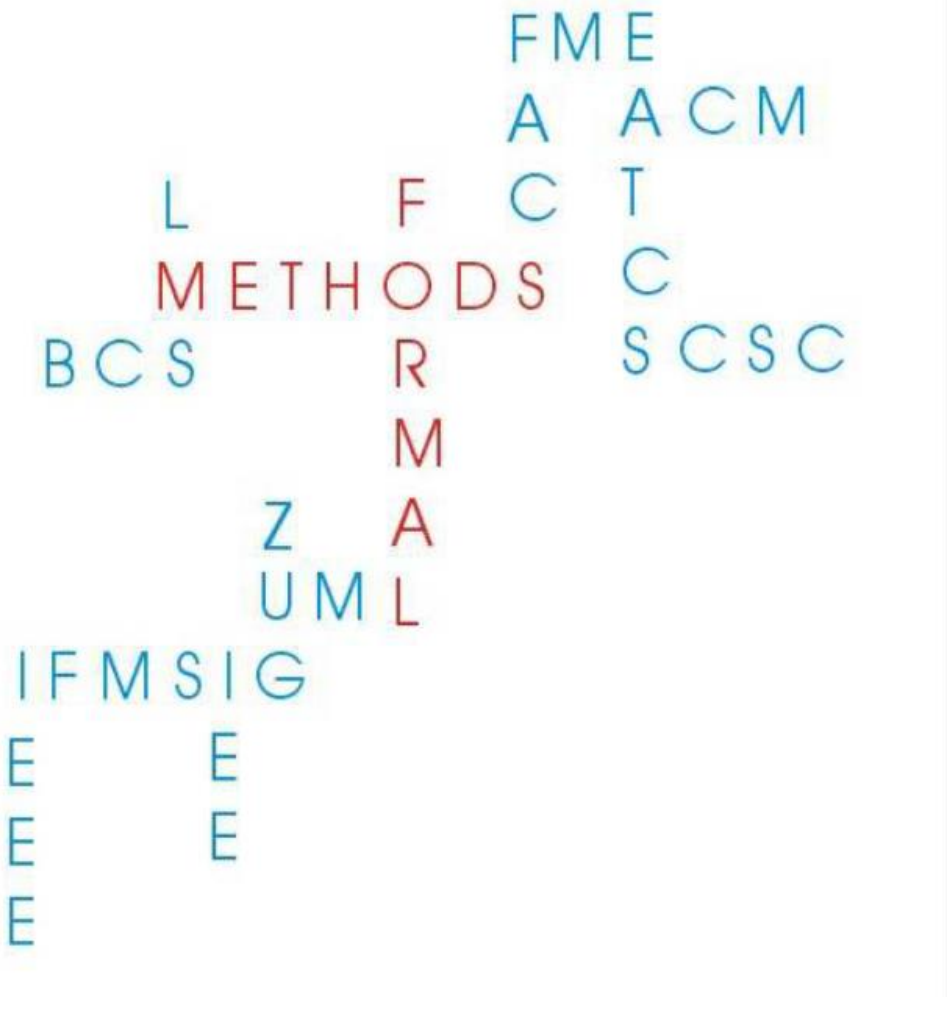
Issue 2023-1
January 2023

FACS

A C T S



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:

Andrei Popescu, Jonathan Bowen, Tim Denvir, Cliff Jones

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 at jonathan.bowen@lsbu.ac.uk.

Editorial

Dear reader,

In this issue we have the Annual Report for 2022 from our chairman, Jonathan Bowen. We have two reports on meetings, from Andrei Popescu on the annual LMS-FACS seminar given by Sam Staton, and from Tim Denvir on the annual Peter Landin seminar given by Nicolas Wu. Then there is a letter from Cliff Jones commenting on the article by John Tucker on PL/I in our previous issue, and a book review from Jonathan Bowen on Steven Wolfram's *Twenty Years of A New Kind of Science*.

Over the past few years, there has been some discussion of the notion of "algorithm" with reference to machine learning. There are now many applications of data science and of systems whose behaviour has been determined through a process of *learning by example*. From an applied mathematical point of view, there is little outward difference (if any) between systems produced using machine learning from a curated data set, and systems produced by using more traditional programming techniques. Both kinds will be systems which operate by giving responses to inputs and both can be deployed in a multitude of ways.

However, there is clearly a huge difference in terms of assessment and assurance. Machine-learned systems do not possess a readily analysable schema of behaviour from which its behaviour can be assessed and assurance gained through an explicit analysis – there is no "program" as such to examine or walkthrough. To see what happens with such a system, it seems necessary to just run the system in situ. Obviously, it is then difficult to be sure that every possible behaviour has been covered!

In many ways, trying to understand how a machine learnt system behaves is a bit like having a computer system whose source code has been hidden and made unavailable. However, it's perhaps even worse than that, since a conventional system will typically involve standard libraries performing familiar tasks – but a machine learnt system does not necessarily possess *any* structural content that conventionally encodes intention. The form of such a program is purely a stimulus-response network without any specific intentionality whatsoever!

Perhaps a good question to ask, then, is what does "correctness" look like in the world of software produced via machine learning? In some sense, "correctness" surely becomes trivialised – the learned behaviour is whatever the training program produces, pure and simple. To get a different series of behaviours and outcomes, the training data would then need to be changed "*in some way*".

Perhaps a more interesting question concerns how to gain *assurance*,

confidence and *trust* in systems produced by using machine learning. The natural approach to providing such assurance would appear to involve constructing audit trails, detailed monitoring of inputs, and the *tracing* through of the decisions made that produce responses and outputs – all in the attempt to provide *explanations*. How do we produce systems which can *introspect* and then explain and report their own behaviour on demand?

This is all very well, but what then would be done with all this additional justification information? Even if these "explanations" could be made explicitly representable, these would likely need even further processing, scrutiny and analysis before coming to any conclusion. Inevitably, any "final" step here still looks like *taking complex decisions based upon observations* – except that it is an even more complex form of decision problem than was had initially! On the face of it, an ever-widening tower of decision problems would seemingly become apparent, each more complex than the previous one.

The conventional approach to solving this "systems integrity" conundrum is that, in the absence of absolute guarantees, each system, whether machine-learned or not, needs to have some *independent design justification*, together with some *separate means* for openly assessing (the range of) its behaviour, demonstrating how well the system acts in practice.

In some cases, this could take the form of an *active monitor* (e.g., akin to a "Black Box Flight Recorder" or audit trail) – in other cases, this might take the form of a *mathematical argument* or *proof*. However, in many cases, it would surely involve some pragmatic mixture of these, and similar approaches. Without something of this nature, it is difficult to see how system behaviour could, in practice, attain widespread assurance, confidence, and trust in a tangible manner.

Despite many years of research and development, issues such as these continue to be quite challenging for system design and integrity, to put it mildly! One can but wonder how these and similar issues will be addressed in future – but one thing is certain, those answers still await us in practice.

With this being said, we do hugely appreciate and look forward to your contributions, including comments, from you, our readers.

We hope you enjoy *FACS FACTS* issue 2023-1.

Tim Denvir
Brian Monahan

Tim Denvir receiving his BCS Certificate of Appreciation from the FACS Chair, Jonathan Bowen, before the 2022 Peter Landin Semantic Seminar, for his long service to the FACS Specialist Group. Photograph by Alvaro Miyazawa.

Table of Contents

BCS-FACS Specialist Group 2022 Chair's Report

Member Group Name:	FACS Specialist Group
Year:	2022
Report By:	Jonathan Bowen

Group Chair:	Jonathan Bowen
Group Treasurer:	John Cooke
Group Secretary:	Roger Carsley
Group Inclusion Officer:	Margaret West
Other Committee Members:	Ana Cavalcanti (FME Liaison), Tim Denvir (*FACS FACTS* newsletter co-editor), Brijesh Dongol (Refinement Workshop Liaison), Keith Lines (Government and Standards Liaison), Brian Monahan (*FACS FACTS* newsletter co-editor), Andrei Popescu (LMS Liaison).

Successes

Success	Additional Comments
1. Continued evening seminars online, with recordings on YouTube	The BCS Zoom facilities and recording transfer to YouTube have widened access to FACS seminars. Thank you to Keith Lines, Andrei Popescu, and the Chair for help with organising four FACS seminars in 2022.
2. Publication of *FACS FACTS* newsletters by BCS-FACS and *Formal Aspects of Computing* journal by ACM	We aim for two major newsletters each year, published online in PDF format. Thank you to Tim Denvir and Brian Monahan for continuing with their excellent editing of the two 2022 newsletters. Contributions by FACS members are always welcome. The associated FAC journal has been adopted by ACM with open access to papers this year. See https://dl.acm.org/journal/fac
3. Move to hybrid evening seminars	In 2022, one evening seminar in June was delivered in hybrid mode and our major Peter Landin Semantics Seminar at the BCS London office (December) is being delivered in hybrid format after the AGM.

Plans

Planned Activity	Additional Comments
1. Continued online and some hybrid evening seminars	The BCS facilities for online and hybrid talks enable these modes of delivery. Alvaro Miyazawa of the University of York has volunteered to take on the currently vacant role of FACS Seminar Organiser.
2. At least two *FACS FACTS* newsletters	We aim for early and mid-year as times of publication.
3. Collaboration with related organizations such as Formal Methods Europe (FME), London Mathematical Society (LMS), and newly the Edinburgh Mathematical Society.	Note that we have a new LMS Liaison Officer for 2022, Andrei Popescu. Tim Denvir is liaising with the Edinburgh Mathematical Society on possible joint seminars in the future.

Impediments

Impediment	Description
1. Lack of volunteers to organise evening seminars	This has mainly fallen to Keith Lines, Andrei Popescu, and the Chair in 2022 but the position of Seminar Organiser will be filled by Alvaro Miyazawa of the University of York in 2023.
2. Reduced BCS funding	Our funding from the BCS has been reduced by 64% for the 2022/23 financial year. Thus, our pre-Covid aim of around six physical seminars per year will realistically be reduced to two physical events per year. That said, online seminars are much more cost-neutral, so we can continue with these with less impediment.
3. Reduced physical meetings	The lack of physical meetings impedes networking of FACS members. On the other hand, online seminars are popular due to the reduced cost for attendees and also a wider national and international reach.

Additional Facts and Figures

We aim for at least two *FACS FACTS* newsletters per year (with two in 2022, in January and July). We also aim for four to six online/hybrid evening seminars per year (with four in 2022, two online and two hybrid).

Further Comments

For the record, FACS organised the following evening seminars during 2022:

1. *Dependent Types for Practical Use*, by **Andre Videla**, University of Strathclyde, 29 March. Online event.
2. *Alan Turing at 110 – and at Oxford!*, by **Jonathan Bowen**, London South Bank University, 24 June. Hybrid event.
3. *Programming-based Foundations for Statistics*, by **Sam Staton**, University of Oxford, 17 November. In association with the London Mathematical Society (LMS), at the LMS headquarters. Online event.
4. *The Next 700 Domain Specific Languages*, by **Nicolas Wu**, Imperial College London, 6 December. The annual Peter Landin Semantics Seminar, in association with the FACS AGM, at the BCS London office. Hybrid event.

Thank you to all the FACS committee members for performing their various roles, as detailed above. New committee members and new ideas for activities, collaborations etc. are very welcome, especially if interested in co-organising events.

Programming-based Foundations for Statistics

Professor Sam Staton

University of Oxford

https://www.cs.ox.ac.uk/people/samuel.staton/main.html

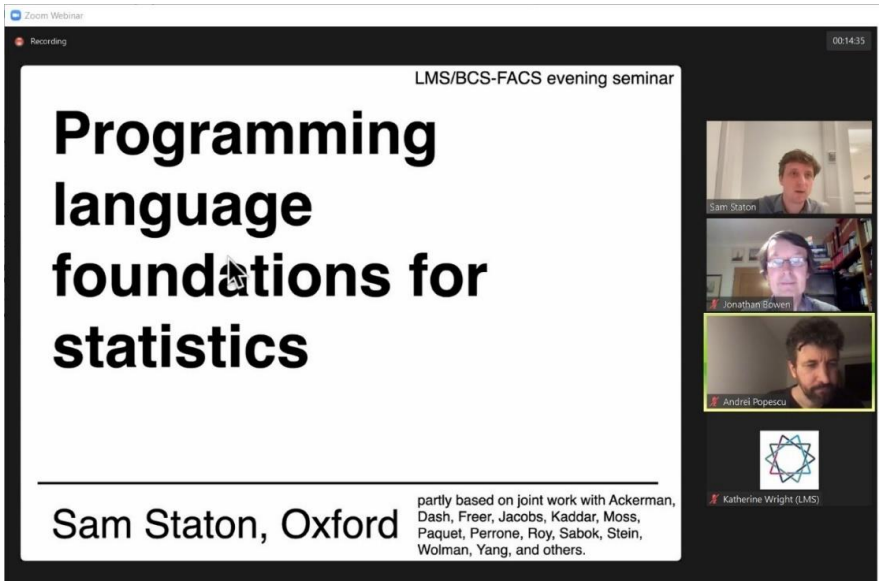
Thursday, 17 November 2022
Venue: Online via Zoom

Reported by: Andrei Popescu

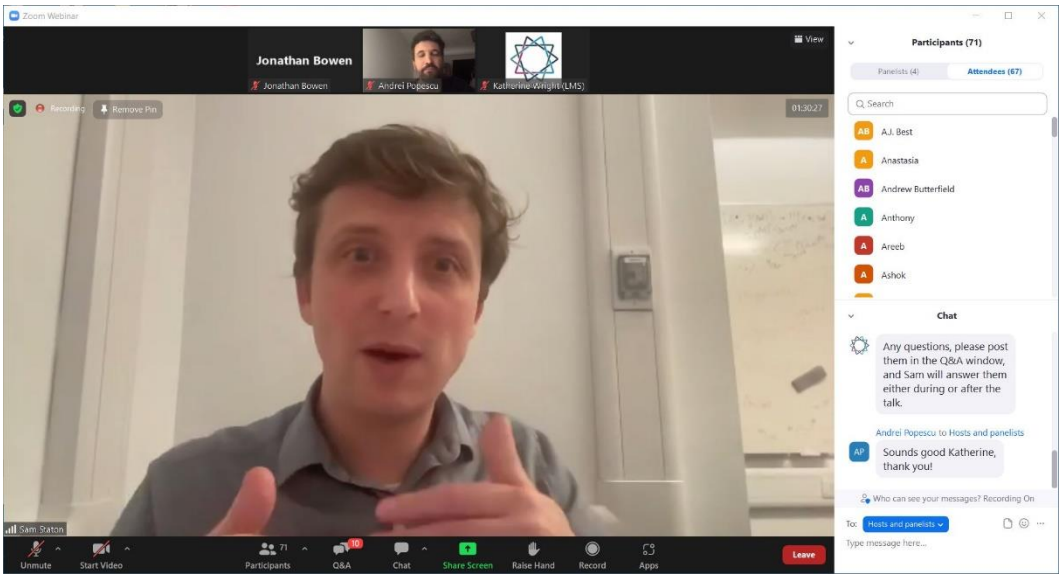
Background

Each year, FACS together with the Computer Science Committee of the London Mathematical Society host an evening seminar with a talk by a leading expert in an area lying at the intersection of computer science and mathematics.

This year, we had the pleasure of having Sam Staton give the talk. Sam is a Professor of Computer Science and Royal Society University Research Fellow at the University of Oxford. There he currently runs an ERC grant "Better Languages for Statistics". Before arriving in Oxford in 2015, Sam spent time in Nijmegen, Paris, and Cambridge. His PhD was in Cambridge with Marcelo Fiore (2007). Sam's main research is in programming language theory, but he is also interested in logic and category theory. He has recent contributions in probabilistic programming languages, and quantum computing and programming languages.



Sam Staton's opening presentation slide on Zoom.
Screenshot by Jonathan Bowen.



Sam Staton answering questions on Zoom.
Screenshot by Jonathan Bowen.

Participation

The talk took place online via Zoom. There were 196 registrants, and 100 live attendees.

Summary of the talk

Sam's talk focussed on the benefits that concepts from programming languages can bring to the statistics community. Two distinctions around which Sam organized his talk were (1) between high-level and low-level aspects of programming languages, and (2) between the engineering perspective and the foundational perspective. He used a wealth of examples, as well as several references to theoretical foundational results by himself and others.

Sam explained how concepts from functional programming such as types, monads and the Curry-Howard correspondence are useful for organizing concepts from statistics. Using examples from Monte Carlo simulation and regression analysis, he argued that higher-order programming is an elegant way of manipulating parameterized distributions and random functions. On the foundational side, Sam recalled a classic result by Aumann stating that the category of measurable spaces, which offer the traditional foundations for probability, does not have well-behaved function spaces (is not cartesian closed). He discussed his solution based on quasi-Borel spaces, which do not have this problem, and provide a semantics to probabilistic programs that enjoys the desired dataflow properties. (He also pointed to other solutions from the literature). On the way, Sam emphasized the connections between programming language concepts and classic results in measure and probability theory, such as dataflow-preserving reordering and Fubini's theorem, or the Haskell-style "repeat" construct and Kolmogorov's extension theorem.

Finally, Sam went on to talk about the role of symmetries concerning names in statistics, using as starting points two well-known examples of stochastic processes that exhibit such symmetries: the Chinese restaurant and the Indian buffet processes. He illustrated the passage between combinatorial models emerging from programs and traditional measure-theoretic models, showing how this can inform the process of feature extraction. Sam pointed out to one of his recent papers, where he describes this passage abstractly as a functor from the category of (Gabbay-Pitts) nominal sets to that of measurable spaces. The takeaway message was that by thinking programmatically about new symmetries, one can discover new interesting statistical models -- opening up exciting directions of research.

Sam concluded the talk by reiterating that programming language and formal methods ideas can contribute to the better understanding and the easier manipulation of statistical models. A lively Q&A session ended the evening.

Acknowledgments

We thank the London Mathematical Society (LMS), and especially Katherine Wright (the LMS business, research & communications officer) and the LMS computer science committee, for advertising the event, providing the zoom infrastructure, and offering technical support during the webinar.

Bibliography

Kenta Cho, Bart Jacobs:
Disintegration and Bayesian inversion via string diagrams. Math. Struct. Comput. Sci. 29(7): 938-971 (2019).

Bart Jacobs, Sam Staton:
De Finetti's Construction as a Categorical Limit. CMCS 2020: 90-111.

Fredrik Dahlqvist, Dexter Kozen:
Semantics of higher-order probabilistic programs with conditioning. Proc. ACM Program. Lang. 4(POPL): 57:1-57:29 (2020).

Swaraj Dash, Younesse Kaddar, Hugo Paquet, Sam Staton: *Affine monads and lazy structures for Bayesian programming.* To appear in Principles of Programming Languages (POPL) 2023.

Daniel J. Navarro, Thomas L. Griffiths:
A Nonparametric Bayesian Method for Inferring Features From Similarity Judgments. NIPS 2006: 1033-1040.

Tobias Fritz, Paolo Perrone: *Stochastic order on metric spaces and the ordered Kantorovich monad.* Advances in Mathematics, vol. 366, 2020.

Chris Heunen, Ohad Kammar, Sam Staton, Hongseok Yang:
A convenient category for higher-order probability theory. LICS 2017: 1-12

Xiaodong Jia, Bert Lindenhovius, Michael W. Mislove, Vladimir Zamdzhiev:
Commutative Monads for Probabilistic Programming Languages. LICS 2021: 1-14.

Cristina Matache, Sean K. Moss, Sam Staton:
Concrete categories and higher-order recursion: With applications including probability, differentiability, and full abstraction. LICS 2022: 57:1-57:14.

Marcin Sabok, Sam Staton, Dario Stein, Michael Wolman:
Probabilistic programming semantics for name generation. Proc. ACM Program. Lang. 5(POPL): 1-29 (2021)

Peter Landin Semantics Seminar 2022

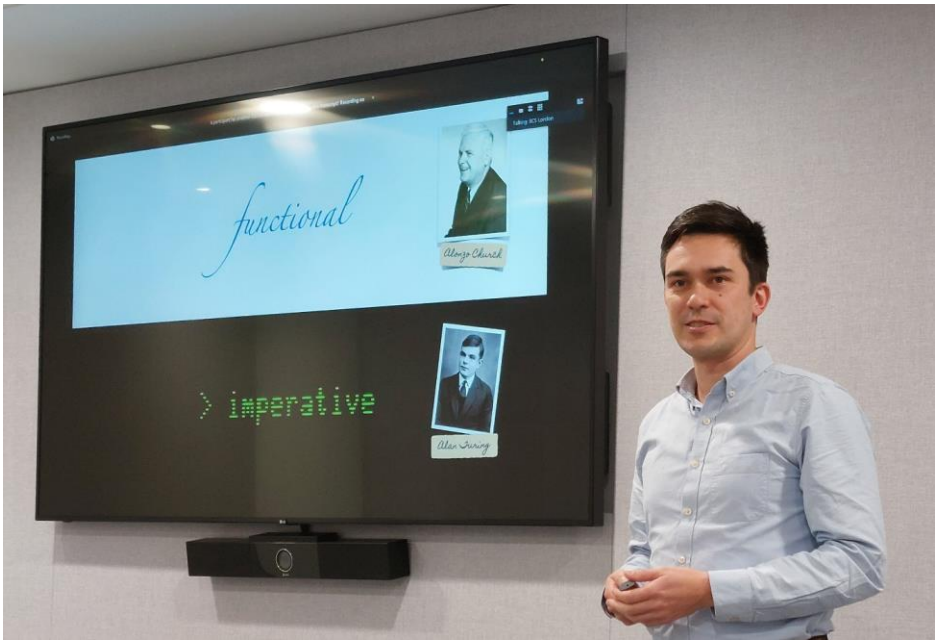
The Next 700 Domain-Specific Languages

Dr. Nicolas Wu
Imperial College, London[1]

Reported by: Tim Denvir

Speaker's Abstract

In this talk, we describe how embedded domain-specific languages with flexible and varied semantics can be composed out of smaller constituent parts using algebraic effects and handlers.



Nicolas Wu presenting the 2022 Peter Landin Semantics Seminar.
Photograph by Jonathan Bowen.

Report

The title gives homage to the Peter Landin paper, *The Next 700 Programming Languages*[2]. This set the scene for the speaker to refer substantially to the work of Peter Landin and its historical computational context: a pleasing emphasis, one not followed

1 The speaker's talk can be found here: https://www.bcs.org/events-calendar/2022/december/annual-peter-landin-semantics-seminar-and-facs-agm/
2 Com. ACM, March 1966 pp 157-166.

by all the Peter Landin Seminar speakers. Peter Landin, asserted the speaker, was at heart a programmer. Observe the distinction between *Imperative*, (Alan Turing) and *Functional* (Alonzo Church) programming.

In 1666 during the year of the great fire of London, Leibniz wrote his *Art of Combinatorics*: with language and logic we can reason about everything; but to quote Emil du Bois-Reymond, "we do not know everything and we shall not know"[3]. (Leibniz successfully constructed a mechanical calculating machine and dreamed of building a machine that could manipulate symbols in order to determine the truth values of both mathematical and linguistic statements[4].)

Contrast David Hilbert: "We must, we shall know": up till 1930 he believed that there would be no such thing as an unsolvable problem. Hilbert and Ackermann in 1928 posed the *Entscheidungsproblem* (decision problem): Is there an algorithm that takes a statement as input and determines whether it is universally valid (i.e. valid in every system satisfying the axioms)?

Gödel proved in 1931 that there were statements that were true but unprovable (Gödel's *Incompleteness Theorem*). Gödel's method of assigning numbers to logical formulas to reduce logic to arithmetic was an influence on both Alonzo Church and Alan Turing.

Both Church and Turing independently in 1936 proved that the answer to the *Entscheidungsproblem* was "No". The speaker showed samples of λ-calculus – a purely symbolic system of substitution rules – including an encoding of successor and addition (Church, 1936). There is a relation between Church's λ-calculus and Turing's computable numbers, specifically between λ-calculus and computable functions. There is an encoding between the halting problem (Turing) and the *Entscheidungsproblem*.

In 1945 John von Neumann wrote a report proposing the design of the EDVAC computer which was built in the US in 1949. This effort was contemporaneous with EDSAC and the Manchester "Baby" in the UK; protagonists of all three argue to this day which was first. Von Neumann devised a language scheme which reflected the way that computers worked but which was computationally equivalent to a Turing machine. In 1954 John Backus devised the Fortran Language, which embodied von Neumann's work.

In 1958 John McCarthy proposed LISP, a list processing language based on λ-calculus. Its features included conditional, recursive functions, list processing and higher order functions.

Again in 1958, Peter Naur, John Backus and John McCarthy designed Algol 58, which after discussion and revision became Algol 60. The syntax of the language was defined in BNF, Backus Normal Form, which later became known as Backus Naur Form. A number

---

3 Emil du Bois-Reymond, 1818-1896, German physician and physiologist, at a Congress of German Scientists and Physicians, 1872. See also https://en.wikipedia.org/wiki/Emil_du_Bois-Reymond

4 Martin Davis, 2000, *Engines of Logic*, W.W. Norton & Company, London, ISBN 0-393-32229-7 paperback.

of Algol's features, especially perhaps *call by name*, show an inheritance from λ-calculus. In 1974 Tony Hoare famously described Algol 60 as "a language so far ahead of its time, that it was not only an improvement on its predecessors, but also on nearly all of its successors". Hoare's program *Quicksort*, written in Algol 60, used recursion to achieve its conciseness and efficiency.

Peter Landin wrote some canonical papers, *The Mechanical Evaluation of Expressions*, *Correspondence between λ-calculus and Algol 60*, *The Next 700 Programming Languages* (1966), the latter 300 years after Leibniz! Also in 1966 he devised "ISWIM", ("If you See What I Mean"), a mathematical notation for functions using "let" and "where" clauses, sometimes dubbed as "Church without λ". The notation was, Landin claimed, more convenient than using λ terms, was indentation sensitive, incorporated garbage collection, had no explicit sequencing and employed algebraic data types. He distinguished between the physical and logical representations of a language, something emphasised in the Algol 60 Report. Principles of language abstraction were described.

Dana Scott and Christopher Strachey in their paper *Towards a Mathematical Semantics for Computer Languages*, initiated the foundations of denotational semantics (1971). In the distinction between functional and imperative programming, Landin preferred the term "denotative".

After this historical perspective, we moved on to Nicolas Wu's own work: **The Next 700 Domain Specific Languages**. Here are some quotes from his slides.

Data Structure and Recursion Schemes: Lists can be evaluated by folding their structure: cons and nil are replaced with a semantics of what to do. This technique generalises to many tree-like data-types. There are many variations on this theme, depending on mutual recursion, recursion with parameters, recursion with historic context, and many more.[5]

Domain-Specific Languages: syntax tree + recursion scheme = language + denotational semantics.

Algebraic Effect Handlers: Syntax represented by the free monad for a functor that provides a signature. Semantics often given in terms of a *fold* over that free monad.[6]

- **Syntactic**: Programs are embedded syntax trees that can be inspected.
- **Denotative**: The semantics is given by a handler that replaces operations with their meaning.

5 R. Hinze, N. Wu, J. Gibbons, *Conjugate Hylomorphisms*, 2015; R. Hinze, N. Wu, J. Gibbons, *Unifying Structured Recursion Schemes*, 2013; Z. Yang, N. Wu, *Fantastic Morphisms and Where to Find Them*, 2022.
6 G. Plotkin, M. Pretnar, *Handlers of Algebraic Effects*, 2009.

- **Algebraic**: Operations must respect *substitution* and *sequencing*.
- **Extensible**: Syntax trees can be extended with new syntactic nodes.
- **Modular**: Handlers can be composed to give combinations of semantics.
- **Flexible**: Multiple semantics can be given to a particular program.
- **Equational**: The relationship between operations can be expressed by laws.
- **Pervasive**: Algebraic effects cover a very large class of useful effects.
- **Efficient**: Intermediate trees can be avoided, to immediately return results.

Handlers and Heuristics: In a tree of nondeterministic computations, there are many different evaluation strategies. For instance, breadth-first, depth-first, depth-bounded, single result etc. This paper shows how these different strategies are handlers of nondeterminism[7]. This was used to model Prolog semantics as a DSL within Haskell, with a handler giving the semantics of each one.

Scoped Effects: Many handlers have related behaviour. These ['search' and 'once' operations illustrated in the talk] are not algebraic operations. Can the idea of algebraic operations be extended?[8].

Theory Meets Practice (GitHub[9]): a dialogue:

> **GitHub**: "Algebraic effects are nice, but we can't express lots of constructs like if statements and try/catch as syntax."
>
> **NW**: "Hmm, sounds like you need scoped effects…"
>
> **GitHub**: "Wow! That works, but is it efficient?"
>
> **NW**: "Yes! It all fuses!"
>
> **GitHub**: "Amazing! 250x faster than our previous attempts! We've rolled this out to production!"

Finally, Nicolas Wu returned to Church, Turing, and Peter Landin: ISWIM brings into sharp relief some of the distinctions that the author thinks are intended by such adjectives as procedural, non-procedural, algorithmic, heuristic, imperative, functional. This all suggests that algebraic effects may offer an exciting new paradigm in which language

---

7 T. Schrijvers, N. Wu, B. Desouter, B. Demoen, *Heuristics Entwined with Handlers Combined,* 2014.

8 N. Wu, T. Schrijvers, R. Hinze, *Effect Handlers in Scope*, 2015; M. Pirog, T. Schrijvers, N. Wu, M. Jaskelioff, *Syntax and Semantics for Operations with Scopes*, 2018; Zhixuan Yang, Marco Paviotti, Nicolas Wu, Birthe van den Berg, Tom Schrijvers, *Structured Handling of Scoped Effects*, 2022.

9 P. Thomson, R. Rix, N. Wu, T. Schrijvers, *Fusing Industry and Academia at GitHub*, 2022; N. Wu, T. Schrijvers, *Fusion for Free*, 2015

features can be given a precise, modular semantic characterisation emphasising clarity and correctness. Dr. Wu ended with the question: “Are Algebraic Effect Handlers denotative? I think they are." There followed some twenty-five minutes of questions and discussion.

Letter to the Editors

School of Computing
Newcastle University
1 Science Square,
Newcastle Helix,
Newcastle upon Tyne,
NE4 5TG
UK
Tel: +44 191 208 8183

2022-11-14

e-mail: cliff.jones@ncl.ac.uk

FACS FACTs

Dear Editors,

I enjoyed John Tucker’s paper on PL/I (and the Vienna Operational Semantics descriptions thereof) in the July FACS FACTS and should like to add a few notes.

First a tiny correction: it was ordained from on high that the name of the language should always use a Roman numeral. (There was a rumour that IBM also copyrighted the names PL/II and PL/III just in case – but I have no documentary evidence for this.)

John’s article lists (even has photos of) the reports comprising ULD-III Version II. Anyone wanting to access these documents (and Versions I and III) can find them on-line at:
http://homepages.cs.ncl.ac.uk/cliff.jones/semantics-library/
(Look about halfway down the page; remember that you are looking for VDL).

John alludes to the complexity of PL/I. Two specific features that are well beyond what John McCarthy tackled in his early *Abstract Interpreters* are:

- Concurrency: PL/I had a tasking feature that required a treatment of non-determinacy
- Under constrained storage mapping: compilers were allowed to choose how composite structures should be mapped onto byte storage – this gave rise to a form of axiomatic definition.

In my first stay at the Vienna Lab (1968-70) I worked with Peter Lucas on using the operational semantics as a basis for compiler design. Although in some ways successful, the excesses of the VDL operational semantics complicated the effort and were a significant push towards the later (I went back 1973-76) Vienna switch to denotational semantics.

John also mentions the VDM description of PL/I (see the same web page for TR25.139); I think it worth emphasising that this is a denotational description.
As well as Troy Astarte's PhD thesis, interested readers might like to look at: "Challenges for semantic description: comparing responses from the main approaches" which can be accessed either in LNCS 11174 or as a Newcastle Tech Report:

http://homepages.cs.ncl.ac.uk/cliff.jones/publications/NU-TRs/CS-TR-1516.pdf

One other note on PL/I and IBM is that the article *Early language and compiler developments at IBM Europe: A personal retrospection* by Albert Andres in Annals of History of Computing fills in the wider IBM Europe story. (I have been involved in discussions about getting an expansion of the IBM UK (Hursley) story put on the web but the Hursley Museum is more focussed on hardware.)

Yours sincerely

Cliff

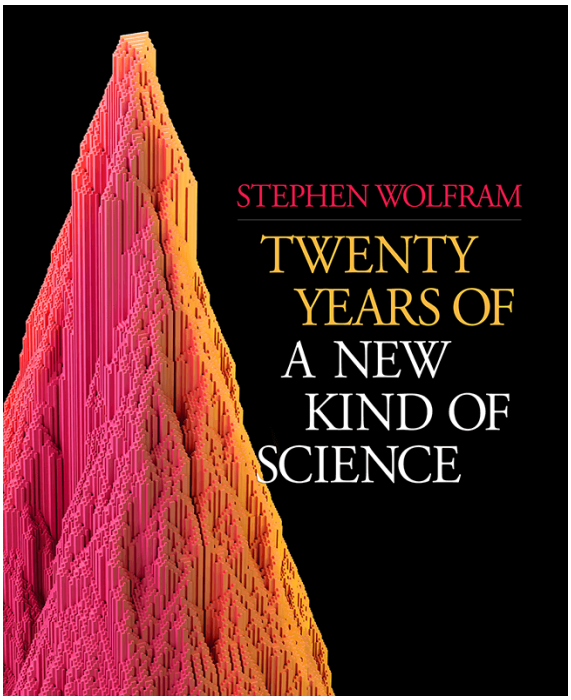
Cliff Jones

Twenty Years of A New Kind of Science

by Stephen Wolfram

Book review

Jonathan P. Bowen

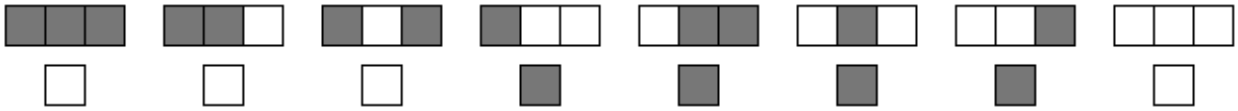


A previous article in *FACS FACTS* (Bowen, 2022) reviewed the book *Combinators: A Centennial View* by Stephen Wolfram (2021), surveying a century of combinatory logic since it was first formulated by the Russian logician Moses Schönfinkel (1888–1942) (Wolfram, 2020; Bowen 2021). The book under review here takes a rather shorter historical perspective, surveying the developments during the twenty years since Wolfram's magnum opus *A New Kind of Science* (1,280 pages) was published (Wolfram, 2002; 2022), as well as the making of the original book.

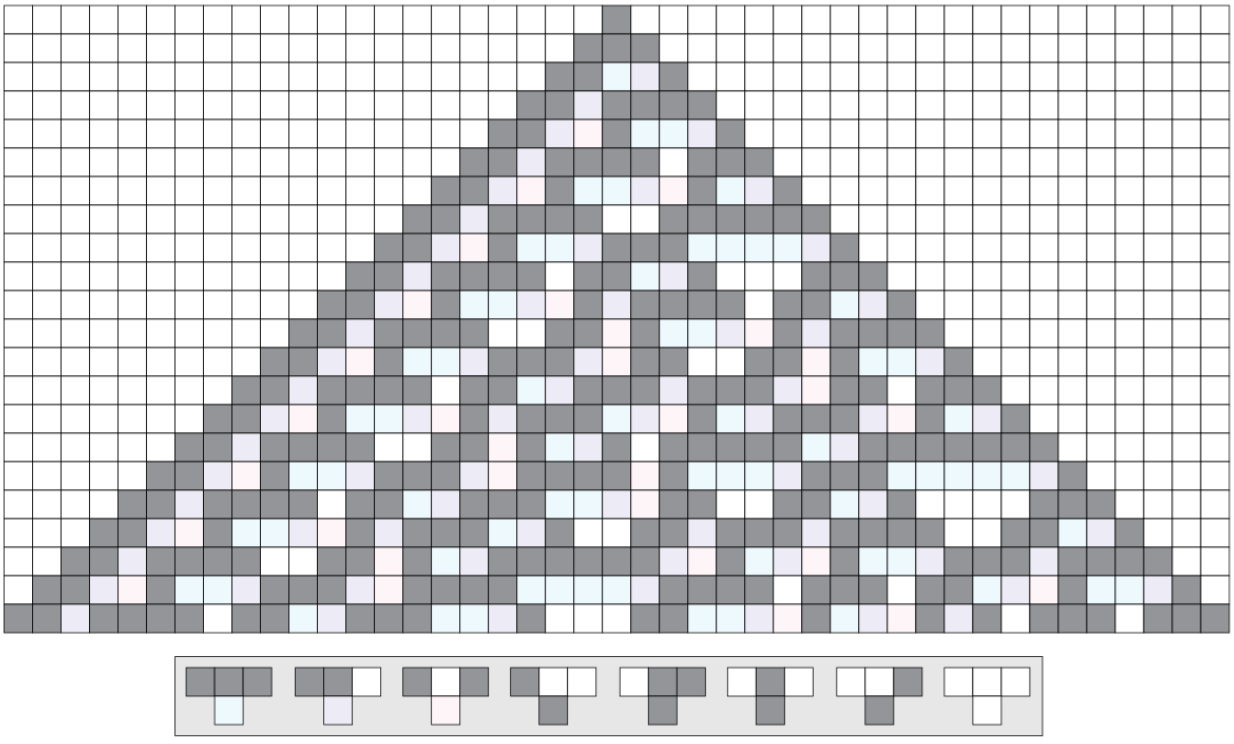
Wolfram is in the fortunate position of having enough support through his own successful company and its products, in particular the *Wolfram Mathematica* mathematical software tool, to be able to follow research interests in any direction that he chooses. He is a mathematician but with support for his writings, and with an interest in historical developments. The 2002 book *A New Kind of Science* was the culmination of a quarter of a century of research, building on ideas originating from cellular automata. This current book provides a personal tale of computational history over the last two decades and the creation of the original book before that. It includes a collection of existing work in a single volume, divided into four parts, previously largely available online in Wolfram's writings (see https://writings.stephenwolfram.com). The parts and chapters are not numbered, but the original dates of publication of chapters are included.

In the initial part, 10, 15, and 20-year views are presented in reverse chronological order. The first chapter gives a 20-year current view from the author. The next chapter gives a 15-year view, and the following three chapters were written in 2012, providing 10-year perspectives. In the second part, a very long chapter (157 pages) discusses the making of the original book from a contemporary perspective. A second chapter discusses recent (2021) thoughts on complexity, in the light of an earlier book, *A Science of Complexity* (Wolfram, 1995).

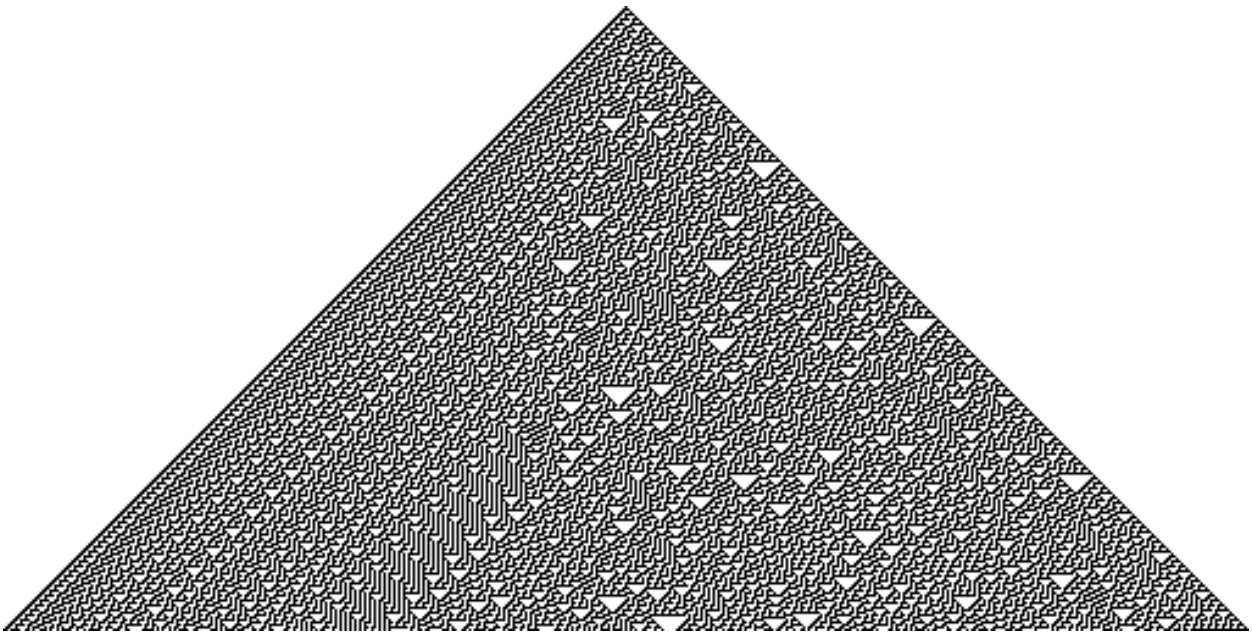
The third part of the book presents some prizes, one for producing a minimal Turing Machine, including the 2007 announcement (2 pages) with a 2007 notice about a proof concerning the simplest possible Universal Turing Machine (UTM), together with a further 2019 prize announcement of "Rule 30 Prizes". Rule 30 is a one-dimensional elementary cellular automaton rule that is relatively simple but which produces great complexity and apparent randomness as it proceeds through its steps. The prizes concern three problems associated with this rule. Wolfram's 2007 Turing Machine prize was awarded after four months. Wolfram predicts that the Rule 30 problems will take significantly longer to solve. *FACS FACTS* readers are welcome to try! See https://rule30prize.org for further details. The Rule 30 specification is of the following graphical form (freely available online under https://commons.wikimedia.org/wiki/Category:Rule_30):



The results for a given cell in the next step depends on the previous values of the three cells to the immediate left, right and that cell itself. The initial sequence of steps, starting with a single "black" cell at the top of the central column of steps, with additional colour shades for differently generated "white" cells, looks like this:



For the first 256 steps, the pattern of steps looks like this:



The three problems posed by Wolfram for the prizes concern whether the central column remains non-periodic, whether the black/white colour of cells in the central column occur equally often on average, and finally if computing the n^{th} cell of the central column requires at least $O(n)$ of computational effort. The central column looks random (at least for the first billion steps!), but is this really the case forever? After a billion steps, the ratio of black/white cells in the central column is 1.00010 and is mostly monotonically decreasing towards 1 for powers of ten steps (but not for 1,000 steps where the ratio is less than 1 at 0.92678). A naive approach to calculating the n^{th} cell of the central column requires $O(n^2)$ of effort, but is there a better algorithm, perhaps even better than $O(n)$?

The book ends with a gallery of digital and other art, largely based on cellular automata and some available as NFTs (Non-Fungible Tokens). The full gallery can be found online (https://www.wolframscience.com/gallery-of-art). Rule 30 has bee used in some physical works of art and designs. For example, the cladding of Cambridge North railway station uses this pattern (see https://commons.wikimedia.org/wiki/Category:Cambridge_North_railway_station).

General view of Cambridge North railway station.

Detailed view of the cladding at Cambridge North railway station, showing Rule 30 patterns.

Overall, this book is relevant to those interested in the production of the original 2002 book *A New Kind of Science* in the second part and relevant developments since then in the first part. The prizes associated with "Rule 30" in the third part may be of interest to those keen on mathematical problems and with plenty of time to spare. No solutions are offered in the book under review! The last part is likely to be appealing to those with more interdisciplinary and especially artistic interests, including this reviewer, who is a member of the BCS Computer Arts Society (CAS) as well as the FACS Specialist Group. In summary, the book forms a nice collection of articles associated with the original book *A New Kind of Science*. Potential readers will find most of the material already freely available as articles online, but the book provides them in a convenient collected form.

References

Bowen, J. P. (2021). *Moses Schönfinkel and combinatory logic*. *FACS FACTS*, 2021-1:23–25, February. https://www.bcs.org/media/6694/facs-feb21.pdf

Bowen, J. P. (2022). *Book review: Combinators: A Centennial View*. *FACS FACTS*, 2022-1:56–59, January. https://www.bcs.org/media/8289/facs-jan22.pdf

Wolfram, S. (1995). *The Science of Complexity*. Addison Wesley Longman. ISBN: 978-0201622737.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media. ISBN: 978-1579550080. URL: https://www.wolframscience.com/nks/

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

Wolfram, S. (2021). *Combinators: A Centennial View*. Wolfram Media. ISBN: 978-1-57955-043-1 (hardback), 978-1-57955-044-8 (eBook) URL: https://www.wolfram-media.com/products/combinators-a-centennial-view/

Wolfram, S. (2022). *Twenty Years of A New Kind of Science*. Wolfram Media. ISBN: 978-1-57955-049-3 (hardback), 978-1-57955-051-6 (eBook) URL: https://www.wolfram-media.com/products/twenty-years-of-a-new-kind-of-science/

Forthcoming Events

We have a new Seminar Organiser on the FACS committee, Alvaro Miyazawa at the University of York. If you have suggestions for future FACS seminar speakers or other events, especially if you are willing to help with co-organisation or even give a talk, please contact Alvaro on Alvaro.Miyazawa@york.ac.uk.

Events Venue (unless otherwise specified):

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

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

Details of all forthcoming events can be found online here:

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

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

FACS Committee

Formal Aspects of Computing Science Specialist Group

Jonathan Bowen
FACS Chair and BCS Liaison

John Cooke
FACS Treasurer and Publications

Roger Carsley
Minutes Secretary

Ana Cavalcanti
FME Liaison

Tim Denvir
Co-Editor, FACS FACTS

Brijesh Dongol
Refinement Workshop Liaison

Keith Lines
Government and Standards Liaison

Alvaro Miyazawa
FACS Seminar Organiser

Brian Monahan
Co-Editor, FACS FACTS

Andrei Popescu
LMS Liaison

Margaret West
Inclusion Officer and BCS Women Liaison

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.