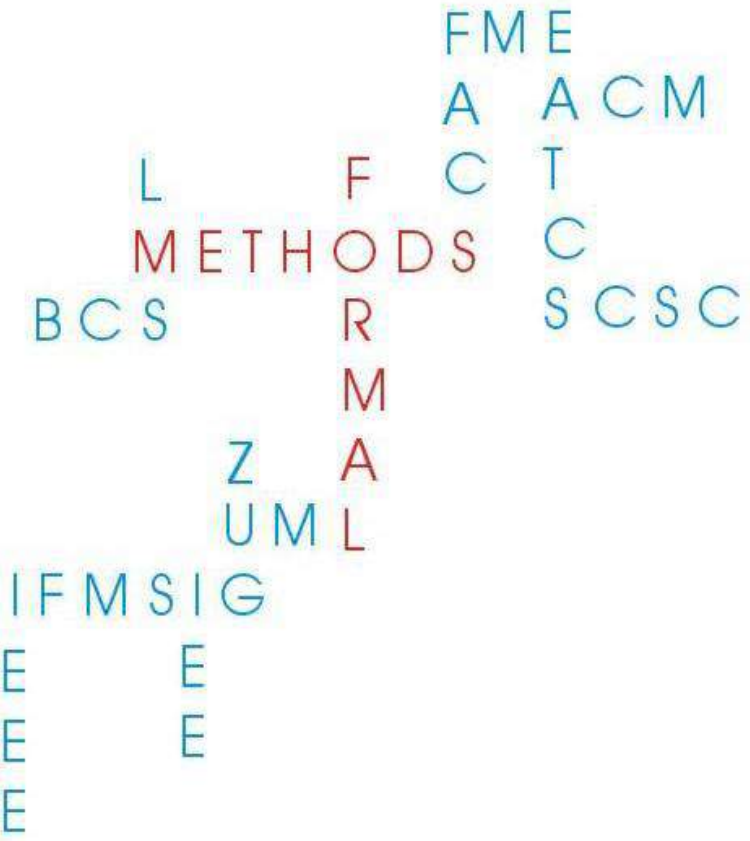
Issue 2020-1
June 2020

FACS ACTS



bcs The Chartered Institute for IT
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 Editors

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

Editorial Team:

Jonathan Bowen, John Cooke, Tim Denvir, Brian Monahan, Margaret West.

Contributors to this issue:

Jonathan Bowen, Tim Denvir, Dennis Furey, Sofia Meacham

BCS-FACS websites

BCS: http://www.bcs-facs.org
LinkedIn: https://www.linkedin.com/groups/2427579/
Facebook: http://www.facebook.com/pages/BCS-FACS/120243984688255
Wikipedia: http://en.wikipedia.org/wiki/BCS-FACS

If you have any questions about BCS-FACS, please send these to Jonathan Bowen on jonathan.bowen@lsbu.ac.uk.

Editorial

Dear readers,

Welcome to FACS FACTS 2020 issue 1. Because we are unable for the bulk of this year to hold seminars and meetings face to face, we are planning two issues of the newsletter in 2020. This will give those who would otherwise contribute a talk or seminar an opportunity to share their offerings in written form.

This first issue of 2020 begins with a report from Sofia Meacham on a seminar that was presented by François Siewe in February 2020, before the COVID-19 virus closed down meetings: *Privacy Assurance in Ubiquitous Systems by Typing in a Calculus of Context-Aware Ambients*.

That is followed by an abstract of a seminar on *Delay Insensitive Circuits* by Dennis Furey. This seminar, scheduled for 28th May 2020, had to be cancelled on account of COVID-19. This abstract contains two links, one to a 1-hour video of the intended talk, so readers can experience it on their screens as a YouTube presentation. It shows the slides with the speaker, Dennis Furey, to one side, a clear and graphic way of reproducing a seminar for later viewing and is well worth watching.

Sixty years ago saw the birth of Algol 60, with the publication of the Algol 60 Report. The syntax of the language was defined precisely with the BNF formalism, itself defined for that specific purpose, but a formal semantics was published by Peter Landin a few years later. Tim Denvir's paper, *Algol 60 @ 60*, marks this anniversary and briefly explores the influences of formality on the language.

Finally, Jonathan Bowen's paper *In Memoriam*, pays tribute to five of his colleagues in computer science who have passed away during the last five years: Robert France, Mike Gordon, Hussein Zedan, Anders Ravn and Sergiy Vilkomir. Note that Hussein Zedan was the PhD supervisor of François Siewe whose seminar is the subject of Sofia Meacham's report in this newsletter.

There is nothing further to report on our project to digitise past copies of the FACS newsletters. All the issues we have been able to lay our hands on have been processed and are available to read or download at https://www.bcs.org/membership/member-communities/facs-formal-aspects-of-computing-science-group/newsletters/

Future Events: owing to COVID-19 we cannot schedule any physical FACS meetings for the moment, but we fully intend to hold our traditional joint meeting with the LMS and the annual Peter Landin Semantics seminar towards the end of the year, in one form or another. Watch for email announcements nearer the time.

Tim Denvir, *FACS FACTS* co-editor

Privacy Assurance in Ubiquitous Systems by Typing in a Calculus of Context-Aware Ambients

François Siewe

De Montfort University

27th February 2020

Reported by: Sofia Meacham

The talk started with a definition of the ubiquitous computing systems, their relation to the Internet of Things and the focus of this talk on modelling and reasoning about the behaviours of these systems.

Specifically, an outline of the talk consisted of an overview of ubiquitous computing, modelling using process calculi, privacy using typed systems and a case study.

For the definition of ubiquitous systems, the speaker used a quote from Mark Weiser, 1991 who claimed that the user should not tell the computer what to do but the computer should be clever enough to figure out what the user wants and provide it as a service. Also, according to Mark Weiser, we don't need to see the computer as it is "distributed" in the environment.

Then the speaker continued with a historical overview of how computing evolved from mainframes in 1940s (one computer many users) to personal computers 1965 (one computer one user) to ubiquitous computing 1990s (many computers one user).

Weiser identified three main properties of ubiquitous computing: distributed, implicit human-computer interaction, context-awareness. In this talk, context awareness was the main theme.

Internet of Things is the latest term for ubiquitous computing and includes many applications from smart homes to smart vehicles and consequently smart cities.

The question "why modelling these systems" was the next main theme of the talk. The elements of abstraction, simulation and property proving for handling the complexity of these systems and potential solutions such as network simulation tools, ontologies, Petri nets, etc were presented. The use of process calculi was identified as the solution proposed by this research as appropriate to the problem.

The calculus developed was called CCA that stands for Calculus of Context-aware Ambients and its main characteristics include concurrency, context-awareness and mobility. The rest of the talk focused on how we can use the proposed CCA to model ubiquitous computing systems with emphasis on their context-awareness.

The introduction of the ambient as a conceptual representation of anything was presented. An ambient can be a house, a wireless access point, etc.

There are two main elements that are used to represent an ambient: a name and its behaviour. With this simple notation, the speaker tried to prove that you can describe any system with examples including elements such as mobility. The communication mechanisms that are supported were also detailed such as local communication inside an ambient, parent to child, between siblings and how these types of communication can be used for specific or any targets.

The capabilities of the ambient and the corresponding grammar was then introduced accompanied by various examples such as RFID and RFID reader system, actuator device (lamp), access to a garage.

It is very important to note that extensive tool support has been built around the CCA formalism. Simulation capabilities and a translator to SPIN model checker have been implemented offering the complete range of tools to formally analyse these systems.

Delay Insensitive Circuits

(Abstract)

Dennis Furey

Plumstead Publishing

An exercise in formal methods

The video lecture prepared in lieu of the May meeting of the FACS special interest group begins with a simple example demonstrating the difficulty of establishing the correctness of a delay insensitive circuit by informal arguments, and motivating a more formal treatment.

A theoretical framework is sketched in three parts: a general purpose algebraic method to express networks of uninterpreted blocks with input and output ports, a way of modelling the primitive components of delay insensitive circuits using Petri nets, and a compositional semantics specifying the effect of multiple components combined in a network expressed algebraically as proposed. In addition, a structure preserving map from the algebraic form to a more straightforward netlist form is described for the sake practical circuit fabrication.

Using concise recurrences in a functional programming style, two worked examples demonstrate the application of this approach to common circuit configurations: a binary tree and a more complex structure implementing a family of arbiters parameterized by the number of ports. Suggested further reading brings the presentation to a close.

References

https://www.delayinsensitive.com/news.html

https://www.youtube.com/watch?v=P4FuFPzxtdo

Algol 60 @ 60

Tim Denvir

Translimina

May 2020

Algol 60 is 60 years old

Algol 60, an inspiration for many languages which followed it, Pascal, C, Simula, Modula, Java and others, was very carefully defined. A series of meetings and conferences, some by invitation, starting with an informal meeting in 1958, published first a preliminary report and then a sequence of drafts culminating in the definitive "Revised Report on the Algorithmic Language Algol 60". This last resulted from an international conference in January 1960 in Paris. There, thirteen representatives from Denmark, France, Germany, the Netherlands, Switzerland, the UK and the USA worked on the previous drafts and produced what is now generally known as the Algol 60 Report. The delegates represented the BCS and corresponding organisations from the other countries, "learned societies" for computer science, as the BCS was considered in those days. Vigorous mechanisms for consultation were set up and a regular publication, the *Algol Bulletin,* produced, initially from 1959 in Copenhagen and continuing for 52 issues until 1988, all under the auspices of IFIP WG 2.1. The *Algol Bulletin* became the forum for discussion on the design of the language. A subsequent conference in 1962 corrected known errors, attempted to eliminate apparent ambiguities and otherwise clarified the Algol 60 Report. Peter Naur, the Danish member of the 13-member team, was the driving force behind the *Algol Bulletin*, and the editor of the Report.

There seems to have been some debate about whether to include recursion, which was won, some say, by sleight of hand [1]. The concept of recursion is certainly to be found in mathematical logic, and all the team members had a strong mathematical training and background. They were mature, highly experienced scientists, their birth dates ranging from 1916 to 1929, and versed in a variety of disciplines: mathematics, mathematical logic, astronomy, physics, formal languages; several were deeply involved in the design and construction of early computers. And after their time spent defining Algol 60, the designers moved on to functional programming, natural language translation, fingerprint recognition, computational linguistics, and program language design.

The rigorous care and attention to detail in the enterprise was not confined to organisation and administration. Reading the report today, it is clear that the whole project was guided by formal considerations, even if there was no specific formal semantics in the definition. The main part of the report is in five sections, the first being titled “Structure of the Language”. This is in fact a description of BNF, which is subsequently used to define the syntax of the whole language. BNF was pretty new at the time, indeed was, I believe, first described and used in the Preliminary Report published in the *Communications of the ACM* in 1958. BNF, Backus Naur Form, was based on work by Noam Chomsky aimed at describing the syntax of human languages. And indeed its originators, John Backus and Peter Naur, are among the thirteen authors of the definitive Revised Report. Later Peter Naur wanted to dissociate himself from BNF and renamed it “Backus Normal Form”. Being a rather new concept at the time, BNF, with its inherent recursive property, is described in the Report with great care and clarity.

The subsequent chapters of the Report define Algol 60 itself, in an essentially bottom-up approach, starting with basic symbols, identifiers, numbers and strings. The rest is built up from there, through expressions, statements, and declarations. The semantics are defined with careful English, mostly in an operational style. For example, in section 3.3.3 we have: “An arithmetic expression is a rule for computing a numerical value. In case of simple arithmetic expressions this value is obtained by executing the indicated arithmetic operations on the actual numerical values of the primaries of the expression, as explained in detail in section 3.3.4 below.” And again: “In the more general arithmetic expression, which include **if** clauses, one out of several simple arithmetic expressions is selected on the basis of the actual values of the Boolean expressions. This selection is made as follows: the Boolean expressions of the clauses are evaluated one by one in the sequence from left to right until one having the value true is found, the value of the arithmetic expression is then the value of the first arithmetic expression following this Boolean (the largest arithmetic expression found in this position is understood).” This subsection 3.3.3 is titled “Semantics”, as are subsequent subsections relating to other language elements. The use of the word in a computer language context was not common at the time and indicates, I feel, an appreciation of the distinction: other contemporary high-level language definitions often defined only the syntax and pretty much left the semantics up to the reader's intuition.

However, there are allusions to more than operational semantics in the Report. There are parts of the explanation of Procedure Statements which suggest to me a knowledge of and influence from term rewriting and, possibly, lambda calculus. To quote two subsections from 4.7:

> **4.7.3.2 Name replacement (call by name).** Any formal parameter not quoted in the value list is replaced, throughout the procedure body, by the corresponding actual parameter, after enclosing this latter in parentheses wherever syntactically possible. Possible conflicts between identifiers inserted through this process and other identifiers already present within the procedure body will be avoided by suitable systematic changes of the formal or local identifiers involved.
>
> **4.7.3.3 Body replacement and execution.** Finally the procedure body, modified as above, is inserted in place of the procedure statement and executed. If the procedure is called from a place outside the scope of any non-local quantity of the procedure body the conflicts between the identifiers inserted through this process of body replacement and the identifiers whose declarations are valid at the place of the procedure statement or function designator will be avoided through suitable systematic changes of the latter identifiers.

Call by name seemed at the time to have a certain purity of thought about it, but in practice was difficult to implement and enabled the possibility of some bizarre and opaque programming. And, of course, one may legitimately complain about the cavalier references to “suitable systematic changes” to identifiers and the like; they could potentially cover a lot of difficulties.

A formal semantics for Algol 60 was published by Peter Landin in two parts in 1965 [2, 3], a few years after the Report. In Part I he writes “Anyone familiar with both Church's λ-calculi and Algol 60 will have noticed a superficial resemblance...”, suggesting that the idea of such a correspondence was present in the beginning, even though Peter Landin was not one of the original Algol 60 team of thirteen. In fact this two-part paper was based on some lectures that George Couloris invited Peter Landin to give in the spring of 1963, in a series on “Logical Foundations of Programming” at the “University of London Computer Unit”, which was probably referring to ULICS, the University of London Institute of Computer Science in Gordon Square, in London's Bloomsbury. Further, Peter Landin presented a talk at an IFIP working conference in Baden in September 1964, “A Formal Description of Algol 60”, so

it seems that there was an impetus to find a formal semantics of Algol soon after its publication.

A few quotes from [2] and [3] may give a flavour of Peter Landin's approach: “Some of the semantics of Algol 60 can be formalised by establishing a correspondence between expressions of Algol 60 and expressions in a modified form of Church's λ-notation”; “Formal syntax has been used to practical advantage by language designers and implementers. There might be analogous advantages in formal semantics” (in motivation for formal semantics in the first place); “Many features of Algol 60, including call by name, declarations, **own**,... have parallels in a language that lacks assignment and hence [is] not essentially... 'imperative'”.

Landin's paper [2, 3] makes much use of terms defined in a previous paper on the mechanical evaluation of expressions [4], thus obliging the reader to have at least a passing knowledge of that one too! He continues to describe the semantics in an operational way in terms of an abstract machine, albeit a machine that interprets a language similar to λ-notation into which an “abstract Algol” has been translated. I think this may reflect an intrinsic struggle in Algol: it tries to be simultaneously an abstraction of what computers can imperatively do, and an expression of aspects of mathematics that can be evaluated by a computing machine. Peter Landin's conclusion summarises: “An abstract language based on Church's λ-notation and an abstract machine that interprets it has been described”. But Landin's “Abstract Algol” is, in his words, “over-tolerant”, “in that a mismatch will not lead to a rejection if the procedure is never applied, or if it is exited unnaturally and thus evades producing a result”. This seems to me rather nonchalant: a compiler built to his model would allow programs – that would run and produce results – which would be legitimately rejected by other 'correct' compilers. Not good for portability! (c.f. my previous FACS Newsletter article “Dangers of Over-Design”, which elaborates the problematic nature of this syndrome).

Furthermore, in [2] he writes: “Declarations are considered as giving initial values to the local identifiers. For instance, integer and real identifiers are initialised respectively to integer zero and real zero. A Boolean is initialised to **false**”. This could clearly radically alter the progress of an execution. But perhaps we should in our turn be tolerant. A domain-theoretic approach to program language semantics, which gives a convenient meaning to undefined values, was not to be conceived for some further eight or more years [5].

The introduction to the Algol 60 Report describes the history of the development of the language definition. The account of the January 1960 Paris conference ends with the sentence: “The present report represents the union of the Committee's concepts and the intersection of its agreements”, reflecting the authors' grounding in set theory!

A final word: the Report starts with a quote from Ludwig Wittgenstein:

> Was sich überhaupt sagen läßt, läßt sich klar sagen; und wovon man nicht reden kann, darüber muß man schweigen.

The Report does not attribute the quote to a particular work of Wittgenstein, but the second clause of the quote is almost identical to the last sentence of his *Tractatus*, and the first clause seems to be a paraphrase of 4.116 from the same work. Translating the first clause in the spirit of C. K. Ogden, the official translator of the *Tractatus*, into English, and following that with Ogden's actual translation of the last sentence, one has:

> What can be said at all, can be said clearly; and whereof one cannot speak, thereof one must be silent.

I have often thought that some of the best computer scientists reveal an influence from Wittgenstein, especially from his *Tractatus Logico-Philosophicus* [6]. It starts with the following:

> The world is everything that is the case. The world is the totality of facts, not of things. The world is determined by the facts, and by these being *all* the facts.

The “world” as seen by a point of control in a computer program is the set of all the variables that are in the scope of that point, and the state of this world is the value of those variables at that dynamic point. One might say that a principle of programs as predicates equates a state of a program's world with the set of all true propositions about it, that is the “facts” or “everything that is the case”.

But perhaps it would be whimsical to conclude that the authors of the Algol 60 report were presaging the concept of programs as predicates. Or perhaps not?

References

[1] Maarten van Emden: How recursion got into programming: a tale of intrigue, betrayal, and advanced programming-language semantics. A Programmer's Place, 18 Jun. 2014. https://vanemden.wordpress.com/2014/06/18/how-recursion-got-into-programming-a-comedy-of-errors-3/

[2] P. J. Landin: A Correspondence Between Algol 60 and Church's Lambda-Notation Part I. CACM Vol. 8 No. 2 Feb. 1965.

[3] P. J. Landin: A Correspondence Between Algol 60 and Church's Lambda-Notation Part II. CACM Vol. 8 No. 3 Mar. 1965.

[4] P. J. Landin: The mechanical evaluation of expressions. Computer Journal Vol. 6 No. 4, Jan. 1964, 308-320.

[5] D. Scott: Data Types as Lattices, SIAM Journal on Computing, Vol. 5, 1976, pp 522-587.

[6] Ludwig Wittgenstein: *Tractatus Logico-Philosophicus*. Routledge and Kegan Paul, 1922.

In Memoriam: A tribute to five formal methods colleagues

Jonathan P. Bowen

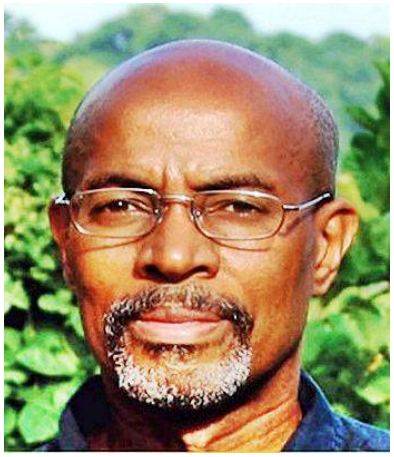
London South Bank University, UK
Southwest University, Chongqing, China

May 2020

It is a sad fact of aging that one's colleagues start to pass away with the passing years. But for some, this is before their time and this "In Memoriam" article is a tribute to five such colleagues of mine who passed away during 2015 to 2020. It is both a brief record of their achievements, largely based on their Wikipedia pages, and a personal reminiscence recording some memories of and collaborations with each of them. I do not feel too bad about borrowing from the former, since I was the original creator for the Wikipedia articles for all five of them, my own small, more public, and long-lasting tribute to them.

Robert France (1960–2015)

Robert Bertrand France (8 October 1960 – 15 February 2015) was a Jamaica-born American computer scientist.



Robert B. France was born in Jamaica on 8 October 1960, the eldest son of Robert W. and Jeanette France. He attended high school in Guyana and studied for a BSc degree in Natural Sciences at the University of the West Indies in Saint Augustine, Trinidad and Tobago, majoring in Computer Science and Mathematics and receiving a first class degree in 1984. He then attended Massey University in New Zealand funded by a Commonwealth Scholarship, where he achieved a PhD degree in computer science in 1990. During the same year, he married Sheriffa R. Soleyn in Saint Vincent. They emigrated to the United States together and in due course moved to Fort Collins, Colorado.

During 1990–92, France was a Postdoctoral Research Associate at the Institute for Advanced Computer Studies, University of Maryland. From 1992–97, he was an assistant professor in the Computer Science and Engineering Department at Florida Atlantic University (FAU), becoming tenured in 1997–98. France was then appointed an associate professor from 1998 until 2004 and then full professor at Colorado State University within the Department of Computer Science. He undertook research on model-driven software development (France & Rumpe, 2007), especially concerning formal software modelling languages and associated analysis tools. He was co-founder and Editor-in-Chief of the *Software and Systems Modeling* journal from 1999 until 2015.

In 2008, Robert France and his co-authors Andy Evans, Kevin Lano, and Bernhard Rumpe, were awarded the *Ten Year Most Influential Paper Award* at the MODELS 2008 Conference on *Model Driven Engineering Languages and Systems* for the 1998 paper "*The UML as a Formal Modeling Notation*" (Evans et al., 1998). In 2013, France was awarded a five-year International Chair at INRIA in France. He was awarded a senior Dahl–Nygaard Prize for his research by the Association Internationale pour les Technologies Objets (AITO) in 2014. In the same year, he was awarded a Colorado State University, College of Natural Sciences Professor Laureate and an Excellence in Science and Technology Award from the Institute of Caribbean Studies.

Robert France died on 15 February 2015 at the age of 54. He had a son and daughter with his wife, Sheriffa.

Reminiscence

One of the earliest conferences at which I presented a published paper was the Second IEE/BCS Conference on Software Engineering in Liverpool, UK (Bowen, 1988). Robert had a formal methods paper published in the same proceedings while he was studying for his PhD at Massey University in New Zealand (France & Docker, 1988). Later I knew Robert through my colleagues Peter Breuer (Breuer et al. 1999) and Kevin Lano (Evans et al., 1998), both of whom worked with me on the European ESPRIT II REDO project (1989–1992) concerning software maintenance and also collaborated with Robert subsequently.

Much later at a conference in April 2010, as co-author of a paper on Bletchley Park (Black, Bowen & Griffin, 2010), I was marooned in Denver, Colorado, USA, after the Icelandic volcano Eyjafjallajökull erupted and prevented air travel in European airspace for about a week. I decided to drive up to Yellowstone Park,

which I had always wanted to visit. Colorado State University in Fort Collins was conveniently on the way, so I contacted Robert and we arranged to meet for lunch. The meal was a very pleasant affair and was provided by students studying on hospitality-related courses. I wondered if we would have to supply marks at the end of the laboratory session! Sadly, this would be the last time I would see Robert, but it is a very happy memory of a very aimable colleague.

Mike Gordon (1948–2017)

Michael John Caldwell Gordon FRS (28 February 1948 – 22 August 2017) was a leading British computer scientist.

Mike Gordon was born in Ripon, Yorkshire, England. He attended Dartington Hall School and Bedales School. In 1966, he was accepted to study engineering at Gonville and Caius College, University of Cambridge, but transferred to mathematics. During his studies, in 1969, he worked at the National Physical Laboratory in London during the summer, gaining his first exposure to computers.

Gordon studied for his PhD degree at University of Edinburgh, supervised by Rod Burstall, finishing in 1973 with a thesis entitled *Evaluation and Denotation of Pure LISP Programs*. He was invited to Stanford University in California by John McCarthy, the inventor of LISP, to work in his Artificial Intelligence Laboratory there. Gordon worked at the Cambridge University Computer Laboratory from 1981, initially as a lecturer, promoted to Reader in 1988 and Professor in 1996.

Gordon led the development of the HOL theorem prover (Gordon & Melham, 1993). The HOL system is an environment for interactive theorem proving in a higher-order logic. Its most outstanding feature is its high degree of programmability through the meta-language ML. The system has a wide variety of uses, from formalising pure mathematics to verification of industrial hardware.

There has been a series of international conferences on the HOL system, TPHOLs. The first three were informal users' meetings with no published

proceedings. The tradition then became for an annual conference in a continent different from the location of the previous meeting. From 1996, the scope broadened to cover all theorem proving in higher-order logics.

Gordon was elected a Fellow of the Royal Society in 1994, and in 2008 a two-day research meeting on *Tools and Techniques for Verification of System Infrastructure* was held there in honour of his 60th birthday.

Mike Gordon was married to Avra Cohn, a PhD student of Robin Milner at the University of Edinburgh, and they undertook research together. He died in Cambridge aged 69 after a brief illness and is survived by his wife and two sons.

Reminiscence

I first collaborated with Mike on the UK Information Engineering Directorate (IED) SAFEMOS project (1989–1993) on "Totally Verified Systems", a collaboration between Oxford University (with Tony Hoare), Cambridge University (with Mike and John Herbert), Cambridge SRI (with Roger Hale), and Inmos in Bristol (with David Shepherd) (Bowen et al. 1994). By the end of the project, we had modified our sights recorded in a book more realistically entitled "*Towards Verified Systems*" (Bowen, 1994). Mike contributed chapters based on HOL (Higher Order Logic) (Gordon & Pitts, 1994; Gordon, 1994). We also collaborated on a shallow embedding of the Z notation in the HOL system (Bowen & Gordon, 1994; 1995). Mike with his characteristic generosity insisted on alphabetic ordering of authors, whereas in reality he had done most of the work. He presented the paper as a keynote speaker at the 1994 Z User Meeting (Bowen & Hall, 1994), held in Cambridge, UK.

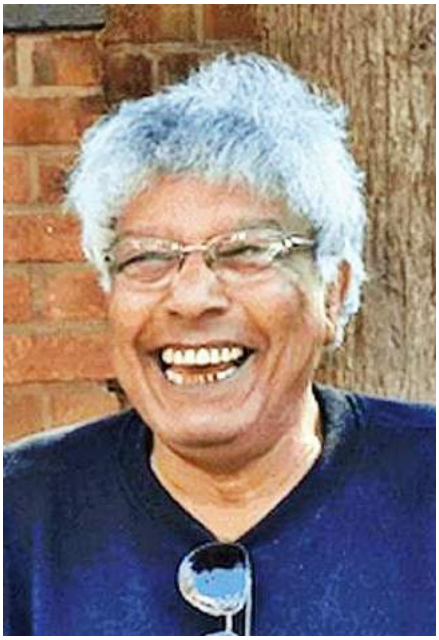
Subsequent to the SAFEMOS project, Mike was very kind as a referee for me. At a traumatic stage of my career, when there were many redundancies of computer science research academics at London South Bank University (LSBU) in 2006, due to the delayed effects of the dot-com crash on student numbers, Mike wrote a very touching letter of support. It is at times like this that you know who your friends are. In the end, this proved to be a fortuitous change of direction for me, with early retirement possible, enabling opportunities to pursue some wider research goals and other pursuits in industry and internationally, without the constraining commitments of teaching and administration. We continued to maintain contact by email and even Facebook, my only Fellow of the Royal Society "friend" there I think! In 2009, Mike gave a

characteristically clear talk for the BCS Formal Aspects of Computer Science (FACS) Specialist Group (for which I have been chair and treasurer), in collaboration with the London Mathematical Society (LMS) at their headquarters. In a final email exchange on the EU referendum petition to the UK Government in 2016, he replied "I already signed it! – Mike". At least he was spared the chaos that was to follow.

After Mike's passing, there were events in Oxford organised by his colleague Tom Melham (which sadly I missed because I was out of the country at the time, even though it was held a few minutes' walk from where I live) and Cambridge, which I did attend, celebrating Mike's life and work. The Cambridge event was admirably hosted at the Cambridge University Computer Laboratory by Larry Paulson and a fitting academic tribute to Mike, attended by his wife and colleague Avra Cohn.

Hussein Zedan (1953–2019)

Hussein S. M. Zedan (1 July 1953 – 23 February 2019) was a computer scientist of Egyptian descent, mainly based in the United Kingdom.



Hussein Zedan was born in 1953. He received his PhD degree in 1981 at the University of Bristol, studying under John Derwent Pryce and Hubert Schwetlick for a thesis entitled *Modified Rosenbrock-Wanner methods for solving systems of stiff ordinary differential equations*.

Zedan was an academic in the Department of Computer Science at the University of York. Prof. Zedan then headed the Software Technology Research Laboratory (STRL) as Technical Director at De Montfort University. He was also Head of Computing Research. Later STRL was headed by Zedan's PhD student and subsequently colleague François Siewe. Zedan was subsequently appointed Assistant Vice-President of Academic Affairs and Development at the Applied Science University in Manama, Bahrain, until 2017.

Hussein Zedan died on 23 February 2019, at the age of 65. He was married with two daughters.

Reminiscence

My first collaboration with Hussein was through a contributed chapter on distributed operating systems (Bowen & Gleeson, 1990) in an edited book on *Distributed Computer Systems* (Zedan, 1990a). In the same year, Hussein was guest editor of a special journal issue of *Microprocessors and Microsystems* on *Formal Aspects of Microprocessor Design* (Zedan, 1990b). I contributed a paper on the instruction set to be used on the European ProCoS and UK SAFEMOS collaborative projects that ran in parallel, a cutdown version of the Inmos transputer, specified using the Z notation (Bowen, 1990). Hussein continued with ProCoS project collaboration, in particular with my Oxford University colleague He Jifeng on refinement of real-time systems (Scholefield, Zedan & He, 1994). He also participated in the subsequent ProCoS-WG Working Group (see the ProCoS-WG group photograph at the end of this article).

Later we both participated in the UK EPSRC FORTEST Network (2001–04) on Formal Methods and Testing (Bogdanov et al., 2003; Hierons et al., 2009). During this period, a number of PhD students joined me at London South Bank University (including Sergiy Vilkomir, see later). Francois Siewe, who I knew from the United Nations University International Institute for Software Technology (UNU-IIST) in Macau, had been accepted for a PhD at LSBU. Sadly, LSBU changed direction following the dot-com crash, and withdrew funding for Francois before he was able to start his PhD. More happily, Hussein, now at De Montfort University (DMU), was able to secure funding for Francois to study there. This collaboration proved very productive from the start (Siewe, Cau & Zedan, 2003), so much so that Francois is now a Reader at DMU and is Head of the Software Technology Research Laboratory (STRL), which Hussein founded there. I continued to visit Hussein there as an external examiner, as a speaker, and also collaborating on publications (Bowen et al., 2014; 2018).

Since 2007, I have been involved with accreditation of computing-related degree programmes in the Middle East, including Bahrain, Oman, Saudi Arabia, and the UAE. Hussein moved to the Applied Science University in Manama, Bahrain, where I was able to aid in a practice accreditation visit for the university in 2016. We had an enjoyable time together, including a lively meal with the head of the university. Hussein was his usual jovial and avuncular self, still calling me “kid” after all these years! Very sadly, this was the last time I was to meet Hussein, although he was able to retire back to the UK in 2017. His last

email to me in 2018 concerning our final joint publication was characteristically friendly, ending with “Keep well and happy – Best, h”.

Anders Ravn (1947–2019)

Anders Peter Ravn (29 October 1947 – 1 August 2019) was a Danish computer scientist.

Anders P. Ravn was born in 1947 in Caracas, Venezuela, the son of Niels and Henny (Sønder) Ravn. He arrived in Denmark in 1948. Ravn received a Master of Science (M.Sc.) degree in Computer Science and Mathematics from the University of Copenhagen in 1973 and a Doctor of Technology (Dr.Tech.) degree in Computer Science from the Technical University of Denmark in 1995.

Between 1969–73, Anders Ravn was a teaching assistant in the Department of Computer Science at the University of Copenhagen (DIKU). From 1972–76, he was a systems programmer on minicomputers at the early Danish computer company A/S Regnecentralen. He returned to academia and rose from assistant professor (1976–80) to associate professor (1980–84) at DIKU. During 1982–83, he was a visiting scientist at IBM‘s Thomas J. Watson Research Center in Yorktown Heights, New York, United States. He joined the Department of Computer Science at the Technical University of Denmark (ID-DTH) as a lecturer (1984–89) followed by reader (1989–99) in the Department of Information Technology. During this time, he was also an adjunct lecturer in the Department of Mathematics at the Royal Veterinary and Agricultural University in Denmark (1985–89), guest researcher at Oxford University (1989–90), and Visiting Professor at the Institut für Praktische Mathematik und Informatik, University of Kiel in Germany (1994). He participated in the ESPRIT ProCoS project on Provably Correct Systems. He was a Research Professor in the Department of Computer Science at Aalborg University (1999–2004), before being appointed Professor of Computing Science there (2004–16). He then retired, becoming an Emeritus Professor.

Ravn specialized in research into formal methods, especially for embedded systems and hybrid systems (Grossman, 1993; Ravn & Rischel, 1998; Liu &

Ravn, 2009). In particular, he worked with Zhou Chaochen and Tony Hoare on the development of Duration Calculus for real-time systems (Zhou, Hoare & Ravn, 1991).

He became a member of the Dansk Selskab for Datalogi (DSfD) in 1972, the Association for Computing Machinery (ACM) in 1983, and was a Life Member of the IEEE Computer Society. He was also a member of the IFIP Working Group 2.2 (covering the Formal Description of Programming Concepts) from 2002.

Anders Ravn received Ulrik and Marie Brinch's honorary award in 1996. He was honoured as a Knight of the Order of the Dannebrog in 2015. He married Annemette Lind on 31 August 1973 and they had two children. Ravn died in Copenhagen on 1 August 2019, at the age of 71.

Reminiscence

I first met Anders through the European ESPRIT II ProCoS project (Bjørner et al., 1989) on "Provably Correct Systems" (1989–1991). He was then at the Technical University of Denmark (DTH), with Dines Bjørner and others. We held a number of joint workshops for members of the project, notably on the island of Bornholm in the Baltic Sea, south of Sweden. Members of the project travelled by boat from Copenhagen to attend. As Dines Bjørner noted much later (Bjørner, 2017), after a talk by E.V. Sørensen at the workshop, he saw Anders, Tony Hoare, and Zhou Chaochen in the break discussing ideas that appeared to be the initial thoughts about what was to become Duration Calculus (Zhou, Hoare & Ravn, 1991), arguably the most influential although unpredicted development to result directly from the ProCoS project.

ProCoS activities continued with Anders' involvement in the ProCoS II project (1992–1995). Anders, I, and others provided an overview of the ProCoS project (Bowen et al., 1993) and tutorial material on the project's research (Ravn et al., 1993; Bowen et al., 1994). Anders and Hans Langmaack of the Christian-Albrechts-Universität zu Kiel also provided another overview of the ProCoS project for the book on the related SAFEMOS project (Langmaack & Ravn, 1994). ProCoS activities continued after the end of the ProCoS II project (Bowen et al., 1996) with the ProCoS-WG Working Group (1994–1997) (Bowen et al., 1998). See a photograph including Anders at the final 1997 ProCoS-WG meeting below.

In the intervening years, we were to meet in a slightly different capacity as reviewers of research project proposals for the Academy of Finland in a wintery

Helsinki. Anders was his usual friendly self, always with that slight smile, tilted head, and thoughtful words in any informal conversation.

In 2015, I co-organized a two-day ProCoS workshop at the BCS London offices in Covent Garden, celebrating a quarter of a century since the original ProCoS project (Bowen, 2016a; Hinchey, Bowen & Olderog, 2017). Anders attended, accompanied by his wife Annemette, and gave a presentation, sadly the last time I would see him. He later had a joint paper in published in the proceedings (Olderog, Ravn & Wisniewski, 2017).

Jonathan Bowen and Anders Ravn during the 2015 ProCoS workshop in London. (Photograph by Xiaohong Chen.)

Sergiy Vilkomir (1956–2020)

Sergiy A. Vilkomir (19 November 1956 – 9 February 2020) was a Ukrainian-born computer scientist.

Sergiy Vilkomir was born in 1956 in Ukraine. He finished Mathematical College at the Moscow State University National Mathematical Boarding High School no. 18 (Head-Academician A. Kolmogorov, 1972–74), studied for an MSc degree in Mathematics and Mathematics Education at Kharkov State University (1974–79), and for a PhD degree at

Kharkov Polytechnic Institute (1985–90). In Kharkiv, Ukraine, he then worked at the Ukrainian Polytechnic Institute (1979–82), the Central Institute of Complex Automation (1985–91), the Institute of Safety and Reliability of Technological Systems (1992–93), the Ukrainian State Scientific and Technical Centre on Nuclear and Radiation Safety (part of the Nuclear Safety Regulatory Authority of Ukraine, 1993–2000). His role included licensing and audits of computer-based safety systems at nuclear power plants.

In 2000, Vilkomir moved to the Centre for Applied Formal Methods at London South Bank University, becoming a Research Fellow there. He then joined the University of Wollongong in Australia, also as a Research Fellow. He subsequently worked with David Parnas at the University of Limerick in Ireland, before moving to the United States, initially as Research Associate Professor and the University of Tennessee during 2007–8, then rising to be an associate professor position at East Carolina University, which he joined in 2008. There he achieved academic tenure in 2012 and was Head of the Software Testing Research Group (STRG).

Vilkomir's main research contributions have been in the formalization of software testing. In particular, he proposed reinforced condition/decision coverage (RC/DC), a stronger version of the modified condition/decision coverage (MC/DC) coverage criterion for software testing in safety-critical systems.

Vilkomir was awarded the Google Faculty Research Award for 2010–11, the East Carolina University Scholar-Teacher Award in 2015, and the UNC Board of Governors Distinguished Professor of Teaching Award in 2017. He was a Senior Member of both the Association for Computing Machinery (ACM, from 2013) and the IEEE.

Sergiy Vilkomir died on 9 February 2020, at the age of 63. He was married to Tetyana Vilkomir.

Reminiscence

I first met Sergiy when I moved to London South Bank University and established the Centre for Applied Formal Methods (CAFM) there in 2000. Sergiy, then based in Ukraine, applied for a PhD studentship as a mature student and was very quickly productive in his research (Vilkomir & Bowen, 2001a–c; 2002a–c), combining his experience in the nuclear industry with

formal aspects of testing, especially through the collaborative UK EPSRC FORTEST Network on Formal Methods and Testing. This led in particular to the formulation using the Z notation of a new software testing criterion, RC/DC, a stronger version of the existing MC/DC (Vilkomir & Bowen, 2006; 2008). Sergiy was promoted to a research position within CAFM and was my righthand man in the centre. He collaborated with Kalpesh Kapoor, one of my PhD students at CAFM (Vilkomir, Kapoor & Bowen, 2003). Sergiy also contributed to the final survey paper produced by the FORTEST Network (Bogdanov et al., 2003; Hierons et al., 2009) and a chapter for the book produced by the network (Vilkomir & Bowen, 2008).

Sadly, as previously mentioned, a change of direction at LSBU caused by falling student numbers in computer science after the dot-com crash, meant that university-funded research posts in computer science were terminated. Sergiy helped to produce a strong ESPRIT research proposal, which narrowly missed being funded. However, I was very glad that he was successful in applying for research positions, first in Australia (Vilkomir, Bowen & Ghose, 2006) and later in Ireland with David Parnas (Baber, Parnas, Vilkomir et al., 2005). Subsequently, he gained a Green Card and achieved tenure in the USA, heading his own research group there. I was very happy to see his academic career flourish. We continued in touch on email periodically and he acted as a reviewer for the *ACM Computing Surveys* journal, which I serve as a member of the Editorial Board. Unfortunately, I never had the opportunity to meet Sergiy again after he left the UK, although he was in my thoughts and I was very sad to hear from his wife Tanya about his passing. It brought back a happy and somewhat alcoholic memory of a Ukrainian-style meal at their flat in London.

Epilogue

I have been privileged to know and collaborate with all of these five colleagues. A complete bibliography of publications with these and other colleagues can be found in Bowen (2019). Thanks are due to Wikipedia for some of the text and photographs.

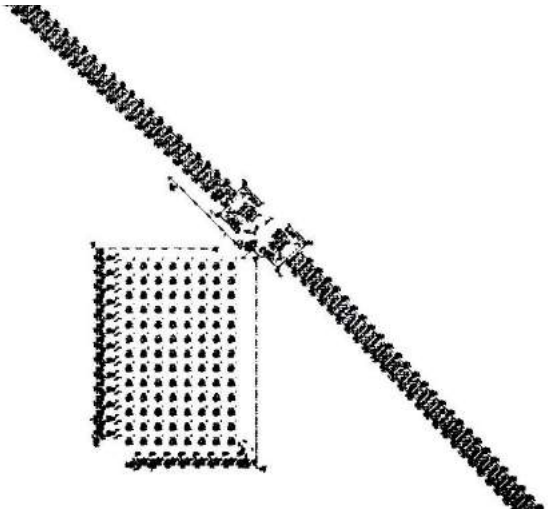
ProCoS-WG group photograph, University of Reading, 1997. Anders Ravn is holding the ProCoS logo below him, Hussein Zedan is sixth from the right, and I am tenth from the right.

This tribute has been written during the time of the coronavirus pandemic. Happily, I know of no formal methods colleagues who have succumbed. However, I have wider more interdisciplinary interests in the field of the arts and its relationship with technology, especially through the EVA London Conference on *Electronic Visualisation and the Arts*, held annually each July at the BCS London offices in recent years (Bowen, 2020). Sadly, one of the conference's former keynote speakers, Kim Veltman, based in the Netherlands, passed away due to the virus on 1 April 2020, aged 71. We have dedicated the EVA London 2020 proceedings (Weinel et al., 2020) to him, which is still to be published even though we are unsure if a delayed physical conference can be held later in the year.

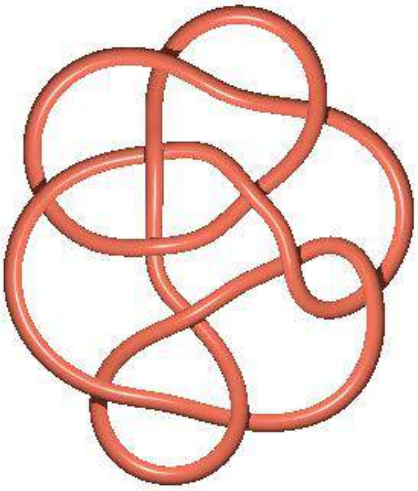
Closer to formal methods is the sad death due to COVID-19 of the British mathematician John Conway FRS on 11 April 2020 in New Jersey, USA. Most computer scientists will know Conway from his *Game of Life* cellular automaton, which can be easily animated on a computer with a graphical display. This is Turing complete, enabling it to perform universal computing (Copeland, Sprevak & Shagrir, 2017, pp. 450–451), and indeed a Turing machine

(Copeland, 2017) can be simulated within the Game of Life (see below) (Bowen, 2016b).

The Conway knot with 11 crossings (see also below) is also named after John Conway. The knot's "sliceness" (its embedding in higher dimensions) has been an open problem for around half a century, but has recently been solved within a week by the young American mathematician Lisa Piccirillo. One wonders what Conway's reaction was, I would imagine one of delight. For those that do not know the answer, the title of the paper is "*The Conway knot is not slice*" (Piccirillo, 2020).



A Turing machine simulated in Conway's Game of Life. YouTube, www.youtube.com/watch?v=My8AsV7bA94



A Conway knot. Wikimedia Commons, commons.wikimedia.org/wiki/File:Conway_knot.png

Despite the current sadness, I hope that the formal methods community can continue to be resilient to the worldwide crisis in which we find ourselves. At least formal methods involve the use of one's brain and the lockdown situation provides more time for thought without interruption.

References

Baber, R.L., Parnas, D.L., Vilkomir, S.A., Harrison, P., O'Connor, T. (2005). Disciplined methods of software specification: a case study. International Conference on Information Technology: Coding and Computing (ITCC'05). IEEE. DOI: 10.1109/ITCC.2005.132

Bjørner, D. (2017). ProCoS: How it all began – as seen from Denmark. In: Hinchey et al. (2017), pp. 3–6. DOI: 10.1007/978-3-319-48628-4_1

Bjørner, D., Hoare, C.A.R., Bowen, J.P., He, J., Langmaack, H., Olderog, E.R., Martin, U., Stavridou, V., Nielson, F., Nielson, H.R., Barringer, H., Edwards, D., Løvengreen, H.H., Ravn, A.P., Rischel, H. (1989). A ProCoS project description: ESPRIT BRA 3104. Bulletin of the European Association for Theoretical Computer Science 39:60–73. URL: researchgate.net/publication/256643262

Black, S., Bowen, J.P., Griffin, K. (2010). Can Twitter save Bletchley Park? In: Bearman, D., Trant, J. (eds.), MW2010: Museums and the Web 2000, Denver, USA, 9–12 April. Archives & Museum Informatics. URL: www.archimuse.com/mw2010/papers/black/black.html

Bogdanov, K., Bowen, J.P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Hierons, R.M., Kapoor, K., Krause, P., Luettgen, G., Simons, A.J.H., Vilkomir, S., Woodward, M.R., Zedan, H.S.M. (2003). Working Together: Formal methods and testing. FORTEST Network, draft of Hierons et al. (2009).

Bowen, J.P. (1988). Formal specification in Z as a design and documentation tool. In: Software Engineering 88, Second IEE/BCS Conference, pp. 164–168. IEE/BCS. URL: ieeexplore.ieee.org/document/196381

Bowen, J.P. (1990). Formal specification of the ProCoS/SAFEMOS instruction set. Microprocessors and Microsystems, 14(10):631–643. DOI: 10.1016/0141-9331(90)90038-W

Bowen, J.P. (ed.) (1994). Towards Verified Systems. Real-Time Safety Critical Systems, vol. 2. Elsevier.

Bowen, J.P. (2016a). BCS-FACS – ProCoS Workshop on Provably Correct Systems. FACS FACTS, 2016-1, March, pp. 14–34. BCS. URL: cdn.bcs.org/bcs-org-media/3087/facs-mar16.pdf

Bowen, J.P. (2016b). Alan Turing: Virtuosity and visualisation. In Bowen, J.P., Diprose, G., Lambert, N., EVA London 2016: Electronic Visualisation and the Arts, pp. 197–204. BCS, Electronic Workshops in Computing (eWiC). DOI: 10.14236/ewic/EVA2016.40

Bowen, J.P. (2019). A Personal Formal Methods Archive. ResearchGate. DOI: 10.13140/RG.2.2.31943.65447

Bowen, J.P. (2020). A personal view of EVA London: Past, present, future. In: Weinel et al. (2020), pp. 8–15. DOI: 10.14236/ewic/EVA2020.2

Bowen, J.P., Fränzle, M., Olderog, E.-R., Ravn, A.P. (1993). Developing correct systems. In: 5th Euromicro Workshop on Real-Time Systems, pp. 176–187. DOI: 10.1109/EMWRT.1993.639088

Bowen, J.P., Gleeson, T.J. (1990). Distributed operating systems. In: Zedan (1990), chap. 1, pp. 3–28. DOI: 10.1016/B978-0-408-02938-4.50006-1

Bowen, J.P., Gordon, M.J.C. (1994) Z and HOL. In: Bowen & Hall (1994), pp. 141–167. DOI: 10.1007/978-1-4471-3452-7_9

Bowen, J.P., Gordon, M.J.C. (1995) A shallow embedding of Z in HOL. Information and Software Technology 37(5–6), 269–276. DOI: 10.1016/0950-5849(95)99362-Q

Bowen, J.P., Gordon, M.J.C., Camilleri, J.A., Pandya, P.K., et al. (1994). Overview of the project. In: Bowen (1994), chap. 2, pp. 35–46. DOI: 10.1016/B978-0-444-89901-9.50011-2

Bowen, J.P., Hall, J.A. (eds.) (1994). Z User Workshop, Cambridge 1994: Proceedings of the Eighth Z User Meeting, Cambridge, 29–30 June 1994. Workshops in Computing, Springer. DOI: 10.1007/978-1-4471-3452-7

Bowen, J.P., Hinchey, M.G., Janicke, H., Ward, M., Zedan, H.S.M. (2014). Formality, agility, security, and evolution in software development. Computer, 47(10):86–89. DOI: 10.1109/MC.2014.284

Bowen, J.P., Hinchey, M.G., Janicke, H.,Ward, M., Zedan, H.S.M. (2018). Formality, agility, security, and evolution in software engineering. In: Hinchey, M.G. (ed.), Software Technology: 10 Years of Innovation in IEEE Computer, chap. 16. Wiley-IEEE Press. DOI: 10.1002/9781119174240.ch16

Bowen, J.P., Hoare, C.A.R., Hansen, M.R., Ravn, A.P., Rischel, H., Olderog, E.-R., Schenke, M., Fränzle, M., Müller-Ulm, M., He, J., Zheng, J. (1994). Provably Correct Systems – FTRTFT'94 tutorial. In: Third International School and Symposium, Formal Techniques in Real Time and Fault Tolerant Systems, Lübeck, Germany, September. ProCoS document [COORD JB 7/1]. URL: researchgate.net/publication/2420842

Bowen, J.P., Hoare, C.A.R., Langmaack, H., Olderog, E.R., Ravn, A.P. (1996). A ProCoS II project final report: ESPRIT Basic Research project 7071. Bulletin of the European Association for Theoretical Computer Science, 59:76–99, June. URL: researchgate.net/publication/2255515

Bowen, J.P., Hoare, C.A.R., Langmaack, H., Olderog, E.-R., Ravn, A.P. (1998). A ProCoS-WG Working Group final report: ESPRIT Working Group 8694. Bulletin of the European Association for Theoretical Computer Science, 64:63–72. URL: researchgate.net/publication/2527052

Breuer, P.T., Madrid, N.M., Bowen, J.P., France, R., Petrie, M.L., Kloos, C.D. (1999). Reasoning about VHDL and VHDL-AMS using denotational semantics. In: Borrione, D. (ed.) DATE'99: Conference on Design, Automation and Test in Europe. pp. 346–352. ACM. DOI: 10.1109/DATE.1999.761144

Copeland, J. (2017). Turing's great invention: The universal computing machine. In: Copeland, Bowen, et al. (2017), chap. 6, pp. 49–56.

Copeland, J., Bowen, J.P., Sprevak, M, Wilson, R., et al. (2017). The Turing Guide. Oxford University Press.

Copeland, J., Sprevak, M., Shagrir, O. (2017). Is the whole universe a computer? In: Copeland, Bowen, et al. (2017), chap. 41, pp. 445–462.

Evans, A., France, R., Lano, K., Rumpe, B. (1998). The UML as a formal modeling notation. In: International Conference on the Unified Modeling Language, pp. 336–348. Springer-Verlag. arXiv:1409.6919. DOI: 10.1007/978-3-540-48480-6_26

France, R.B., Docker, T.W.G. (1988). A formal basis for structured analysis. In: Software Engineering 88, Second IEE/BCS Conference, pp. 191–195. IEE/BCS. URL: ieeexplore.ieee.org/document/196388

France, R.B., Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In: Future of Software Engineering (FOSE'07), 23–25 May. IEEE: 37–54. arXiv:1409.6620. DOI: 10.1109/FOSE.2007.14

Gordon, M.J.C. (1994). State transition assertions: A case study. In: Bowen (1994), chap. 5, pp. 93–113. DOI: 10.1016/B978-0-444-89901-9.50014-8

Gordon, M.J.C., Melham, T.F. (eds.) (1993). Introduction to HOL: A theorem proving environment for higher order logic. Cambridge University Press.

Gordon, M.J.C., Pitts, A.M. (1994). The HOL logic and system. In Bowen (1994), chap. 3, pp. 49–70. DOI: 10.1016/B978-0-444-89901-9.50012-4

Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) (1993). Hybrid Systems. Lecture Notes in Computer Science, vol. 736. Springer-Verlag. DOI: 10.1007/3-540-57318-6

Hierons, R.M., Bogdanov, K., Bowen, J.P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Kapoor, K., Krause, P., Luettgen, G., Simons, A.J.H., Vilkomir, S.A., Woodward, M.R., Zedan, H.S.M. (2009). Using formal specifications to support testing. ACM Computing Surveys (CSUR), 41(2):1–76. DOI: 10.1145/1459352.1459354

Hinchey, M.G., Bowen, J.P., Olderog, E.-R. (2017). Provably Correct Systems. Springer, NASA Monographs in Systems and Software Engineering. DOI: 10.1007/978-3-319-48628-4

Langmaack, H., Ravn, A.P. (1994). The ProCoS Project: Provably Correct Systems. In Bowen (1994), app. B, pp. 249–265. DOI: 10.1016/B978-0-444-89901-9.50022-7

Liu, Z., Ravn, A.P. (eds.) (2009). Automated Technology for Verification and Analysis. Lecture Notes in Computer Science, vol. 5799. Springer-Verlag. DOI: 10.1007/978-3-642-04761-9

Olderog, E.-R., Ravn, A.P., Wisniewski, R. (2017). Linking discrete and continuous models, applied to traffic manoeuvres. In: Hinchey et al. (2017), pp. 95–120. DOI: 10.1007/978-3-319-48628-4_5

Piccirillo, L. (2020). The Conway knot is not slice. Annals of Mathematics, 191(2):581–591. DOI: 10.4007/annals.2020.191.2.5

Ravn, A.P., et al. (1993). Provably Correct Systems (ProCoS). In: Larsen, P.G. (ed.) (1993). Tutorial Material, Formal Methods Europe '93: Industrial-Strength Formal Methods. Formal Methods Europe, Odense, Denmark, April, pp. 437–450.

Ravn, A.P., Rischel, H. (eds.) (1998). Formal Techniques in Real-Time and Fault-Tolerant Systems. Lecture Notes in Computer Science, vol. 1486. Springer-Verlag. DOI: doi.org/10.1007/BFb0055330

Scholefield, D., Zedan, H., He, J. (1994). A specification-oriented semantics for the refinement of real-time systems. Theoretical Computer Science, 131(1):219–241. DOI: 10.1016/0304-3975(94)90096-5

Siewe, F., Cau, A., Zedan, H. (2003). A compositional framework for access control policies enforcement. FMSE '03: Proceedings of the 2003 ACM workshop on Formal Methods in Security Engineering, pp. 32–42. ACM. DOI: 10.1145/1035429.1035433

Vilkomir, S.A., Bowen, J.P. (2001a). Formalization of control-flow criteria of software testing. Technical Report SBU-CISM-01-01, South Bank University, SCISM, London, UK.

Vilkomir, S.A., Bowen, J.P. (2001b). Application of formal methods for establishing regulatory requirements for safety-critical software of real-time control systems. Technical Report SBU-CISM-01-03, South Bank University, SCISM, London, UK.

Vilkomir, S.A., Bowen, J.P. (2001c). Formalization of software testing criteria using the Z notation. In: COMPSAC 2001: 25th Annual International Computer Software and Applications Conference. pp. 351–356. IEEE. DOI: 10.1109/CMPSAC.2001.960638

Vilkomir, S.A., Bowen, J.P. (2002a). From MC/DC to RC/DC: Formalization and analysis of control-flow testing criteria. Technical Report SBU-CISM-02-17, South Bank University, SCISM, London, UK.

Vilkomir, S.A., Bowen, J.P. (2002b). Establishing formal regulatory requirements for safety-critical software certification. In: AQuIS 2002: 5th International Conference on Achieving Quality In Software, Venice, Italy, March, pp. 7–18. URL: researchgate.net/publication/2494218

Vilkomir, S.A., Bowen, J.P. (2002c). Reinforced condition/decision coverage (RC/DC): A new criterion for software testing. In: Bert, D., Bowen, J.P., Henson, M.C., Robinson, K. (eds.), ZB 2002: Formal Specification and Development in Z and B: 2nd International Conference of B and Z Users Grenoble, France, January 23–25. Lecture Notes in Computer Science, vol. 2272, pp. 229–239. Springer-Verlag. DOI: 10.1007/3-540-45648-1_15

Vilkomir, S.A., Bowen, J.P. (2006). From MC/DC to RC/DC: Formalization and analysis of control-flow testing criteria. Formal Aspects of Computing, 18(1):42–62. DOI: 10.1007/s00165-005-0084-7

Vilkomir, S.A., Bowen, J.P. (2008). From MC/DC to RC/DC: Formalization and analysis of control-flow testing criteria. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.): Formal Methods and Testing: An Outcome of the FORTEST Network. Lecture Notes in Computer Science, vol. 4949, pp. 240–270. Springer-Verlag. DOI: 10.1007/978-3-540-78917-8_8

Vilkomir, S.A., Bowen, J.P., Ghose, A.K. (2006). Formalization and assessment of regulatory requirements for safety-critical software. Innovations in Systems and Software Engineering: A NASA Journal, 2:165–178. DOI: 10.1007/s11334-006-0006-8

Vilkomir, S.A., Kapoor, K., Bowen, J.P. (2003). Tolerance of control-flow testing criteria. In: COMPSAC 2003: 27th Annual International Computer Software and Applications Conference, pp. 182–187. IEEE. DOI: 10.1109/CMPSAC.2003.1245339

Weinel, J., Bowen, J.P., Diprose, G., Lambert, N. (2020). EVA London 2020: Electronic Visualisation and the Arts. BCS, Electronic Workshops in Computing (eWiC). ScienceOpen. DOI: 10.14236/ewic/EVA2020.0

Zedan, H.S.M. (ed.) (1990a). Distributed Computer Systems: Theory and Practice. Butterworths. DOI: 10.1016/C2013-0-06277-9

Zedan, H.S.M. (1990b). Formal aspects of microprocessor design. Microprocessors and Microsystems, 14(10):621–622. DOI: 10.1016/0141-9331(90)90036-U

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

Forthcoming events

Events Venue (unless otherwise specified):

NEW *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.

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

Jonathan Bowen
FACS Chair and BCS Liaison

John Cooke
FACS Treasurer and Publications

Roger Carsley
Minutes Secretary

Ana Cavalcanti
FME Liaison

Brijesh Dongol
Refinement Workshop Liaison

Rob Hierons
LMS Liaison

Keith Lines
Government and Standards Liaison

Sofia Meacham
Seminar Organiser

Margaret West
Inclusion Officer and BCS Women Liaison

Tim Denvir
Co-Editor, FACS FACTS

Brian Monahan
Co-Editor, FACS FACTS

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, Professor Jonathan Bowen at the contact points below:

BCS-FACS
c/o Professor Jonathan Bowen (Chair)
London South Bank University
Email: jonathan.bowen@lsbu.ac.uk
Web: www.bcs-facs.org

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

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 subscribers. An archive of messages is accessible under http://www.jiscmail.ac.uk/lists/facs.html, including facilities for subscribing and unsubscribing. The additional <facs-event@jiscmail.ac.uk> list is specifically for announcements of relevant events. Similarly, an archive of announcements is accessible under http://www.jiscmail.ac.uk/lists/facs-events.html with subscribe/unsubscribe options. 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.