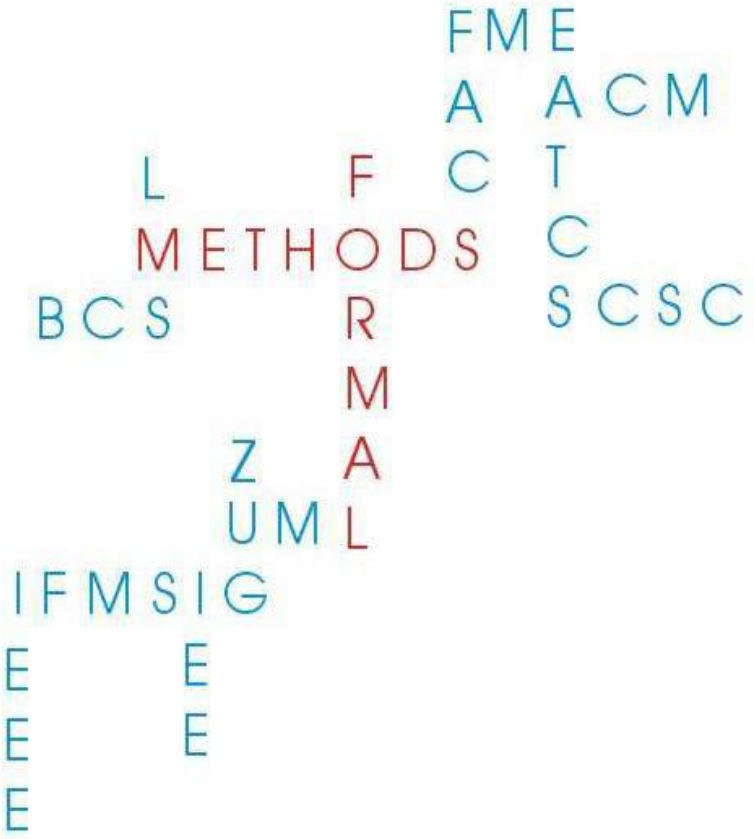
Issue 2021-1
February 2021

FACS

A C T S



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 briangmonahan@googlemail.com

Editorial Team:

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

Contributors to this issue:

Jonathan Bowen, Cliff Jones, Carroll Morgan, John V Tucker

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* for 2021. Despite an understandably quiet year for all activities, including FACS, we nonetheless have quite a packed and diverse issue for you. This has a predominately historical theme running through it, touching upon the 60th anniversary of ALGOL 60, the formalisation of programming language semantics, the origins of combinatory logic and lambda calculus, as well as the History of Computing Collection based at Swansea!

As is customary following our AGM, we open with the Annual Chair's Report from Jonathan Bowen. This summarises the offering that BCS FACS makes to its members and presents our objectives for the coming year.

This is followed by John Tucker discussing the ongoing important work of the History of Computing Collection, which highlighted, among other areas, the archive of formal methods related materials that is being curated there. This provides an invaluable much needed opportunity to create a historical record of formal methods activity In research and development that also draws attention to the need for us to re-visit, re-interpret and preserve what we know about software development.

We continue with a personal recollection from Cliff Jones which documents the evolution of the formal semantics and specification frameworks known as VDL, through to VDM and from there into SOS. This is an extraordinary journey which encompasses, as it does, both denotational and (modern) operational semantics formalisms and techniques.

Related to the previous article, is the subject of the lambda calculus which provides a foundational bedrock for so much of formal methods and software development. Our next article from Jonathan Bowen reports on an online talk given by Stephen Wolfram about Moses Schönfinkel. one of the early pioneers of combinatory logic and lambda calculus. Schönfinkel worked in David Hilbert's group, influenced Haskell Curry, and was the first to show how combinator theory could provide a logical calculus via equational reduction.

In September of 2020, we learned that, sadly, Ken Robinson had passed away. Ken was well known to many working in the program refinement community at Oxford PRG and Royal Holloway during the 1980s: many contributors have

provided the eulogy below. We offer our condolences to his wife, Rosalie, and his surviving family in Sydney, Australia.

Lastly, John Tucker also reports on the 2020 Peter Landin Semantics Lecture given in December entitled "ALGOL 60 @ 60", co-presented by Tim Denvir and Troy Astarte in the form of a Zoom online meeting. This was well attended and, because it was provided online, it was also enjoyed by an international audience with attendees from the US and Europe, as well as the UK.

Also included herein is a fascinating aside from Jonathan Bowen, concerning works with ALGOL 60 going on in Oxford during the early 1970s—from two very different points of view!

STOP PRESS: We have just received an anonymous missive giving news of our esteemed colleague F.X.Reid! We understand that he has been abroad on expedition for several years, working on a range of, dare we say, "secret projects". Confusing the rumours even further, it is our very great pleasure to reveal this somewhat mysterious report to his loyal followers through FACS FACTS! Or could the anonymous correspondent be F.X.Reid himself, writing incognito? You decide!

Brian Monahan, Tim Denvir

FACS-FACTS co-editors

BCS-FACS Specialist Group

Formal Aspects of Computing Science
Chair's Annual Report (2020)

Jonathan Bowen

Overview

BCS-FACS is one of the more academic BCS specialist groups, with an associated journal and newsletter. We mainly organise evening seminars, usually at the BCS London office. Annual events include a joint London Mathematical Society LMS/FACS seminar at the LMS headquarters and the Peter Landin Semantics Seminar, presented online by Tim Denvir and Troy Astarte in December 2020, and chaired by Jonathan Bowen. 2020's FACS/LMS seminar, normally organised by Rob Hierons, was cancelled due to the pandemic. We occasionally have larger events, sometimes with a publication, but the pandemic has curtained many activities this year. We sometimes organise joint meetings with Formal Methods Europe (FME); again this year we did not hold such an event, but aim to do so in 2021. A project funded by a Supplementary Funding Request (SFR) enabled much early FACS material held by Tim Denvir, including newsletters, to be digitised and this material is now accessible on the BCS-FACS website.

Programme

Events are normally mainly evening seminars in lecture style, most recently at the new BCS London office in Moorgate, central London, with audience numbers ranging from around 20 to 75. Several talks were planned during 2020 but were cancelled due to the pandemic. We aim to run some of these as online Zoom talks or after the BCS London office reopens, hopefully in 2021. The 2020 AGM and Peter Landin Semantics seminar held on Thursday 3 December 2020 was our first online talk using Zoom.

Note that the BCS London Office is now at 25 Copthall Avenue, London EC2R 7BP. The nearest tube station is Moorgate, but Bank and Liverpool Street are within walking distance as well. However, since the start of the pandemic in Spring 2020, this facility has been closed and as of the writing of this report in January 2021, no reopening date has been announced.

FACS's Role as a BCS Specialist Group

Formal methods are very important in safety and security critical and related IT systems. We mainly provide a forum for academics and professionals, especially through our evening seminars. We encourage a significant question time at the end of seminars (up to half an hour, normally about quarter of an hour, depending on the level of interest and interactivity during the talk itself), followed by an informal networking session for further discussion for all attendees and the speaker (typically an hour or more).

Our mission is to promote the awareness, development and application of:

- A mathematical basis for computer science;
- Theories underpinning practice in computing;
- Rigorous approaches to information processing in computer-based systems.

Members receive the *FACS FACTS* newsletter and are eligible for a discount at our workshops and conferences when appropriate.

Community

We serve the United Kingdom's formal methods community, with some international support largely through collaboration with Formal Methods Europe (FME). A significant number of attendees at our events are non-members of the BCS, but are potential members. We sponsor some external events, sometimes as an alternative to our more normal evening seminars in London, which helps to raise the profile of both the BCS and FACS internationally.

Leadership

We are the leading national formal methods group in the United Kingdom. We are informally affiliated with Formal Methods Europe (FME), the leading formal methods organisation in the world. We organise joint meetings at the BCS London office and elsewhere. Future events for the moment are likely to be online, although we find the evening seminars at the BCS London office are very useful for community networking with refreshments after the talks.

We also normally co-organise a joint meeting with the London Mathematical Society (LMS) the United Kingdom's leading mathematical society. We plan to continue with these activities when it is possible to meet at the LMS headquarters in London again, hopefully in November 2021. In addition, we

previously undertook a Supplementary Funding Request (SFR) funded project to digitise our material that is still in print format in celebration of our 40th anniversary. This is accessible on the BCS-FACS website under: https://www.bcs.org/membership/member-communities/facs-formal-aspects-of-computing-science-group/newsletters/

Thanks are due to Tim Denvir for saving this material and arranging for its digitisation. We may digitise further material in due course.

Excellence

We provide high-quality talks by leaders in formal methods. We are one of the most academic of the BCS Specialist Groups, perhaps the most academic group. We have an associated academic journal, *Formal Aspects of Computing*, published by Springer, and a more informal online newsletter, *FACS FACTS*. We publish occasional books and proceedings associated with FACS events, mainly through Springer, which is well regarded in academic circles. We liaise with related organisations that have interests aligned with those of FACS, such as Formal Methods Europe (FME) and the London Mathematical Society (LMS).

Objectives for 2021

FACS events and Evening Seminars and associated events are planned to be online until the BCS London office reopens again. Information on FACS events can be found on the BCS-FACS website, accessible under www.bcs-facs.org. Currently plans for further physical events are on hold due to the pandemic, but we hope to resume normal operation at the BCS London office in 2021, after it has reopened. We welcome ideas for further FACS events from FACS members. We are also seeking an active member to join the FACS committee to help as a Seminar Organiser. Sadly our previous Seminar Organiser, Sofia Meacham of Bournemouth University, was unable to continue helping the Chair with the organisation of FACS evening seminars after the end of 2020. Please contact the FACS Chair, Jonathan Bowen on jonathan.bowen@lsbu.ac.uk, if you are able to help. It is a rewarding role in that you can invite people you have always wished to hear talk or meet. You can also claim travel expenses to organise and chair FACS talks. FACS looks forward to using the new BCS London office for future BCS-FACS events once it has reopened. The Chair has produced a written guide on organising FACS seminars, for anyone willing to take on this role.

Achievements

Our liaison with similar groups such as FME and LMS is mutually beneficial, with joint talks and events. Our Annual Peter Landin Semantics Seminar each December is the highlight of our year, with a high-profile speaker and normally a networking reception after the talk. Holding our AGM before this major event works well each year.

The periodic online newsletter, *FACS FACTS*, edited by Tim Denvir and Brian Monahan, allows FACS members to publish short articles, provides reports on FACS talks and other events, book reviews and announcements, etc. The previous issue was in June 2020. Our affiliated journal published by Springer, *Formal Aspects of Computing*, is considered by many as the leading journal on formal methods in the world, for peer reviewed papers. We sometimes publish books and proceedings associated with FACS events. See for example:

- *Formal Methods: State of the Art and New Directions*, Boca, Paul, Bowen, Jonathan P., Siddiqi, Jawed (eds.). Springer, 2010. ISBN 978-1848827356. See: http://www.springer.com/gb/book/9781848827356
- *Provably Correct Systems*, Hinchey, Mike, Bowen, Jonathan P., Olderog, Ernst-Rüdiger (eds.). Springer, NASA Monographs in Systems and Software Engineering, 2017. ISBN 978-3319486277. See:

  http://www.springer.com/gb/book/9783319486277

Over the years we have published 24+ books and proceedings through Springer. We have also published proceedings through the BCS Electronic Workshops in Computing (eWiC) series. In addition, the *Formal Aspects of Computing* journal has now completed over 30 years of publication.

Recommendations for BCS Appreciation Awards

Such recommendations are welcome by any FACS members. If any FACS member is aware of someone who warrants an award by the BCS, then please contact the BCS Community Team on **groups@bcs.uk** for a form, which can also be found on the BCS Volunteer Portal under:

https://volunteer.bcs.org/content/volunteer-awards-recognition

Challenges

We would like to make more of our evening seminars available online. We have experimented with YouTube. However, this is problematic for committee

members who only do this occasionally. With the advent of the pandemic, BCS support in this area has improved. Zoom talks can now be recorded on the Cloud and placed on YouTube by the BCS. We have done this for the 2020 Peter Landin Semantics Seminar. We hope that this will alleviate one of the aspects of running a FACS evening seminar for volunteers in the future. Of course the pandemic has been a challenge in terms of organising events, as has the loss of our Seminar Organiser.

Inclusion and Diversity

We very much welcome anyone who is willing to be actively involved in committee activities. We encourage new active members, mainly through advertising via several relevant electronic mailing lists and networking with attendees at our evening seminars. Margaret West, University of Huddersfield, is our Inclusion Officer on the FACS committee. We wish to be inclusive for those not close to London by providing recordings and slides of evening seminars, as well as other print material online. Holding online Zoom talks as for the 2020 Landin Seminar is helpful with this issue.

About the FACS Committee

FACS has a relative low turnover of committee members. However, we encourage new active members, mainly through networking with attendees at our evening seminars. Concerning the FACS officers, Jonathan Bowen is the Chair, John Cooke is the Treasurer and Roger Carsley is the Secretary. Tim Denvir and Brian Monahan are co-editors of the *FACS FACTS* newsletter. Rob Hierons, University of Sheffield, undertakes LMS Liaison and normally organises the joint FACS-LMS Seminar at the LMS headquarters in central London. Ana Cavalcanti, University of York and Chair of Formal Methods Europe (FME), undertakes liaison with FME. Margaret West, University of Huddersfield, is our Inclusion Officer. Keith Lines of the National Physical Laboratory is our Government and Standards Liaison representative and has co-organised evening seminars by NPL speakers. Sofia Meacham of Bournemouth University has been our Seminar Organiser; many thanks for her help in this role up to December 2020. From December 2019, Brijesh Dongol, University of Surrey, has replaced John Derrick on the committee, undertaking Refinement Workshop Liaison. And thanks to all on the FACS Committee for their help in the BCS-FACS Specialist Group's continued operation and success.

History of Computing Collection at Swansea University

John V Tucker

"We had the experience, but we missed the meaning."
--- T. S. Eliot

Swansea University has a **History of Computing Collection**. It consists of a museum, archive, library and legacy laboratory. It was founded by my friend Steve Williams, Head of the University Library, and myself in Autumn 2007 and is intended to be a University resource for the study of both the technical development of computing and the impact of computing technologies on society and people. After we made its existence public in 2013, it soon accumulated a list of concrete aims and aspirations to:

- Rescue materials charting the history of computing
- Engage the interest of colleagues, students and the public in the history of data and computing broadly conceived
- Help resurrect legacy systems and obsolete digital media
- Promote research on the local history of computing.

Some of these activities have matured, others remain in infancy. Today, the Collection contains equipment, software, reports, ephemera, oral histories, photographs and videos. It is very modest. It is a work in progress. But, it is far from uninteresting for readers of this Newsletter, for one of its specialisms is the *history of formal methods for software development.*

In writing about the Collection here, I want to make it better known to attract reflections, testimonies, materials, and users. I also want to draw attention to the need for us to re-visit, re-interpret and preserve what we know about software development. The formal methods community needs to start making histories of their emergence, development, achievements and failures. Here, I will try give an impression of the whole Collection, but will begin with the formal methods archive.

Formal methods for software development

Our particular interest in programming and software development has quietly attracted a number of gifts of personal archives. For example, we have received large collections of notes, articles and books from some well-known computer scientists, including:

- Willem Paul de Roever (Kiel),
- Peter van Emde Boas (Amsterdam)
- Dines Bjørner (Copenhagen)
- Jonathan Bowen (Museophile Ltd and London South Bank)

and there are my own files, of course. Taken together, the archive contains original material that can illustrate a decent amount of the development of formal methods for syntax, semantics, specification, program development, and verification. It has many interesting papers on language definitions and constructs (e.g., for data types, concurrency, refinement, etc.)

However, our formal methods archive is far from being comprehensive. It is patchy on pioneering projects to develop or apply formal methods (it has material on some Alvey and ESPRIT projects). It needs plenty more anecdotes and contextual material to help contemporary and future scholars and students to explore software development as technological heritage. Its collection of early branded ephemera – posters, coffee mugs, tee shirts etc. – could be more comprehensive!

I have relied on the Collection to help me write a short historical survey of formal methods for software engineering. At the invitation of my colleague Markus Roggenbach, this will appear as a chapter in a new textbook on formal methods, Roggenbach et al (2021); I am now working on an expanded version with more detail and wider scope.

The Collection is also active in other specialist areas of computing. The choice of subjects reflects my own and my colleagues' interests, and the interests of some friends of the Collection. For example, we have a fledgling collection on security and, recently, we have begun to encourage and support investigations of HCI.

History of Human Computer Interaction.

At Swansea, my colleagues Harold Thimbleby and Alan Dix have worked on many aspects of HCI over the years and have been keen to draw on formal methods where relevant. Alan has created a small community of researchers and practitioners through an online workshop on which was held this year in July (see: https://hcibook.net/hcihistory). The emphasis is on the UK, but many readers will recall the substantial work on formal methods applied to HCI at

York University, by Harold and Alan, together with Michael Harrison and Gregory Abowd.

Computation before Computers

The centre piece of this area is the *Leslie John Comrie Archive*. Leslie John Comrie FRS (1893-1950) was the doyen of numerical computation and simulation in the age just before digital computers. He set new high standards of analysis and accuracy in modelling in many areas of science and engineering, and developed large-scale computation by mechanical devices. Comrie also introduced computational methods as an independent academic topic in the 1920s and is popularly known for his impeccable mathematical tables. Our archive contains notes, papers and his large collection of mathematical tables associated with his computation company, *Scientific Computing Service Ltd.*

The *Comrie Archive* includes rare volumes and papers by Comrie and many other table makers including Barlow and Babbage. It is a convenient and useful resource for those interested in the development of numerical methods for simulation and their application, in tables of mathematical data and physical measurements, and methods for mathematical table making.

Local History of Computing

One important focus is the development of computing in Wales. I have found that by investigating the *local* history of computing I am better able to discover and understand the complicated interplay of technical, social, economic and cultural factors that are ever present in computing. Certainly, local tech history is an eye-opener for our staff and students.

This year is the Centenary of Swansea University and I was able to use the Collection to help me write an article charting the arrival and growth of computing at Swansea University up to 1989, the year I arrived: Tucker (2020). I hope my essay is of interest as a UK institutional history of computer science, of which there are too few.

In 1967, Swansea's first professor of computer science, David Cooper, arrived from Carnegie Tech (now CMU). He was a pioneer in formal methods and created a research group here on program verification with Science Research Council (now EPSRC) funding. One of the members of the group was Robin Milner who wrote many reports about theorem proving and program equivalence, which are in the Collection of course.

The Collection contains interviews, written reflections, and notes of conversations with people on a wide range of subjects, which help with context and new research questions. Consider the question:

How have we learned to develop, apply and use computers and software?

An obvious and naïve answer is through schools, colleges and universities. But the scale and strength of competence and achievement suggests that it is as much through informal learning as formal learning: people have taught themselves in work and at home, supported by colleagues and friends. This seems to be a somewhat neglected area of historical research. The Collection has paid some attention to the historical origins and growth of people's knowledge and skills through local topics such as

- Early computing services in the steel industry at Port Talbot
- The creation of the Driver and Vehicle Licensing Agency in Swansea 1965-75
- Early computer science education in schools

This topic has some relevance to thinking about dissemination problems for formal methods.

Finally, popular parts of any computing collection are the machines, peripheral devices and software, especially software familiar to users such as operating systems and games. The majority of the items in the Collection have local connections with south Wales. For example, we have mechanical and electronic calculators, the remains of the University's first computer, an IBM 1620; Swansea's first web server from 1993; the first Linux host in the UK which, together with several early Linux releases were based at Swansea, and many personal computers, even an iMac made in Newport.

Why the Collection?

I'd like to think there is little need to explain *why* a university should make a local collection on computing. I expect our aims apply in all sorts of places. To be specific in the case of Swansea, I have been interested in the history of science and technology since I was a student. From the beginning, I have sprinkled my lectures with historical context. I began teaching history of computing courses in 1994, much inspired by Martin Campbell-Kelly, and motivated by the wish to learn a lot more about it. Only recently do I feel able to write histories. I need hardly point out that the history of computing is not in

good shape in this country. Courses are few and far between and the subject does not find a natural home outside computer science – at the moment.

The Collection at Swansea was not designed simply for students, mine in computer science, or of colleagues in media, history, heritage etc. The big step to make the Collection was inspired by an earlier Swansea University initiative of the 1960s, when Hywel Francis set about rescuing the material from the miners' institutes belonging to the declining industrial communities of South Wales. He created a remarkable collection about the economy, society and culture of the coalfield. It is called the *South Wales Miners'* Library and is the most dynamic of the University's industrial collections, Francis and Williams (2013).

But why bother with making an archive to piece together the history of formal methods? My answer is this: formal methods, broadly conceived, have played the primary role in providing scientific foundations for computer science. The core activities of modelling, specifying, implementing and reasoning have been applied to all areas of computing, albeit to greater or lesser degrees. There are major achievements and there is no shortage of hard problems to tackle – some classic, some forgotten, some fashionable, some only just coming into view. The philosophical issues of computing are never far from its formal foundations.

It starts with individuals putting their personal archives in order. Jonathan Bowen has written about this in Bowen (2019). Readers of this article who are luke-warm about history may prefer to reflect on the need of a forward radical looking agenda. It seems to me that formal methods have opportunities galore in the new directions computer science are taking. For this there needs to be a new push on educating computer scientists, for as new themes and directions emerge, formal thinking needs renovation and new courses. What we know is what has been revealed by the past.

We hope you will help us build the collection; you are most welcome to contact me.

The Collection

The *History of Computing Collection* is in the Margam Building on the Singleton Campus of Swansea University; it can be visited by appointment. A small number of items from the Collection are on display in the Computational Foundry, Bay Campus, which is the home of the Computer Science Department. For basic information about the Collection see:

http://hocc.swan.ac.uk

Some images of items are available here:

https://www.flickr.com/photos/hocc_swansea

Some video material is available here:

https://www.youtube.com/channel/UCn00nU-ndo9tDMY4AezF_uQ

Let me recommend videos of two lectures: Willem Paul de Roever on his research programme and the personal archive he presented to the Collection in 2014

https://www.youtube.com/watch?v=PJPTZpdck7k

and Brian Randell on *40 Years of Software Engineering*

https://www.youtube.com/watch?v=ANCA8vG3Tbo

also in 2014.

HoCC Foyer (Photo: History of Computing Collection)

HoCC Study Area

(Photo: History of Computing Collection)

HoCC Store

(Photo: History of Computing Collection)

Jonathan Bowen's contribution of 64 box files for the archive arriving at Swansea in September 2018

(Photograph by Jonathan Bowen)

References

Jonathan P. Bowen, *A Personal Formal Methods Archive*. ResearchGate, October 2019. DOI: https://doi.org/10.13140/RG.2.2.31943.65447

Hywel Francis and Sian Williams, *Do Miners Read Dickens? The Origins and Progress of the South Wales Miners Library*, Parthian Books, 2013. https://www.swansea.ac.uk/library/south-wales-miners-library/

M Roggenbach, A Cerone, B-H Schlingloff, G Schneider, S A Shaikh, *Formal Methods for Software Engineering. Languages, Methods, Application Domains,* Springer, 2021, to appear. https://www.springer.com/gp/book/9783030387990

John V Tucker, *The Computer Revolution and Us: Computer Science at Swansea University from the 1960s,* Swansea University 2020. https://collections.swansea.ac.uk/s/swansea-2020/page/computer-science

John V Tucker, The origins and development of formal methods for software engineering, in Roggenbach et al (2021), to appear.

From VDL to VDM to SOS

Cliff B. Jones

School of Computing, Newcastle University

Having played a significant part in the switch from the operational VDL (Vienna Definition Language — see [LW69]) to the denotational approach used in VDM (Vienna Development Method — see [BJ82]), a reader might ask why, in 2020, I based [Jon20] on operational semantics. This brief note (I predicted "a page") explains why I have returned to the operational approach for teaching formal semantics. (A possibly unkind description would be to say this is just a plug for my new book — I do make a more general point at the end of this note.)

1 In the beginning was VDL

Long, long ago, I went to the IBM Lab in Vienna to learn about VDL. The background to this was that I had been involved in testing the PL/I "F" compiler at IBM's Hursley Lab. Before Dijkstra's most famous aphorism, I had learned that it was not possible to test quality into an "imperfect" body of code.

There was a one week course in Vienna in April 1968 and by September that year I started a two-year assignment at the the Lab there. They were just wrapping up the third version of the huge "ULD III" description of PL/I. One of my early tasks was to review Peter Lauer's VDL description of ALGOL 60 [Lau68] which had been undertaken to show that the (telephone directory) size of the PL/I description were the fault of PL/I not of the VDL approach to language description.

My main aim was to look at how a (any) formal semantic description could be used as a firm starting point for a rational compiler design process. Peter Lucas had already had the "twin machine" inspiration and he and I published [JL70]. But it was clear that the "grand state" ULD operational descriptions made proofs gratuitously difficult. An operational semantics for a non-deterministic language takes a state to a set of states. The problem with the states in VDL descriptions was that they contained everything (including the proverbial kitchen sink). I'll come to the "Control Component" shortly but to take a different example, the state of the PL/I (or ALGOL) descriptions had a stack of environments — one for each active block or procedure. In [JL70], we had to prove that the environment of the $i+1$st statement was the same as that for the ith even if the latter was a block or a call statement. It is far wiser to split the environment out of the state and show it as a parameter (but not a result) of the interpreting function.

A series of ideas came up in my first stay in Vienna; some that matter for this purpose are a view of data refinement [Jon70] (although I now prefer the term "reification") and the "exit" idea [HJ70] for handling exceptional statement ordering such as **goto** statements.

2 Back in Hursley

I went back to the Hursley Lab at the end of 1970 to manage an "Ad Tech" group. One activity relating to semantics was a description of ALGOL [ACJ72] that we called a "Functional Semantics" — it used the exit idea but without any "combinators" so it looked rather heavy. For the part of what became VDM that concerned development of general (not just compiler) programs, there was work on relational post conditions [Jon73] and data refinement [Jon72].

3 Return to Vienna = VDM

Then came a call from Peter Lucas in late 1972 saying that the Vienna Lab had the chance to build a PL/I compiler for a new machine architecture. I jumped at the chance to return.

Several things had been going on in the period before I moved back in early 1973:

- Hans Bekič had spent a year at Queen Mary College London with Peter Landin.
- I'd attended some of Christopher Strachey's lectures at PRG in Oxford.
- Peter Mosses had published [Mos74] a denotational semantics of ALGOL that used "continuations" with a specific swipe at [ACJ72].
- There had been an exchange of letters between Peter, Hans and me on overcoming some of the difficulties in using operational semantics as a basis for compiler design.

We got to work straight away on writing a denotational description of the ECMA/ANSI subset of PL/I (and heaved a sigh of relief that the tasking "features" had been dropped). Our PL/I description appeared as a Technical Report [BBH+74] and used a combinator form of the exit idea. (Peter Mosses looks at links to "monads" in [Mos11].) We also wrote several Technical Reports on the use of a formal semantics for compiler design [BIJW75,Jon76].

Then IBM decided not to build the "novel machines" and a compiler for a non-existent machine is not useful. Many key members of the Lab moved to other posts and the good work could have been lost. Dines Bjørner and I edited [BJ78] (which was later rewritten as [BJ82]).

My own next step was to move to IBM's "European Systems Research Institute" where I worked on the general program development aspects of "VDM" which were first published in [Jon80]. In 1979, I then went to Oxford to fill a small gap in my academic credentials. There I started the work on concurrency that has occupied most of my research time since.

4 SOS

In 1981 I moved to Manchester and began teaching Denotational Semantics in the VDM style. The bigger event in 1981 was the publication of Gordon Plotkin's

SOS notes [Plo81] from Arhus (I lent these to so many people that my copy was falling apart and it was a relief when the notes were reprinted as [Plo04b] (the accompanying notes [Plo04a] are a must-read)).

After a three-year interlude back in industry at Harlequin, I returned to academia with a move to Newcastle. At Harlequin, Brian Monahan and I met again, after he had valiantly sorted out the semantics of VDM [Mon87] several years previously.

Another chance to teach semantics presented itself. Why did I choose to use operational semantics? My aim was to communicate the idea that it was not difficult to "model" a programming language without the need to build a compiler. Thousands of programming languages have been devised — very few of them were so modelled — most of them have nasty corners. I felt these two observations were connected. Even in clean languages such as Pascal there are "feature interactions": I remember Derek Andrews almost begging me to let him remove either untagged variant records or pass-by-reference from the Pascal description in [AH82] because the combination of these two features significantly expands the model.

The overriding reason for switching to SOS was however concurrency. The Control Component in a VDL description was essentially the text of the program to be executed; the tree (upside down of course) could have many leaves that indicated points (think statements but it was actually finer granularity) that could be executed next. The key contribution of SOS descriptions is that this text is factored out as one part of the "configuration" — the semantics becomes a relation between configurations (with all of the other small state ideas such as factoring out environments).

"UPLs" (as we refer to [Jon20]) goes as far as modelling a concurrent object oriented language.

5 Postscript

There are of course arguments for the denotational approach. Andrzej Blikle kindly commented on a draft of my book and is currently writing his own — he avoids much of the complexity of domain theory by using the ideas in [Bli83].

If there is an underlying message to this "brief" note, it is that there is virtue in using formal ideas that work rather than always jumping to publish some new approach.

For more details on semantic descriptions, see [JA16,JA17]; a readable account of the Vienna and Oxford work on semantics is given in [Ast19]; and an even easier path into this subject can be found by watching the recording [AD20] of Tim Denvir's and Troy Astarte's Landin seminar.

References

[ACJ72] C. D. Allen, D. N. Chapman, and C. B. Jones. A formal definition of ALGOL 60. Technical Report 12.105, IBM Laboratory Hursley, 8 1972.

[AD20] Troy K. Astarte and B. T. Denvir. Algol 60 @ 60: its place in formal semantics, 12 2020. Peter Landin Semantics Seminar.

[AH82] Derek Andrews and Wolfgang Henhapl. Pascal. In Bjørner and Jones [BJ82], chapter 6, pages 175–252.

[Ast19] Troy K. Astarte. *Formalising Meaning: a History of Programming Language Semantics*. PhD thesis, Newcastle University, 6 2019.

[BBH+74] Hans Bekič, Dines Bjørner, Wolfgang Henhapl, Cliff B. Jones, and Peter Lucas. A formal definition of a PL/I subset. Technical Report 25.139, IBM Laboratory Vienna, 12 1974.

[BIJW75] H. Bekič, H. Izbicki, C. B. Jones, and F. Weissenböck. Some experiments with using a formal language definition in compiler development. Laboratory Note LN 25.3.107, IBM Laboratory, Vienna, 12 1975.

[BJ78] D. Bjørner and C. B. Jones, editors. *The Vienna Development Method: The Meta-Language*, volume 61 of *Lecture Notes in Computer Science*. Springer-Verlag, 1978.

[BJ82] Dines Bjørner and Cliff B. Jones, editors. *Formal Specification and Software Development*. Prentice Hall International, 1982.

[Bli83] A. Blikle. A metalanguage for naive denotational semantics. Technical Report 104, Consiglio Nazionale Delle Ricerche, ETS, Pisa, 1983.

[HJ70] W. Henhapl and C. B. Jones. On the interpretation of GOTO statements in the ULD. Technical Report LN 25.3.065, IBM Laboratory, Vienna, 3 1970.

[JA16] Cliff B. Jones and Troy K. Astarte. An exegesis of four formal descriptions of ALGOL 60. Technical Report CS-TR-1498, Newcastle University School of Computer Science, 9 2016.

[JA17] Cliff B. Jones and Troy K. Astarte. Challenges for semantic description: comparing responses from the main approaches. Technical Report CS-TR-1516, Newcastle University School of Computer Science, 11 2017.

[JL70] C. B. Jones and P. Lucas. Proving correctness of implementation techniques. Technical Report TR 25.110, IBM Laboratory Vienna, 8 1970.

[Jon70] C. B. Jones. A technique for showing that two functions preserve a relation between their domains. Technical Report LR 25.3.067, IBM Laboratory, Vienna, 4 1970.

[Jon72] C. B. Jones. Formal development of correct algorithms: an example based on Earley's recogniser. In *SIGPLAN Notices*, volume 7:1, pages 150–169. ACM, 1972.

[Jon73] C. B. Jones. Formal development of programs. Technical Report 12.117, IBM Laboratory Hursley, 6 1973.

[Jon76] C. B. Jones. Formal definition in compiler development. Technical Report 25.145, IBM Laboratory Vienna, 2 1976.

[Jon80] C. B. Jones. *Software Development: A Rigorous Approach*. Prentice Hall International, Englewood Cliffs, N.J., USA, 1980.

[Jon20] Cliff B. Jones. *Understanding Programming Languages*. Springer, 2020.

[Lau68] Peter E. Lauer. Formal definition of ALGOL 60. Technical Report 25.088, IBM Laboratory Vienna, 12 1968.

[LW69] Peter Lucas and Kurt Walk. On the formal description of PL/I. *Annual Review in Automatic Programming*, 6:105–182, 1969.

[Mon87] B. Q. Monahan. A Type Model for VDM. pages 210–236. 1987.

[Mos74] Peter David Mosses. The mathematical semantics of ALGOL 60. Technical report, Programming Research Group, 1 1974.

[Mos11] Peter D. Mosses. VDM semantics of programming languages: combinators and monads. *Formal Aspects of Computing*, 23(2):221–238, 2011.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, 1981.

[Plo04a] Gordon D. Plotkin. The origins of structural operational semantics. *Journal of Logic and Algebraic Programming*, 60–61:3–15, July–December 2004.

[Plo04b] Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60–61:17–139, July–December 2004.

Moses Schönfinkel and combinatory logic

Jonathan P. Bowen

December 2020

Based on a talk by Stephen Wolfram

I attended an online talk by Stephen Wolfram (2020d) celebrating the Russian logician Moses Schönfinkel (1888–1942). It was delivered exactly a century after a talk by Schönfinkel at the University of Göttingen in Germany, where he was a member of the group there headed by David Hilbert (1862–1943). Schönfinkel's talk in 1920 was entitled *Elemente der Logick* ("Elements of Logic"), where he summarised his foundational ideas on combinatory logic (Stanford, 2020), that he later published (Schönfinkel, 1924). Haskell Curry (1890–1982) also took up these ideas (Curry, 1927; 1930; Curry & Feys, 1958). This led to the term "currying", the technique of converting a function with multiple arguments into a sequence of functions each taking a single argument. This is important in lambda calculus (Cardone& Hindley, 2009), later mathematically modelled by Dana Scott in an early Programming Research Group Technical Monograph at Oxford (Scott, 1970). Subsequently, currying and lambda calculus gained importance in computer science, through functional programming.

Christopher Strachey (1916–1975) commented originally in lecture notes of 1967: "There is a device originated by Schönfinkel, for reducing operators with several operands to the successive application of single operand operators" (Strachey, 2000). Later, John Reynolds (1935–2013) commented: "In the last line we have used a trick called Currying (after the logician H. Curry) to solve the problem of introducing a binary operation into a language where all functions must accept a single argument." (Reynolds, 1998). The referee commented that although "Currying" is tastier, "Schönfinkeling" might be more accurate!

Stephen Wolfram's talk has an associated blog post on Moses Schönfinkel's contribution to the concept of combinators (Wolfram, 2020c) and other related blog posts (Wolfram, 2020a; 2020b). Wolfram and his team have undertaken

archival research on Schönfinkel's life. Sadly, not long after his contribution of combinatory logic, Schönfinkel later suffered from mental illness and spent the rest of his life in Moscow. On his death in poverty during 1942 at the time of World War II, his neighbours burnt his papers for heating. Despite his short period of research productivity, I believe that Moses Schönfinkel deserves to be better known for his contributions to computer science through combinatory logic. Stephen Wolfram's presentation was followed by an interesting online networking event, with participants able to roam around a virtual room and form parallel audio discussion groups (https://www.highfidelity.com), such are the rapid technological developments accelerated by the current pandemic.

Stephen Wolfram presenting on Zoom

Online networking afterwards

References

Cardone, F. and Hindley, J. R. (2009). Lambda-Calculus and Combinators in the 20th Century. In Gabbay, D. M. and Woods, J. (eds.), *Handbook of the History of Logic*, vol. 5, Logic from Russell to Church, pp. 723–817. North Holland, Elsevier. DOI: https://doi.org/10.1016/S1874-5857(09)70018-4

Curry, H. B. (1927). *Notes on Schönfinkel, Über die Bausteine der mathematischen Logik*, *1924*. T271128A. Haskell P. Curry papers, 1911–1984, Pennsylvania State University, USA, 28 November 1927. URL: https://libraries.psu.edu/findingaids/222.htm

Curry, H. B. (1930). Grundlagen der Kombinatorischen Logik ("Foundations of combinatorial logic"). American Journal of Mathematics, 52(3):509–536, July 1930. The Johns Hopkins University Press. DOI:

https://doi.org/10.2307/2370619

Curry, H. B. and Feys, R. (1958). Combinatory Logic, vol. 1. North-Holland Publishing Company.

Schönfinkel, M. (1924). Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92(3–4):305–316. DOI: https://doi.org/10.1007/bf01448013 (Translated into English: Bauer-Mengelberg, S. (1967). On the building blocks of mathematical logic. In van Heijenoort, J. (ed.), *A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press, pp. 355–366.)

Scott, D. (1970). *Outline of a Model of Computation*. Technical Monograph PRG-2, Oxford University Computing Laboratory, Programming Research Group, November 1970. URL: https://www.cs.ox.ac.uk/publications/publication3720-abstract.html

Stanford (2020). Combinatory Logic. *Stanford Encyclopedia of Philosophy*, Stanford University. USA, 14 November 2008 (revised 16 November 2020). URL: https://plato.stanford.edu/entries/logic-combinatory/

Strachey, C. (2000). Fundamental Concepts in Programming Languages. *Higher-Order and Symbolic Computation*, 13:11–49, April 2000. DOI: https://doi.org/10.1023/A:1010000313106

Reynolds, J. C. (1998). Definitional Interpreters for Higher-Order Programming Languages. Higher-Order and Symbolic Computation, 11(4):374, December 1998. DOI: https://doi.org/10.1023/A:1010027404223

Wolfram, S. (2020a). *Combinators: A Centennial View*. Stephen Wolfram Writings, 6 December 2020. URL: https://writings.stephenwolfram.com/2020/12/combinators-a-centennial-view/

Wolfram, S. (2020b). *Combinators and the Story of Computation*. Stephen Wolfram Writings, 7 December 2020. URL: https://writings.stephenwolfram.com/2020/12/combinators-and-the-story-of-computation/

Wolfram, S. (2020c). *Where Did Combinators Come From? Hunting the Story of Moses Schönfinkel*. Stephen Wolfram Writings, 7 December 2020. URL: https://writings.stephenwolfram.com/2020/12/where-did-combinators-come-from-hunting-the-story-of-moses-schonfinkel/

Wolfram, S. (2020d). *Combinators: A 100-Year Celebration*. YouTube, 7 December 2020. URL: https://www.youtube.com/watch?v=PG2G5xSz0NQ

Ken Robinson - A Remembrance

Kenneth Arthur (Ken) Robinson joined the staff at the University of New South Wales in the mid-1960s. Always a high achiever, Ken had been dux of primary school, dux of secondary school and had topped his joint Bachelor of Science/Bachelor of Engineering degree course at the University of Sydney.

When Ken moved to UNSW, he at first tutored Mathematics, but by 1968 he had become one of the three systems programmers in charge of the University's sole general-purpose computer — for which academics used keypunches the size of ironing boards to prepare their IBM-FORTRAN 80-column-card decks. They got one run a day, and their data were read from 7-megabyte multi-platter disks — like wedding cakes — carefully lowered and screwed tight into drives that were as big as washing machines. Undergraduates however were not allowed to use the keypunches. Instead they had to buy specially perforated 40-column cards, learn the letter-codes by heart and then poke-out the corresponding holes with a specially bent paperclip. For them, it was only one run a week — and usually resulted in a compile-time syntax error.

So that was where Ken's Computer-Science career started, at UNSW. He remained there until 2012 and passed away in late 2020. Ken was an engineer at heart, and so had the advantage of an extra perspective of rigour and precision. Over the many decades of his association with UNSW, he was variously lecturer, Head of the Department of Computer Science (later School of Computer Science and Engineering, CSE) and, as Associate Professor, the designer and force behind CSE's Software-Engineering Programme, the only such programme in Australia with a solid formal-methods core and furthermore one of the foremost of its kind in the world. Ken was the only engineer among

his colleagues, and yet, in spite of being surrounded by computer hardware and software, he still loved his pen and pencil — he saw them as craftsmen's tools.

Through all those years, Ken was the intellectual mentor of hundreds of students, many of whom considered him — at the time and also now — to have been the most inspiring teacher they had ever experienced. He had a habit of not quite giving a direct answer, so that the student had to invest some original thought as well. His lectures were not "performances" (except perhaps occasionally when he had to brush back his long, blond hair as he spoke) but were marked rather by his quiet incisiveness of thought and exposition, placing him among the first four academics at UNSW to receive the "Vice-Chancellor's Award for Teaching Excellence". Being in his class transmitted an infectious compulsion to absorb as much from him as one absolutely, possibly could — a kind of explosive exhilaration. One felt "**This** is how teaching should be."

Why was Ken so conspicuously excellent? In spite of Australia's distance from the rest of the computer-science world in the early 1970s: journals and conference proceedings arriving through the ordinary post, after months at sea; telephone calls even to neighbouring capital cities in Australia still needing to go via the operator; recent papers from the northern hemisphere requested by aerogramme with the paper arriving (perhaps) in a manila folder weeks later, sometimes addressed to the University of North South West. Nevertheless Ken Robinson put us — himself and his students — right at the very front of the astonishing new developments in formal methods and rigorous computing generally.

In 1971, Ken's front-line courses in computer science (in second year because then, at UNSW, Computer Science had no first-year courses), there was ALGOL-W (from Stanford), WATFOR (student FORTRAN from Waterloo), Plago (student PL/1 from Brooklyn), SNOBOL (originally from Bell Labs), and even IBM/360 assembly language — for the last, using an assembler that Ken wrote himself. (The IBM version was too slow and unforgiving for student use.) All that in one year.

From "load register from offset(base)" all the way up to Floyd-Hoare-Dijkstra high-level reasoning about programs at the source level: their abstractions, refinements, specifications, implementations. All of that was brought to UNSW, then promoted and further developed by Ken — who understood, and propagated the truth that to be effective and professional you really had to understand things from top to bottom, and back again. You can't abstract properly unless you know what you're abstracting from and (therefore) understand the cost of not doing so. And you can't refine unless you know the

environment you are aiming for. Ken worked and taught at the crossroads of elegance and practicality — the things that caught his eye had to be both beautiful and useful.

What about the very latest systematic compiler-writing methods? For those, there was Ken's fabulously popular third-year course, the first introduction of a compiler subject in Australia. And the astonishing PASCAL language, that mysteriously could compile its own compiler? Obtained directly from ETH Zürich by Ken who *again* was the first to teach it in Australia. And UNIX, a system recognised by Ken as being "too good to be true"? He wrote to Dennis Ritchie at Bell Labs to acquire a copy, which arrived in 1975. It made UNSW the first University outside the USA to run UNIX as a production facility. And functional programming? There too Ken was the first to use Miranda in Australia as a teaching language. And for the hard-core formalists, a course on Denotational Semantics? The text (Gordon) was "not available" directly from Sydney booksellers and "would take months" to obtain. Ken braved the international telephone operator and ordered 20 of them to be sent directly to Australia from Blackwell's in Oxford. By airmail.

Ken has often been described by his colleagues as "The Father of Formal Methods in Australia". The above shows why. However it was not all just one way: Ken's passion for sharing his expertise in teaching Formal Methods and Software Engineering extended in the other direction too, that is, from the South back to the North. He established a strong connection with Oxford University's Programming Research Group, and he helped many, many colleagues in the UK and France who were involved in the B community founded by J-R Abrial — now many of them professors themselves. He was limitlessly generous in sharing his resources and giving of his time to talk-through examples that would form good exercises, creating material that showed how he wanted students to learn in depth but also to be able to apply what they learned. He was keen that students should see B and Event-B systems as *practical* methods with *effective* tool support, as any engineering discipline must be. And so he incorporated the use of those systems into his Software-Engineering stream back in Australia, and made sure that code could be generated from the specifications with ease. He showed the students where formal methods could and should be used in a software-engineering lifecycle through group projects that enabled the students to see and *experience* their relevance to their own future careers. When the South and North were together at Formal-Methods conferences, Ken was always keen to hear how teaching was progressing "up there" and to learn from the student feedback.

And Ken always encouraged younger academics to care about their students and to take the time to find their own research strengths. He was always available to provide quiet mentoring, and his interest in their careers was long lasting.

As an engineer, Ken was especially passionate about computer science as a branch of engineering and crucially, for him, that included not only the social and methodological aspects of engineering, but also the "mathematics" (think differential equations for Electrical Engineering) that, for computing, was closer to logic. He was not a fan of what he called "high-level hacking" without a solid basis. As the methods and tools for reasoning about programs, and the systems made from them, continued to develop throughout the later '70s, then '80s and '90s, Ken contributed directly to the development and teaching of what is currently the "Rodin" toolkit for constructing, organising and proving-correct large-scale computer systems. Rodin's latest release has been dedicated to Ken. His "Software-Engineering Stream" within the UNSW computer-science curriculum started with a Requirements-Analysis project in First Year (Computer Science had a first year by then), followed by a formalisation of its specification in "Event B" (within Rodin) in Second Year, and then a rigorously controlled development trajectory from there via program refinement to actual running code, in Third Year — all of that built on the foundational work of J-R Abrial and others from the early 1980s.

Indeed, Ken was one of Rodin's very early adopters, and was not shy in sharing his views, both positive and negative, with the development team. The comments were always constructive, and they stemmed not only from his deep knowledge of model-based formal methods and refinement but also from his focus on effectiveness and usability of formal-methods tools. Many of Rodin's features resulted from email exchanges between Ken and the tool's main architects. His use of Rodin in teaching began around 2007. At the first Rodin User and Developer Workshop in Southampton in 2009, he gave an inspiring talk on how he had managed to incorporate formal modelling and proof, supported by Rodin, into the undergraduate software-engineering courses at UNSW. The key message from that talk was the importance of good design and how abstraction and refinement support it. It was very encouraging to see an affirmation of the software-engineering value of tools like Rodin by someone as respected as Ken.

On the more personal side, Ken had an insatiable curiosity about the world, an endless desire to find out more about everything — whether it was how things worked, how people thought or just ideas, history, films, music... and more.

Ideas were most important: juggling viewpoints, and finding different ways of approaching problems. His life at university was of course a natural extension of that. Education was one of Ken's strongest passions and it was through teaching that he transferred his enthusiasm for whatever it was. It was important to him to encourage this curiosity, this thirst for knowledge, in others.

Although Ken dedicated a huge part of his life to the university, to computer science and software engineering, he also had a deep concern for humanity and for the state of the world generally. He was a person of enormous generosity, with whom it was immediately possible to fall into a simple friendship that seemed somehow to have existed already. People were very important to Ken. Those lucky enough to know Ken personally often mention the perpetual twinkle in his eye, the feeling when speaking with him that a smile was always about to break out. He cared deeply about others who were less privileged. He was a determined advocate for the rights of everyone. He was a person of strong principles, and so it was important to him to remain true to those things he valued. It didn't matter too much what others thought — it was more important to stand up for what you yourself believed in.

Ken was down to earth with absolutely no pretensions, and indeed had an intolerance for those valuing status in whatever form. In the early days, much to the horror of many, he refused to stand for the Australian national anthem at the time, "God save the Queen". (In Australia it used to be played at the end of concerts, and in cinemas before the beginning of every film!) The frills and glitz of life were irrelevant to him. His working life at university reflected these values and his students were his equals: "Just call me 'Ken'!"

For Ken everything had to be done, in his words, "properly". He was a perfectionist, and there was always more work to be done on improving whatever tasks were at hand. And yet at the same time Ken was unbelievably untidy, being described by colleagues (with some understatement) as "a little messy". But that was just part of his concentrating on what the important things really were. His dishevelled room at UNSW was well known, and in one story he related how the police arrived to investigate a robbery in the building. Apparently they were sure his room had been ransacked. It hadn't: it was just in its "normal" state, filled with the clutter of ever-increasing papers and student assignments — just as it was at his home, where once the babysitter couldn't find the telephone buried beneath mounds of books and papers!

Further outside of academic life, Ken was an explorer of almost everything, with an enthusiasm that carried along with him anyone within range — very often his

whole family of five: from bushwalking, to cycling, to woodwork, to photography, to all five living (temporarily and including a 4-week old baby) in a van during a sabbatical in the UK until better accommodation could be found. He loved the outdoors, from where many a funny, and sometimes not-so-funny family story would emerge. There was rarely a bushwalk where the family didn't get lost taking one of Ken's so-called "short cuts". But this was — of course — just part of his finding new and different ways of doing things!

In his early days Ken spent many a weekend fixing and polishing his MG-TD as well as playing the clarinet. He was very musical, and his appreciation extended from early music right through to the contemporary. Bach was a particular favourite, anything from the Amsterdam Loeki Stardust Recorder Quartet's interpretation of The Art of Fugue ("As precise as a pipe organ!") to, in more recent years, humming along with Glen Gould playing the Goldberg Variations (which the family included in his funeral service). He also on occasion sang in the chorus for the annual Handel's Messiah at the Sydney Opera House. Shakespeare was another favourite, whether it was reading his plays or attending performances. Among Ken's other passions were cooking and bread-making. Apart from his much-loved Thai and Indian curries, his extraordinary home-made bread -a "perfected" recipe- won many prizes at the annual Sydney Easter Show.

Many, many people would have led different lives had it not been for Ken Robinson: so many dimensions in the one person, so much to miss — and so much for us all to remember and to celebrate.

This article's contributors included:

> Jonathan Bowen, Michael Butler, Paul Compton, Ian Hayes, Martin de Groot, Peter Ho, Son Thai Hoang, Carroll Morgan, Maurice Pagnucco, Aaron Quigley, Rosalie Robinson, Claude Sammut, Steve Schneider, Martin Schwenke, Arcot Somya, Bill Stoddart, Jeff Tobias and Helen Treharne.

Peter Landin Semantics Lecture

ALGOL 60 @ 60: Its Place in Formal Semantics

Tim Denvir and Troy Astarte

3 December 2020, 6.00pm - 7:30pm via Zoom

Reported by: John V Tucker
Dept. of Computer Science and History of Computing Collection
Swansea University

ALGOL is 60 years old in 2020. This event before Christmas featured a pair of talks that celebrated the programming language and reflected on its formal aspects and legacy. The subject was well suited for a Peter Landin Lecture, for Landin had thought hard about semantics and ALGOL. Our speakers were ideally suited to the task of delivering it: you may have read Tim's article marking the anniversary in this Newsletter earlier in the year – Denvir (2020) – or Troy's research for his recent PhD thesis – Astarte (2018, 2019). Several pioneers attended whose work would figure in their lectures. The audience was not confined to the UK, with notables from The Netherlands and USA.

ALGOL 60 was an elegant high-level language with a very precise general definition for its time. It became a direct inspiration for many languages which followed it, such as Pascal, Simula, ALGOL 68, C, Java and others. The language was invented in stages, starting with the International Algorithmic Language (IAL), later called ALGOL 58 and announced in the CACM, where its syntax was specified in Backus Normal Form. The language's mature form was specified in the *Revised Report on the Algorithmic Language ALGOL 60* – see Figure 1.

It is clear that the whole project was guided by formal considerations, even if there was no attempt at giving a formal semantics for the constructs. Actually, Peter Landin was one of several people who made an early formal semantics for ALGOL 60: Landin's was published in two parts in February and March 1965, in Communications of the ACM (Volume 8). These attempts were early contributions to the emerging subject of formal semantics for the specification of language constructs.

Tim's and Troy's talks described the history of the language and the background of its designers, surveyed semantic descriptions of ALGOL 60, and discussed the impact and influence of ALGOL 60. But the aim of this report is simply to recommend you view these excellent lectures on this important topic, which are available on video along with the slides (see below).

The Two Talks

Tim began with a detailed account of the emergence of the Revised Report, covering drafts, meetings, the ALGOL Bulletin, and personalities. He introduced the authors and sketched their subsequent careers. Most are well-known to people of a certain age and to bibliophiles: Naur, Bauer, Wijngaarden, Samelson, Rutishauser, Woodger, Backus, Perlis, McCarthy, Wegstein; and one or two not so: Julien Green, Charlie Katz.

Next came the international take-up of the language and the first compilers, authored by: Dijkstra and Zonneveld for the Electrologica X1; Randell and Russell for DEUCE and KDF-9; Hoare for the Elliott 803; Hopgood and Bell for the supercomputer, Atlas; Abbott for the PDP8. More than 20 were to follow in the decade.

Tim's section ended with Landin. Rigorous mathematical models of assembly programs had appeared in the computability literature, notably in Shepherdson and Sturgis (1963). But ALGOL's natural higher-level algorithmic elegance suggested a need for a whole new level of modelling. It was the use of procedures that most obviously needed deeper semantical understanding. For this, Peter Landin's semantics drew attention to ALGOL's relation to Church's λ-calculus. These first steps give an early hint of what was to come: various informal notions of call by this-or-that mechanism, and various kinds of semantical models, mathematical structures and fixed-point methods for solving recursion equations.

Troy's talk began by giving us an impression of the atmosphere and new thinking about programming that was still emerging in the period. He drew attention to the Oxford Summer Schools on computing organised by Leslie Fox, which Christopher Strachey and others complained about in *The Computer Journal* because of its exclusive focus on applications (with PDEs and Crystallography). This resulted in the Third Summer School in 1963 being devoted to Programming and Non-numerical Computation – see, e.g., the proceedings Fox (1966) and papers by theoretical pioneers like Strachey, Landin and Cooper. Troy then turned to the significant of role of ALGOL in emergence of formal description of languages, ranging from McCarthy's MicroAlgol, Peter Lauer's description via abstract machines, and the growth of the Vienna Definition Language under Zemanek. IBM Vienna took on PL/1 in 1964. These formative studies at Vienna, and later IBM Hursley, were taken much further by Cliff Jones, Dines Bjørner and others. The emergence of denotational semantics under Strachey and its early mathematical theory developed by Scott, de Bakker and others brought us to the end of the decade.

The *Revised Report* was the basis of a denotational semantics for ALGOL given by Peter Mosses in his 1974 doctoral thesis for Strachey. The lecturers give references for all the technical points and the broader historical issues in their slides.

In the questions, attention was drawn to modern work by Peter Mosses and Cliff Jones on collecting, and making available in a modular way, semantic models, including historical models, of programming constructs (see below).

Concluding Remarks

The fascination with ALGOL is easy to understand. For example, it became and remains the inspiration and style of pseudocode. Its constructs constitute the core of imperative models of programming essential for the education of students, and for experiments with novel theories. And so many of us met ALGOL early on and were smitten. For me, it was ALGOL on an Elliott at Warwick University in 1970; years later, I saw its influence in Oslo and, at the Mathematical Centre Amsterdam; I saw the ALGOL 68 group disperse to go on to other things, and van Wijngaarden's retirement. When I first taught formal languages, I used that excellent celebration of ALGOL syntax, Backhouse (1979). We should ensure its preservation as an important item of intellectual heritage. As Tony Hoare commented most famously in Hoare (1973):

> "Here is a language so far ahead of its time, that it was not only an improvement on its predecessors, but also on nearly all its successors."

Links

You can watch a video of the lecture:

https://www.youtube.com/watch?v=7NVWaYmVNjo&feature=youtu.be

You can download the slides here:

https://www.bcs.org/events/2020/december/webinar-bcs-facs-2020-agm

The *PlanComp Project* of Peter Mosses, Cliff Jones, Adrian Johnstone and Elizabeth Scott is here:

https://plancomps.csle.cs.rhul.ac.uk

Their semantic library is here:

http://homepages.cs.ncl.ac.uk/cliff.jones/semantics-library

Tim Denvir presenting on Zoom

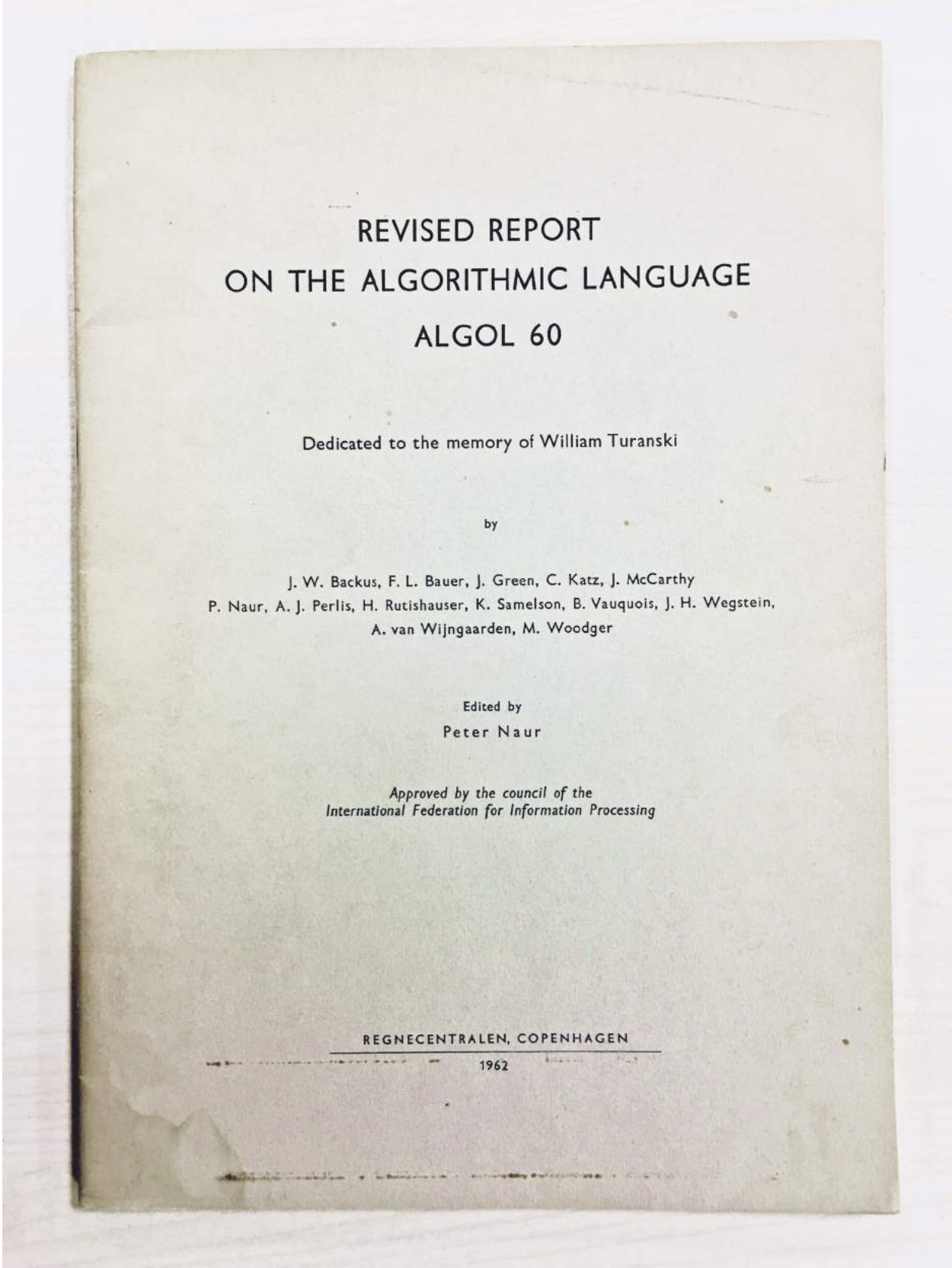
(Screenshot by Jonathan Bowen)

Troy Astarte presenting on Zoom

(Screenshot by Jonathan Bowen)

Cliff Jones and Peter Mosses

(Photo: History of Computing Collection)



REVISED REPORT
ON THE ALGORITHMIC LANGUAGE
ALGOL 60

Dedicated to the memory of William Turanski

by

J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy
P. Naur, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein,
A. van Wijngaarden, M. Woodger

Edited by
Peter Naur

Approved by the council of the
International Federation for Information Processing

REGNECENTRALEN, COPENHAGEN
1962

References

Troy K Astarte and Cliff B. Jones, Formal Semantics of ALGOL 60: Four Descriptions in their Historical Context. In: Liesbeth De Mol and Giuseppe Primiero (eds), *Reflections on Programming Systems - Historical and Philosophical Aspects*, Springer Philosophical Studies Series, 2018, pp. 71–141.

Troy K Astarte, *Formalising meaning: a history of programming language semantics*. PhD thesis, Newcastle University, June 2019.

Roland Backhouse, *Syntax of Programming Languages: Theory and Practice*, Prentice Hall, 1979.

Tim Denvir, ALGOL 60 @ 60, *FACS FACTS Newsletter*, June 2020, pp. 7-12. https://www.bcs.org/media/5842/facs-jun20.pdf

Leslie Fox (ed), *Advances in Programming and Non-numerical Computation*, Pergamon Press, 1966.

C A R Hoare, *Hints On Programming Language Design*, Stanford Computer Science Department, Report STAN-CS-73-403, 1973.

J C Shepherdson and H. E. Sturgis, Computability of recursive functions, *J. Association for Computing Machinery*, 10 (1963), 217–255.

ALGOL 60 @ Oxford

Jonathan P. Bowen

When I heard about Tim Denvir's suggestion of an *ALGOL 60 @ 60* talk for FACS's 2020 Peter Landin Semantics Seminar (see the report by John Tucker in this issue of the *FACS FACTS* newsletter), I was alerted to look out for any reference to ALGOL 60 in anything that I read.

I happened to have a copy of the excellent 2013 memoir *An Appetite for Wonder* by the zoologist Richard Dawkins, purchased from the also excellent Oxfam bookshop in St Giles', Oxford. Although this book does not include ALGOL 60 in the index, and I would not expect it to do so, it does include a chapter entitled *Computer Fix*. It seems that Dawkins was something of a computer enthusiast (indeed a self-described "addict") in his youth.

On his return from Berkeley, California, in 1969 to Oxford, he was based in the Animal Behaviour Research Group of the Zoology Department at 13 Bevington Road (now the African Studies Centre) in North Oxford, with access to a PDP-8 computer, and became the programming expert there.

His first project on the computer was "Dawkins Organ", a system where animal behaviour could be recorded in the field with a keyboard and tape recorder, later to be analysed using the computer. He later invented his own programming language *BEVPAL*, presumably a reference to Bevington Road. He also devised an interpreted language called *SysGen* with (at least conceptually) simultaneously executed statements.

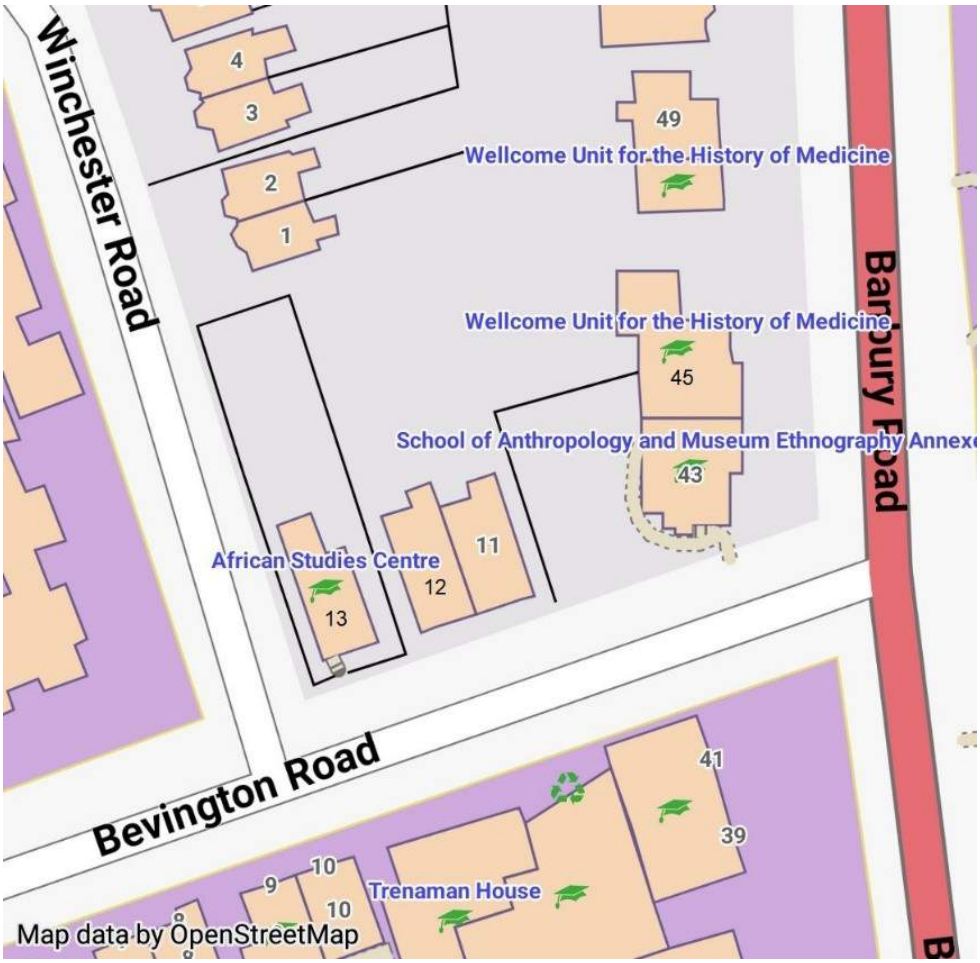
He investigated Chomsky hierarchy grammars, writing a program to generate random sentences. He gained an interest in recursive procedures and used an ALGOL 60 compiler on the PDP-8, written by the engineer Roger Abbott (who sadly died age 60 in 2002), also based at Oxford. This allowed Dawkins to write programs producing correct grammars recursively using procedures with names like *PrepositionalClause*, *RelativeClause*, etc., potentially calling any other procedure, or even themselves. This helped with his understanding of syntax and semantics.

Meantime, literally just around the corner, in 1965 Christopher Strachey (1916–1975) had established the Programming Research Group (PRG) at 45 Banbury Road (see map, now part of the Faculty of History), with an interest in formal semantics and developing denotational semantics with Dana Scott. There is no

evidence of Dawkins and Strachey being aware of each other's research, despite their proximity. It would have perhaps been interesting if they had collaborated.

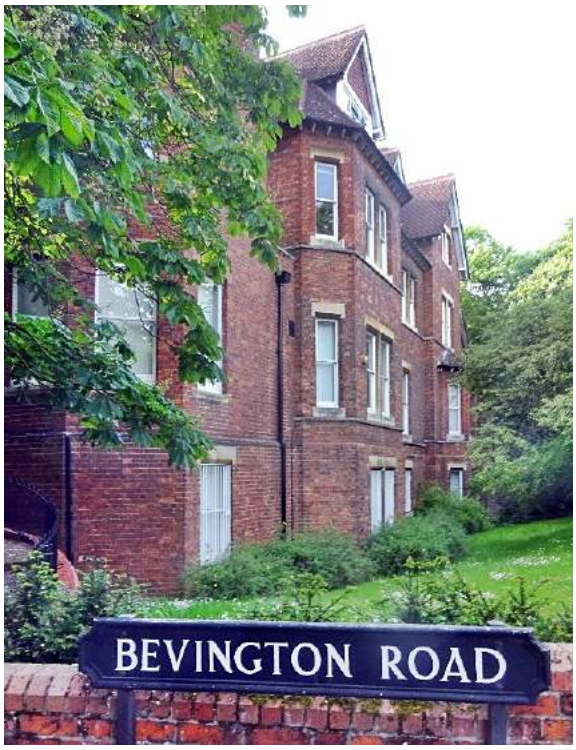
Meanwhile I was living further up Winchester Road, north of Bevington Road, as a schoolboy and then attending engineering tutorials in 43 Banbury Road (adjoining no. 45) as a student, also completely unaware of any of these activities going on around me. My first experience of programming while still at school was using ALGOL 60 on paper tape. But my main passion was in electronics until my interest in programming was rekindled in my first industrial job using PDP-11 computers.

Eventually I moved back to Oxford to work as a Research Officer on a project involving formal methods at the PRG, by then under the leadership of Tony Hoare at 8-11 Keble Road, not far from its original location. Such are the ways of serendipity!

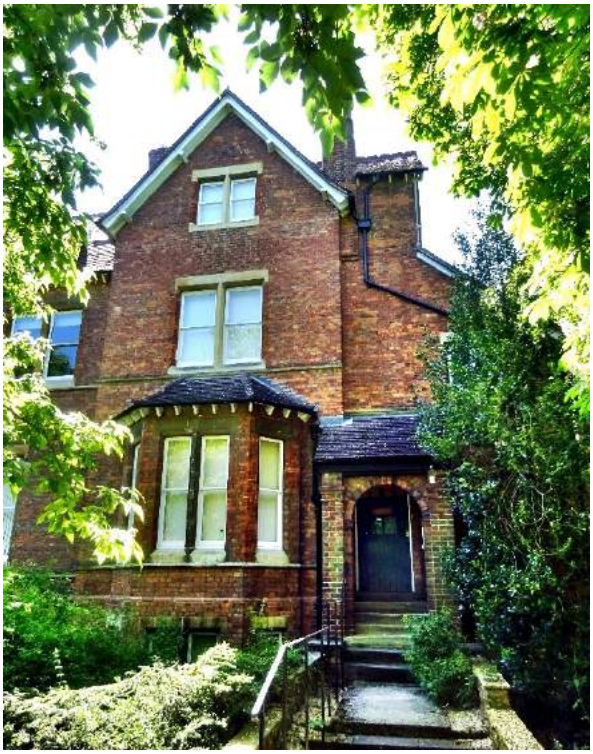


Map showing the proximity of 13 Bevington Road
to 43 and 45 Banbury Road

13 Bevington Road, on the corner of Winchester Road



43 & 45 Banbury Road



45 Banbury Road

Online Resources

Abbot, R. *An efficient ALGOL-60 system for the PDP8*. A.R.C. Unit of Muscle Mechanisms & Insect Physiology, Department of Zoology, University of Oxford, South Parks Road, Oxford. URL: http://pdp8.de/download/RogAlgol.pdf

Obituaries: Roger Abbott (d. 2002). Oxford University Society of Change Ringers. URL: https://ouscr.org.uk/index.php/obituaries?id=54

ROGALGOL ALGOL-60 Compiler. Manual, Internet Archive. URL: https://archive.org/details/hack42_ROG_ALGOL_Compiler/mode/2up

ROGALGOL ALGOL-60 Compiler. Source code, bitsavers.org. URL: http://www.bitsavers.org/bits/DEC/pdp8/papertapeImages/russ.ucs.indiana.edu/Langs/Algol/

Dear X.

You may recall a certain late-night (say-no-more) chat over a few (!) glasses of Glen-something-or-other, in which the subject arose of our old friend (?) F. X. Reid. Had he really (as some have darkly averred) defected eastward? Had he taken over some spurious political or religious sect? Had he retired to some secluded nook to translate Finnegans Wake into demotic Greek (as he had long threatened to do)? You may remember that our speculations become more and more outrageous as we glass followed glass and night glimmered into day.

But the truth, sad to say, is far more mundane.

It was while I was visiting my aunt in Xlendi ('X' is pronounced 'sh', here) that I first caught wind of F. X. Reid. A strange, dishevelled man was apparently squatting on a soap box in Valletta ranting that adherents of interleaving semantics were disseminating a false perspective on

Concurrency Theory. Not only are they not right, they aren't even wrong (allegedly). What more could my aunt tell me?

Facts were unfortunately hard to come by, Reid was being held incommunicado, somewhere near Balzan by his 'secretary', Dr. F. X. Lurk, and is only seldom to be seen 'free'. It is on these occasions that the populace are treated to his denunciation of Algebraic process methods, of the so-called Oxford Mafia, of the Glitch phenomenon, of G'd knows what!

But what is really going on? Only the shadowy F. X. Lurk would seem to have the answers. Suffice it to say that there is nothing of which he has not been accused and there is no evidence of any of it.

Will there be more? Who knows?

Forthcoming Events

Events Venue (unless otherwise specified):

Online using Zoom until further notice.

Physical talks, when they can resume, will be at:

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.

Thursday, 6th May 5:15 pm	**FACS Evening Seminar – with Formal Methods Europe** **Speaker**: Michael Leuschel https://www.bcs.org/events/2021/may/webinar-evening-seminar-facs-sg/

Details of all forthcoming events can be found online here:

https://facs.bcs.org

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

Rob Hierons
LMS Liaison

Ana Cavalcanti
FME Liaison

Brijesh Dongol
Refinement
Workshop Liaison

Keith Lines
Government and
Standards Liaison

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.

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 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 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.