods" informally? And to whom? And when?

Tracing the evolution of formal methods from Hoare to today, this online talk by Carroll Morgan, held on 25 June 2026, and associated with his recently published book *Formal Methods, Informally* [7], suggested ways to teach program reasoning to all programmers.

Synopsis: Formal methods (for computing) was enabled by Hoare's "An axiomatic basis for computer programming" in 1969 [4], where "formal" meant "in the sense of formal logic", i.e., reasoning based on manipulation of symbols which themselves had no intrinsic meaning (in the sense of the German mathematician David Hilbert).

Hoare's paper had itself been enabled by Floyd's "Assigning meanings to programs" [3] just two years earlier, whose informal message might well have been – in retrospect at least – "Do not write your comments inside a flowchart's boxes: instead, write them on the lines connecting those boxes."

After that, however, formal methods gradually became "Use preconditions, postconditions, and invariants, all written in predicate logic over your program's variables" to improve the chance that

your program does what it is supposed to do, and even "Use the very formality of that logic, and its own calculus even if by hand, to help you do that." Even better, "Use that formality to help you develop your program, your algorithm: strive for correctness by construction."

And so, more than fifty years later, it is that very formality that has made it possible to verify extremely complicated programs, even whole operating systems, by using computer-implemented proof tools: for "formality" is all those tools understand. Yet over all those decades, the quality of code produced by the average programmer might not have improved much at all.

This talk presented some ideas for fixing that, targeting in particular the "average programmer". It included suggestions about when we might introduce students to reasoning about programs, how to do that introduction, and what kind of assessment might work for many hundreds of submissions of exercises, quizzes, and exams that are too onerous at that scale to mark by hand.

Biography: Carroll Morgan obtained a BSc at the University of New South Wales (UNSW) and a PhD at the University of Sydney. He then worked in industry for about three years, as a programmer/analyst at a small company in Sydney.

In 1982, he moved to the UK to join Tony Hoare's Programming Research Group at Oxford University, first as a post-doctoral researcher at Wolfson College, then, in 1985, taking up a permanent Fellowship in Computation at Pembroke College, Oxford. When he returned to Australia in 2000, he first worked again in industry (working on J2EE-based web frontends), but soon (re-)joined UNSW, collaborating as an adjunct professor with Ken Robinson.

In 2002, about to lead the Formal Methods Group within the newly formed NICTA (National ICT Australia), he was at the same time awarded a five-year Australian Professorial Fellowship at UNSW to start in 2003. Taking the latter, he remained at UNSW from then on, within NICTA that became Data61, which in turn became the Trustworthy Systems Group, which is now led by Gernot Heiser.

Carroll Morgan has authored (in some cases jointly) the texts *Programming from Specifications* [6], *Abstraction, Refinement and Proof for Probabilistic Systems* [5] with Annabelle McIver, *The Science of Quantitative Information Flow* [2] with Mário Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Catuscia Palamidessi and Geoffrey Smith, and, most recently, the topic of this talk, *Formal Methods, Informally: How to Write Programs That Work* [7]. In 2003, Morgan was awarded the Best Z/B Paper overall for "Probabilistic termination in B" with Annabelle McIver and Thai Son Hoang at the ZB2003 conference [1]. In 2015, he was awarded (with the other five authors for [2] above) the NSA (US National Security Agency) Best Scientific Cybersecurity Paper award.

Morgan became Professor Emeritus at UNSW upon his retirement in 2024, and remains a member of the Trustworthy Systems Group there.

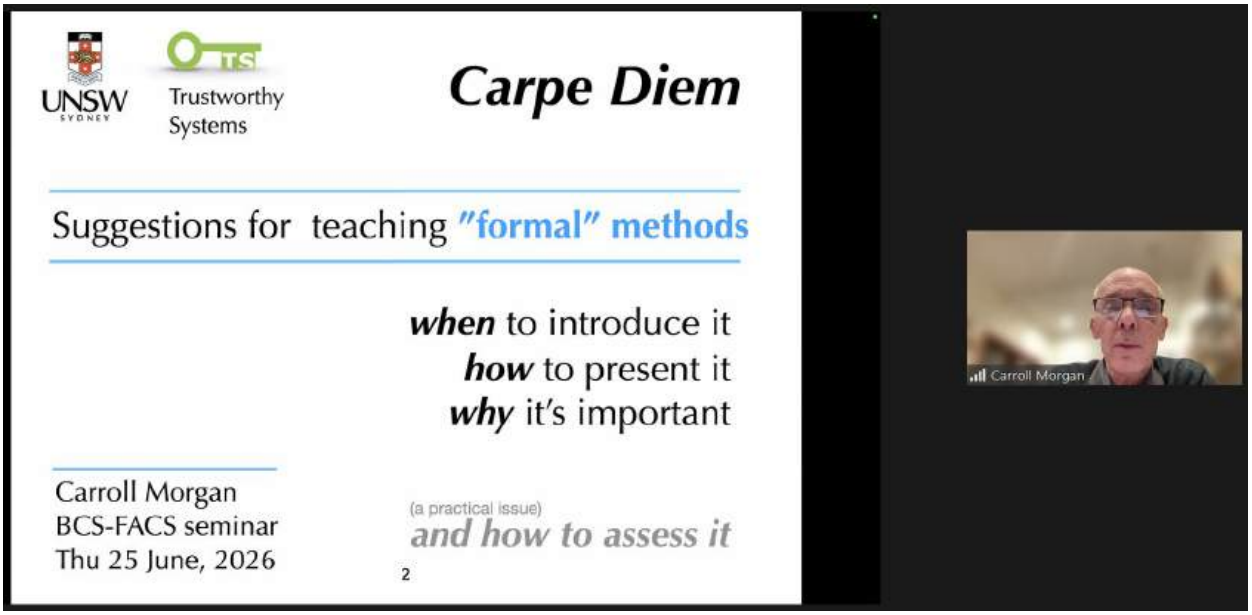


Figure 1: The first slide of the talk.

Talk Review

The online talk took place at 11 a.m. (a convenient time for the speaker in Australia) on Wednesday, 25 June 2026, a day of record-breaking June temperatures in the United Kingdom, but an unusually cold day in Australia, where the speaker, Carroll Morgan, was based, wearing a jumper. The talk's title on the first slide was *Suggestions for teaching "formal" methods*, with subtopics of 1) *when* to introduce it, 2) *how* to present it, 3) *why* it's important, and the practice issue of 4) how to assess it (see Figure 1).

The talk assumed a background of formal methods in the audience, which was well judged in the circumstances. There were up to 70 attendees online, sadly interrupted by a technical glitch at the end of the talk, but still with a good discussion and set of questions when we were able to resume.

Part I of the talk on when to introduce the suggested approach noted that first-year students are still "vulnerable" to believe what they are told. The traditional computer science programme approach is to teach discrete mathematics and logic, with the hope that this will feed into an introductory programming. However, it was noted that this often does not happen as intended in practice. Instead, a different approach is suggested.

In Part II of the talk, Carroll suggested that it is best to start such a course in a simple way and discuss the documentation that is beneficial even in a simple program. He provided an example of a simple loop to sum numbers in an array used to illustrate the point. The naive approach is to add comments indicating what each important line in a program does. However, the speaker suggests that it is more useful to record what is true at any important point in the program (i.e., with "assertions" in formal methods parlance, be they preconditions, invariants, or postconditions. It is normally not significantly more difficult than the traditional approach. The equivalent of a specification can be included at the start of the program, even if not fully formal. Indeed, this style

of comment leads to the much more natural and even more elegant generation of program code.

A notable comment from one of the students on such a course was:

> "After taking this course I felt as if I was privy to a powerful ancient magic. Everything that was introduced was so useful yet not yet known to most CS students."

The talk noted that it is best to start this approach at the same time as the existing first programming course on a computer science programme, following the same course structure and with the same exercises, but using the "What's true here?" mindset instead. It was noted that the use of invariants in designing loops is a critical skill, even if not done fully formally.

Part III of the talk noted that it is important to gain the confidence of the students that this approach is beneficial for their programming skills. An example was given of a relatively simple set of semi-formal requirements for subsegments in a sequence. In the first attempt at this, intuition and diagrams were used interactively with students, making mistakes and correcting them in the traditional informal approach to programming. It was noted that this can be a "fun" but not very efficient way to program. It can be done interactively with student participation on a whiteboard. However, the reliability of the result is questionable.

The approach with invariants can then be tried with students. Once suitable invariants have been determined, a program can be generated from these. In the example, the resulting program was about half the length of that produced using the traditional ad hoc approach. What is more, it is likely to be more reliable as well. Of course, the program should still be tested, but it has more chance of being correct with respect to the requirements than the longer program generated in the more traditional manner.

After experiencing this alternative approach, one student commented:

> "This course radically changed the way I view programming, and has certainly improved my programming skills immensely. This course should be compulsory for all Computer Science students, or have its contents integrated into the first year."

The talk covered an attempt to get ChatGPT to solve the problem. The specification needed to be rephrased into a form more acceptable to ChapGPT. It did produce a programming answer (in Python), but considerably more complex than the "informal" formal methods approach with invariants.

In Part IV of the talk, Carroll discussed some practical assessment issues, for example, if there are a thousand students taking the course. In this case, at least partly automated marking becomes helpful. It is desirable to assess the skills that students possess in *constructing* programs. Some examples of experience in delivering the course during 2022 and 2023 were given, including changes made between these deliveries, at the University of New South Wales in Australia on a course entitled *(In-)Formal Methods; the Lost Art*. The course included a project and weekly quizzes, but no final exam. The reasoning given was the programming in the timescale of an exam was too short to be realistic.

The quizzes could be done over a weekend with randomized questions/answers on Moodle, and with two attempts allowed. Also only the best 7 of 9 quizzes counted towards the 40% of the total marks so one or two disastrous quizzes did not count against students. A typical quis was shown with "what's-true-here" comments and a randomized selection of possible program fragments provided for menu selection by the students. Thus there were many randomized possible answers, with only one being correct.

The project for 50% of the marks lasted for a full term, which may be two complex for first-year students, but the course was delivered in the second year. 10% of the marks were allocated for attendance.

Discussion

The Zoom session failed fortuitously near the end of the talk, so a discussion was still possible once the session was reestablished (see Figure 2). The discourse included the issue of how the rapid developments in artificial intelligence (AI) may impact the suggested approach, not mentioned in the book, but covered with a ChatGPT example in the talk itself. At the moment, ChatGPT cannot compete with humans with the use of "informal" formal methods, but of course, this could change in the future, as more examples become available online for Generative AI (GenAI) Large Language Models (LLMs).

An audience member educated in the formal methods tradition noted that it is a necessary but not sufficient condition to be a "believer" in formal methods. Probably most of the audience for the talk are still believers! The question of whether this approach could be used in a functional programming setting was raised. Carroll noted that this type of programming is a very logical technique anyway, so those using it are naturally drawn to a formal methods style of thought.

It was noted that many formal methods courses exhibit a "U" curve rather than the more desirable bell curve in the distribution of student results; i.e., students either get it or are flummoxed by it. It was pleasing that Carroll was able to report that the presented approach exhibited a bell curve distribution of marks.

Whether this approach could be used in the first year of a computer science programme rather than the second year was discussed. Carroll Morgan comments that at the moment, many computer science lecturers are not au fait with the approach adopted in the book, let alone students. So making this approach generally accepted may be a long-term battle. At least the students who have attended the course seem to appreciate it, and some of these may become lecturers in the future. Let us be optimistic that they can gradually spread the word!

A recording of the talk and full discussion can be found on YouTube:

https://www.youtube.com/watch?v=GRdDVsHxc_M

Figure 2: Online discussion after the talk.

References

[1] Boca, P., Bowen, J. P. (2003). "BCS-FACS: A report from the trenches". *FACS FACTS*, vol. **2003**, no. 1, pp. 2–9, July. BCS. **https://www.bcs.org/media/5037/issue-2003-1-july-2003.pdf#page=2**

[2] Alvim, M. S. , Chatzikokolakis, K., McIver, A., Morgan, C., Palamidessi, C., Smith, G. (2020). *The Science of Quantitative Information Flow*. Springer, Information Security and Cryptography. doi:10.1007/978-3-319-96131-6

[3] Floyd, R. W. (1967, reprinted 1993). "Assigning meanings to programs". In: Colburn, T. R., Fetzer, J. H., Rankin, T. L. (eds), *Program Verification. Studies in Cognitive Systems*, vol. **14**, pp. 65–81. Springer, Dordrecht. doi:10.1007/978-94-011-1793-7_4

[4] Hoare, C. A. R. (1969). "An axiomatic basis for computer programming". *Communications of the ACM*, vol. **12**, no. 10, pp. 576–580, 583. doi:10.1145/363235.363259.

[5] McIver, A., Morgan, C. (2005). *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, Monographs in Computer Science. doi:10.1007/b138392

[6] Morgan, C. (1990, 2nd ed. 1994), *Programming from Specifications*. Prentice Hall, Series in Computer Science. **https://www.cs.ox.ac.uk/publications/books/PfS/**

[7] Morgan, C. (2026). *Formal Methods, Informally: How to Write Programs That Work*. Cambridge University Press. **https://www.cambridge.org/Morgan**

[8] Owicki, S., Gries, D. (1976). "An axiomatic proof technique for parallel programs I". *Acta Informatica*, vol. **6**, pp. 319–340. doi:10.1007/BF00268134

An Obituary of Sir Charles Antony Richard Hoare (1934–2026)

Jonathan P. Bowen

July 2026

Abstract

Sir Charles Antony Richard (Tony) Hoare was a titan of computer science whose work bridged the gap between the abstract elegance of mathematical logic and the practical necessity of reliable software. Best known for the Quicksort algorithm, the development of Hoare Logic, and the formalization of Communicating Sequential Processes (CSP), his career was defined by a relentless pursuit of "correctness by construction". This obituary chronicles his journey from a student of the Classics to the recipient of the Turing Award, exploring a legacy that transformed programming from a craft into a disciplined science.

"Excellence is never an accident. It is always the result of high intention, sincere effort, and intelligent execution; it represents the wise choice of many alternatives – choice, not chance, determines your destiny." — attributed to Aristotle

Prologue: A life of logic

The world of computing lost its most rigorous conscience in early 2026 with the passing of Sir Tony Hoare at the age of 92. In an era where software often prioritized speed and "feature creep" over safety and reliability, Hoare stood as a steadfast advocate for the mathematical proof. To Hoare, a program was not merely a set of instructions for a machine; it was a mathematical object, subject to the same laws of logic as any geometric theorem.

His passing marks the end of an era – the generation of pioneers who built the foundations of high-level programming. Yet, his influence remains embedded in every safety-critical system, every recursive function, and every concurrent language used today.

Early Life and the Path to Computing

Born in Colombo, Ceylon (now Sri Lanka), in 1934, Tony Hoare's upbringing was far removed from the digital world he would eventually help build. His early education focused on the "Greats" at Oxford University, officially known as "Literae Humaniores" (Classics) at Oxford.

This background in classicial philosophy and logic was more than a mere footnote; it was foundational. Hoare's later insistence on the precision of language and the structural integrity of logic can be traced directly back to his study of Aristotle and the rigourous grammar of the ancients. He famously noted that his training in the Classics provided a better preparation for programming than many realized, as both required an obsession with syntax and the internal consistency of systems.

The Birth of Quicksort: Moscow and the Navy

In the late 1950s, following a period of national service in the Royal Navy, where he learned Russian, Hoare studied at Moscow State University. It was here, while working on a project for machine translation involving the sorting of words in a dictionary, that he conceived of *Quicksort* [9, 10].

At the time, the computing world relied on far less efficient methods. Hoare's insight was the "partition" strategy: selecting a "pivot" element and organizing the list into elements smaller and larger than the pivot. Mathematically, the elegance of Quicksort is expressed in its average-case complexity:

$$T(n) = O(n \log n)$$

Though he initially struggled to explain the recursive nature of the algorithm to his colleagues, it eventually became the most widely used sorting algorithm in the world, a testament to his ability to find the most efficient path through a sea of data.

The Professional Era: Elliott Brothers and ALGOL

Upon returning to the UK, Hoare joined Elliott Brothers, a small computing firm [15]. His work on the ALGOL 60 compiler was a formative experience in software engineering. He was a champion of the ALGOL language for its clarity and structural soundness, but he was also a vocal critic of its "dangerous" features. It was here that Tony met his future wife, Jill Pym, an important member of the ALGOL 60 compiler team.

It was during this period that Hoare introduced the *null reference* – a decision he would famously label his "billion-dollar mistake" decades later [8]. Designed to be a convenient way to represent the absence of a value, the null pointer led to decades of system crashes and security vulnerabilities. His public apology for this feature exemplified his humility and his belief that a computer scientist's greatest duty is to admit when a design lacks the necessary safeguards.

Hoare Logic: Programming as a science

In 1968, Hoare moved into academia, taking a chair at Queen's University Belfast. It was here that he published one of the most influential papers in the history of the field: *An Axiomatic Basis for Computer Programming* (1969) [11].

In this work, he introduced what is now known as the *Hoare Triple*:

$$\{P\}C\{Q\}$$

This notation revolutionized software verification. It posits that if a precondition P is true before the execution of a command C, then the postcondition Q must be true upon its completion (assuming the program terminates). This allowed developers to formally prove the correctness of their code, moving the industry away from "debugging" and toward "verification".

Oxford and Concurrency

Hoare returned to Oxford in 1977 as the Professor of Computing, where he built the Programming Research Group (PRG) into a world-class institution, taking over after the untimely death of the PRG's founder, Christopher Strachey (1916–1975).

At Oxford, as hardware evolved toward multi-core processors, the problem of concurrency – how different parts of a program interact simultaneously – became paramount. Hoare addressed this with *Communicating Sequential Processes* (CSP) [12, 14]. This influenced the development of the Occam parallel programming language and the related Transputer microprocessor hardware by Inmos [24].

Prior to CSP, concurrency was often handled through shared memory, which was notoriously prone to "race conditions". Hoare proposed a model where processes communicate by sending messages through channels, ensuring that no two processes could interfere with each other's internal state. This model provided the theoretical foundation for languages such as Occam, Erlang, and the modern-day Go (Golang) and Rust.

Provably Correct Systems

The initial inspiration for the European "ProCoS" initiative [4] stemmed from Hoare's 1986 sabbatical at the University of Texas at Austin in the United States, where he observed the "Stack" verification project undertaken by Computational Logic, Inc. (CLInc). Driven by the challenge of comprehensively verifying a computing system across multiple layers, Hoare, with the support of Dines Bjørner at the Technical University of Denmark, conceived the idea of mathematically formalizing system correctness properties from top to bottom.

In 1989, Hoare facilitated European collaboration through a meeting in Oxford with some key formal methods researchers in Europe. This led to the formation of the initial ProCoS consortium, which brought together institutions including Oxford University, the Technical University of Denmark (DTU), and Kiel University in Germany.

Hoare served as a core leader throughout the lifespan of the initiative. During the first ProCoS project (1989–1992), he was one of the principal planners and investigators who defined the project's multi-layered approach to mastering system complexity. For ProCoS II (1992–1995), Oxford University took over as the overall coordinating site for the second phase of the project, with Hoare serving as the overall project manager and coordinator. Under his management, the project explored connecting diverse levels of abstraction – moving from real-time requirements to system specifications, through compilation down to machine code, and directly into hardware netlists using a representative subset of the occam programming language and the Transputer architecture.

Beyond administrative leadership, Hoare's collaborative research at Oxford yielded one of the most significant and enduring theoretical outputs of the ProCoS initiative, namely *Duration Calculus*. Together with Zhou Chaochen and Anders P. Ravn, Hoare co-authored the seminal first paper on Duration Calculus [30]. This interval-based logic became a core conceptual framework within ProCoS for formalizing and specifying real-time requirements and dynamic system prop-

erties.

Unifying Theories of Programming

Working closely with Professor He Jifeng at Oxford, including during the ProCoS projects, Hoare investigated how different computational paradigms (sequential, parallel, centralized, distributed, hardware, and software) could be unified. Influenced heavily by their ProCoS collaborations, they proved that these diverse paradigms could be embedded within a single mathematical theory of relations. This research culminated in their landmark jointly authored 1998 book, *Unifying Theories of Programming* [16], establishing a framework that allows a single model-based language (like the Z notation or VDM) to specify complex, heterogeneous systems at the highest level of abstraction.

Later Years: Microsoft Research

After reaching the mandatory retirement age at Oxford, Hoare did not cease his work; instead, he joined Microsoft Research in Cambridge as a Senior Researcher. He also had the opportunity for some retrospection [15, 23, 24].

During his time at Microsoft, he focused on the "Grand Challenge" of the *Verified Software Initiative* [18]. He argued that the world's critical infrastructure – banking, air traffic control, medical devices – should be built on software that is mathematically proven to be correct. He remained active in the community well into his 80s, mentoring a new generation of researchers who sought to realize his vision of a world with reduced software "bugs".

Honours, Legacy, and Philosophical Impact

The accolades bestowed upon Sir Tony Hoare were numerous, and included:

- The *ACM A.M. Turing Award* (1980) for his fundamental contributions to the definition and design of programming languages [13].
- The *Kyoto Prize* (2000) for Information Science.
- A *Knighthood* (2000) for services to education and computer science.
- The *IEEE John von Neumann Medal* (2011).
- The *Royal Medal* of the Royal Society (2023).

However, his true legacy is not found in medals, but in the philosophy of software correctness and simplicity. He taught us that [13]:

> "There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult."

Epilogue: The logic remains

Sir Charles Antony Richard Hoare was a man of great intellect, but also of immense kindness and wit. He treated the lowliest student and the highest-ranking executive with the same polite, inquisitive attention. He believed that the human mind was capable of mastering the complexity of the machine, provided it had the right logical tools.

Celebratory volumes of contributed chapters for Tony Hoare by colleagues were produced in 1989 [17], in 1994 [25], for his retirement in 1999 [6], and then later in 2010 [21] and 2021 [20]. He was interviewed in 2006 [2, 26] and 2015 [19]. A celebratory set of reminiscences for Tony Hoare's 90th birthday appeared in *FACS FACTS* in July 2024 [7].

Since Tony Hoare's passing in March 2026, obituaries have appeared in the national newspapers, *The Guardian* [27] and *The Times* [29], and also in *Resurrection: The Bulletin of the Computer Conservation Society* [3]. "In memoriam" tributes have appeared in computer sciences journals such as the *Communications of the ACM* [8], *The Computer Journal* [5], and *Formal Aspects of Computing* [28]. Tony Hoare's effect on computer science has been evaluated by Bertrand Meyer in a *CACM* blog [22] and an in-depth technical evaluation for the period 1961–1977 has been published by Krzysztof Apt in the *Formal Aspects of Computing* journal [1]. No doubt, further retrospective insights will be published in the future.

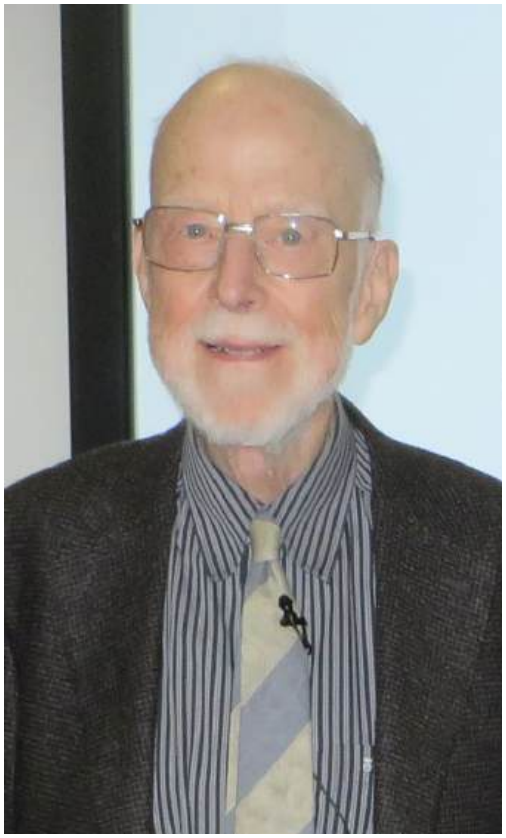


Figure 1: Tony Hoare during the 2018 celebration at the BCS London office for the 20th anniversary of his book, *Unifying Theories of Programming* [16]. (Photograph by Jonathan Bowen.)

As we look toward a future increasingly dominated by artificial intelligence and autonomous systems, the lessons of Tony Hoare are more relevant than ever. He reminded us that in the world of silicon and code, truth is not a matter of opinion (as espoused by some politicians) – it is a matter of fact and proof.

Requiescat in pace. Logica manet.

References

[1] Apt, K.R.: Tony Hoare: His path to the ACM Turing Award. Formal Aspects of Computing **38**(2) (2026). doi:10.1145/3816707

[2] Bowen, J.P., Hoare, C.A.R.: Oral history of Sir Antony Hoare. Tech. Rep. X3698.2007, Computer History Museum, California, USA (Sep 2006), **http://www.computerhistory.org/collections/catalog/102658017**, Catalog Number 102658017

[3] Bowen, J.P.: Obituary: Sir Tony Hoare (1934–2026). Resurrection: The Bulletin of the Computer Conservation Society **111**, 33 (Summer 2026), **https://www.computerconservationsociety.org/resurrection/pdfs/res111.pdf#page=35.0**

[4] Bowen, J.P., Fränzle, M., Olderog, E.R., Bjørner, D., Hansen, M.R., Langmaack, H., Liu, Z., Martin, U.: Experiences from the European ProCoS projects: Provably Correct Systems. Formal Aspects of Computing **38** (2026). doi:10.1145/3803555

[5] Crick, T.: In memoriam: C. A. R. Hoare. The Computer Journal **69** (May 2026). doi:10.1093/comjnl/bxag048

[6] Davies, J., Roscoe, A.W., Woodcock, J. (eds.): Millennial Perspectives in Computer Science: Proceedings of the 1999 Oxford–Microsoft Symposium in Honour of Sir Tony Hoare. Cornerstones of Computing, Palgrave (2000)

[7] Denvir, T., He, J., Jones, C.B., Roscoe, A.W., Stoy, J.E., Sufrin, B.A., Bowen, J.P.: Tony Hoare @ 90. FACS FACTS **2024**(2), 5–42 (Jul 2024), **https://www.bcs.org/media/1wrosrpv/facs-jul24.pdf#page=5.0**

[8] Garfinkel, S.L., Spafford, E.H.: In memoriam: C.A.R. Hoare. Communications of the ACM **69**(5), 26–28 (2026). doi:10.1145/3802605

[9] Hoare, C.A.R.: Algorithm 63: Partition; Algorithm 64: Quicksort; Algorithm 65: Find. Communications of the ACM **4**(7), 321–322 (Jul 1961). doi:10.1145/366622.366642

[10] Hoare, C.A.R.: Quicksort. The Computer Journal **5**(1), 10–15 (1962). doi:10.1093/comjnl/5.1.10

[11] Hoare, C.A.R.: An axiomatic basis for computer programming. Communications of the ACM **12**(10), 576–580, 583 (Oct 1969). doi:10.1145/363235.363259

[12] Hoare, C.A.R.: Communicating sequential processes. Communications of the ACM **21**(8), 666–677 (Aug 1978). doi:10.1145/359576.359585

[13] Hoare, C.A.R.: The emperor's old clothes. Communications of the ACM **24**(2), 75–83 (Feb 1981). doi:10.1145/1283920.1283936

[14] Hoare, C.A.R.: Communicating Sequential Processes. Series in Computer Science, Prentice Hall (1985)

[15] Hoare, C.A.R.: My early days at Elliotts. Resurrection: The Bulletin of the Computer Conservation Society **48**, 21–29 (Autumn 2009), **https://www.computerconservationsociety.org/resurrection/pdfs/res48.pdf#page=23.0**

[16] Hoare, C.A.R., He, J.: Unifying Theories of Programming. Series in Computer Science, Prentice Hall (1998)

[17] Hoare, C.A.R., Jones, C.B. (eds.): Essays in Computing Science. Series in Computer Science, Prentice Hall (1989)

[18] Hoare, C.A.R., Misra, J.: Verified software: Theories, tools, experiments vision of a grand challenge project. In: Meyer, B., Woodcock, J. (eds.) Verified Software: Theories, Tools, Experiments, First IFIP TC 2/WG 2.3 Conference, VSTTE 2005, Zurich, Switzerland, October 10–13, 2005, Revised Selected Papers and Discussions. Lecture Notes in Computer Science, vol. 4171, pp. 1–18. Springer (2005). doi:10.1007/978-3-540-69149-5_1

[19] Jones, C.B.: An interview with Tony Hoare: ACM 1980 A.M. Turing Award recipient. ACM (2015), **https://amturing.acm.org/pdf/HoareTuringTranscript.pdf**

[20] Jones, C.B., Misra, J. (eds.): Theories of Programming: The Life and Works of Tony Hoare. ACM (2021)

[21] Jones, C.B., Roscoe, A.W., Wood, K.R. (eds.): Reflections on the Work of C. A. R. Hoare. Springer (2010). doi:10.1007/978-1-84882-912-1

[22] Meyer, B.: Tony Hoare and his imprint on computer science. Communications of the ACM (20 March 2026), **https://cacm.acm.org/blogcacm/tony-hoare-and-his-imprint-on-computer-science/**, BLOG@CACM

[23] Numerico, T., Bowen, J.P.: Program verification and semantics: The early work. IEEE Annals of the History of Computing **24**(1), 90–92 (2002). doi:10.1109/MAHC.2002.988586

[24] Numerico, T., Bowen, J.P.: 25 years of CSP. FACS FACTS **2004**(3), 25–31 (Nov 2004), **https://www.bcs.org/media/3097/facts200411.pdf#page=25**

[25] Roscoe, A.W. (ed.): A Classical Mind: Essays in Honour of C.A.R. Hoare. Series in Computer Science, Prentice-Hall (1994)

[26] Shustek, L., Hoare, C.A.R., Bowen, J.P.: An interview with C.A.R. Hoare. Communications of the ACM **52**(3), 38–41 (2009). doi:10.1145/1467247.1467261

[27] Sufrin, B.A.: Sir Tony Hoare obituary. The Guardian (12 April 2026), **https://www.theguardian.com/technology/2026/apr/12/sir-tony-hoare-obituary**

[28] ter Beek, M.H., Johnsen, E.B.: Tony Hoare: In memoriam. Formal Aspects of Computing **38**(2) (2026). doi:10.1145/3816421

[29] Sir Tony Hoare obituary: Turing Award-winning computer scientist. The Times (28 April 2026), **https://www.thetimes.com/uk/obituaries/article/sir-tony-hoare-obituary-turing-award-winning-computer-scientist-6ml9k8lcq**

[30] Zhou, C., Hoare, C.A.R., Ravn, A.P.: A calculus of durations. Information Processing Letters **40**(5), 269–276 (1991). doi:10.1016/0020-0190(91)90122-X

Forthcoming Events

If you have suggestions for future FACS seminar speakers or other events, especially if you are willing to help with co-organisation or even give a talk, please contact Alvaro Miyazawa on Alvaro.Miyazawa@york.ac.uk.

Events Venue (unless otherwise specified):

BCS, The Chartered Institute for IT
Ground Floor, 25 Copthall Avenue, London, EC2R 7BP

The nearest tube station is Moorgate, but Bank and Liverpool Street are within walking distance as well. The new Elizabeth Line is now very convenient for the BCS London office, by alighting at the Liverpool Street stop and leaving via the Moorgate exit.

Details of all forthcoming events can be found online here:

https://www.bcs.org/membership/member-communities/facs-formal-aspects-of-computing-science-group/

Please revisit this site for updates as and when further events are confirmed.

FACS Committee

Formal Aspects of Computing Science Specialist Group

Keith Lines
FACS Chair

Roger Carsley
Treasurer

Alvaro Miyazawa
Secretary and Seminar Organiser

Jonathan Bowen
Newsletter Editor
(Emeritus FACS Chair and BCS Liaison)

John Cooke
Publications
(Emeritus Treasurer)

Margaret West
Inclusion Officer

Andrei Popescu
LMS Liaison

Ana Cavalcanti
FME Liaison

Tim Denvir
Emeritus Newsletter Co-editor

Brian Monahan
Emeritus Newsletter Co-editor

FACS is always interested to hear from its members and keen to recruit additional helpers. Presently we have vacancies for officers to help with fund raising, to liaise with other specialist groups such as the Requirements Engineering group and the European Association for Theoretical Computer Science (EATCS), and to maintain the FACS website. If you are able to help, please contact the FACS Chair, Keith Lines, at the contact points below:

BCS-FACS
c/o Keith Lines (Chair)
National Physical Laboratory, Teddington, TW11 0LW
Email: **klinesfacs@gmail.com**

You can also contact the other Committee members via this email address.

Mailing Lists

As well as the official BCS-FACS Specialist Group mailing list run by the BCS for FACS members, there are also two wider mailing lists on the Formal Aspects of Computer Science run by JISCmail.

The main list: **facs@jiscmail.ac.uk** can be used for relevant messages by any subscriber. An archive of messages is accessible under:

http://www.jiscmail.ac.uk/lists/facs.html

including facilities for subscribing and unsubscribing.

The additional list: **facs-event@jiscmail.ac.uk** is specifically for announcement of relevant events.

Similarly, an archive of announcements is accessible under:

http://www.jiscmail.ac.uk/lists/facs-events.html

including facilities for subscribing and unsubscribing.

BCS-FACS announcements are normally sent to these lists as appropriate, as well as the official BCS-FACS mailing list, to which BCS members can subscribe by officially joining FACS after logging onto the BCS website.