10 Teaching at Belfast and Oxford†

Bernard Sufrin

> Nine years is a long delay for an inaugural lecture; but it has taken all those nine years to introduce an undergraduate curriculum in Computing at Oxford.
>
> C.A.R. Hoare
>
> *The Mathematics of Programming.* An Inaugural Lecture delivered before Oxford University, 17th October 1985.

10.1 Introduction

Although as a youth he had been known as 'the Professor' by his younger siblings, Tony Hoare's first academic appointment was in 1968, as Professor and Head of the Department of Computer Science at Queen's University Belfast. His second was in 1977 as Professor of Computation and Head of the Programming Research Group of the Computing Laboratory at the University of Oxford — until 2011 the closest thing that the University had to a Department of Computer Science.

Queen's and Oxford were fortunate to appoint a pioneer with Tony's tastes, experience, and temperament to departments whose tradition of teaching had hitherto been mainly in numerical computing, for he revolutionised each of them intellectually during his tenure.

He has occasionally been a bit too modest when he speaks about his own role in these transformations: 'I just found some really good people, then I let them get on with it.' In fact his extraordinary inventiveness and his talent for exposition meant that good people sympathetic to his approach flocked towards him; and he has always had a remarkable capacity for evoking in his collaborators the sense that they are part of a shared scientific and educational mission.

† This is an expanded version of the identically-named chapter in the ACM-published volume: *Theories of programming: the life and works of Tony Hoare*, edited by Cliff B. Jones and Jayadev Misra.

Figure 10.1 Until 1982 the Programming Research Group was located in a Victorian semi-detached house

He made it clear what was expected of his 'good people', and always took an active interest in their teaching and in the progress of their research. He published jointly with many of them, and encouraged some to develop academic textbooks of their own. In these endeavours his inability to overlook a clumsy piece of prose or an insufficiently tractable theory was complemented by his skill at making pointed editorial annotations; although the people on the receiving end always knew that he thought of them as peers in the search for clarity, not subordinates over whom he was pulling rank. This view is confirmed by Jones and Roscoe in [26].

Here we give an account of his role in the introduction of masters and undergraduate programmes in computing at Queen's and at Oxford. The challenges he faced were different in the two universities. At Queen's he was fresh to academic leadership, and there was not yet a well-developed approach to treating computing as a science. But he was starting with an adequately-staffed department in a university committed to teaching computer science degrees. And the Queen's organizational structure was simple, so it was straightforward for him to identify and convince the people whom he had to enlist in support of his initiatives.

On the other hand, by the time he reached Oxford many of the ideas that would naturally be included in the core curriculum had already evolved — several under his own influence. But taught degrees in computing were unheard of, he inherited premises in a cramped and

ill-equipped Victorian semi-detached house,[1] and he had just a single academic colleague in his subject. And in contrast to Queen's, Oxford was organizationally complex with some decision-making loci inhabited by people whose willingness to cooperate with initiatives was unpredictable at best. As Tony puts it:

> When I got to Oxford, everything was turned on its head. In Belfast, one could make an argument based, for example, on the public perception. [...] Or you could base it on the potential benefits of the successful application or exploitation of the research. These arguments carried no weight at all in the Faculty of Mathematics at that time. Starting up a new course was something that the University was able to contemplate sort of — I exaggerate slightly — once every decade. [22]

Acquiring the staff, the space, and the resources to transform this nucleus into a broad and successful research department that could also deliver undergraduate and masters degrees would require a good deal of his patience and diplomatic skill, as well as an ability to navigate external constraints and exploit external opportunities as they arose.

10.2 Queen's University Belfast

10.2.1 Background

The history of automatic numerical computation at Queen's had begun in the early 1860s when James Thomson invented the mechanical analogue integrator — the basic mechanism of the Differential Analyser built for computing tide tables by his younger brother, Lord Kelvin. The tradition of computational invention was continued in the 1930s by the distinguished mathematical physicist Harrie Massey, whose design for a differential analyser is reported to have been built at a cost of only £50.

A sub-Department of Digital Computing appeared in the Department of Applied Mathematics and Theoretical Physics in 1959, and the Computing Laboratory was established in 1960. In 1965 the first Professor of Computer Science, Jim Browne, a theoretical physicist from Texas, was appointed and the Department of Computer Science was established a couple of years later. The emphasis in the new department was naturally on numerical analysis, and programming was in FORTRAN. Browne himself began the switch of emphasis to computing as a science in its own right, before returning to the U.S.

[1] Photocopying was done onto chemically-developed single photosensitive sheets. There was no direct access to the University Computing service, and the only directly-accessible computing service in the house ran on a shared, superannuated 32K Modular One [40].

10.2.2 Appointment

Tony's transition from industry to academia had started when he left the UK company into which Elliott Brothers, his original firm, had been absorbed to take up a Senior SRC[2] Research Fellowship at the UK National Computing Centre in Manchester. He applied for the Belfast chair soon after he arrived there. To paraphrase him:

> It occurred to me, perhaps a little late, that maybe the best way of finding out about the academic computing scene was to go for a few interviews for posts. So I rather tentatively drafted a letter of application and wondered if I would finish it in time to catch the post. I said to myself that if I could catch the post then I'd submit the application. Well I did catch the post! I went for an interview, and to my intense surprise I was chosen for it. [22]

But

> when I returned to my office in Manchester my boss at the NCC was furious with me. I had only worked for six weeks [there]. He was the external assessor for the post at Belfast, and he assured me that I would not get the post because there was a far better candidate. [...] I served out the remaining three months of my probationary period [at the NCC having given notice].

In retrospect it is easy to see that Queen's showed great foresight, but the appointment was then unconventional in that Tony's background was entirely in industry, he had no doctorate, and he had never before held an academic post. The fact that he had earlier been invited to apply for chairs in Oslo and in Manchester puts his 'rather tentatively' into perspective, and if Queen's had any qualms at all, they must have been allayed by the knowledge of his achievements while in industry. These included inventing his Quicksort algorithm then analysing its performance in detail [14, 15], leading the team that produced the first commercial ALGOL compiler, and jointly publishing a proposal, implemented later, for the design of a successor to ALGOL 60[47]. His membership of IFIP WG2.1 provided ample additional evidence of peer esteem.

He describes what it felt like to be parachuted in to his first academic post this way

> It's quite an experience coming in at the top, as it were.
> [...]
> I was a bit shocked when one of the first things I had to do when I arrived in October

[2] The UK Science Research Council; now the Engineering and Physical Sciences Research Council. This research funding body has had a variety of names since its inception as the Department of Scientific and Industrial Research (DSIR). In this chapter we shall continue to use the initials SRC.

> was to decide something about the syllabuses for the following year's courses. We never thought that far ahead in industry.
> [...]
> The other thing was getting used to academic politics, which is quite different from industrial politics. I realised that all professors were equal under the vice-chancellor, but that you had to understand which of the professors were more equal than the others. [22]

As was not uncommon at the time, Queen's ran a computing service from the Computing Laboratory, and its direction was delegated *ex-officio* to Tony. He took this responsibility seriously, but being faced with a genuine computing scientist may have been more than a couple of his colleagues on the management committee for the service could stomach.

> Well it was pretty unpleasant for the first two years actually. [...] The manager of the Computing Laboratory and the Professor of Medical Statistics, who was chairman of the computing services committee, attempted to dislodge me. [...] In the end I went to the vice-chancellor and said 'Am I the Director or am I not the Director?' He said 'You are the Director.' Anyway I explained the problem to him and he said he'd look into it, and he came back with the right decision: I was *not* the Director. It was a great relief. [22]

Tony was never one for getting involved in fruitless altercations, and it is now evident that he put the energy liberated by this decision to much better use.

10.2.3 Taught Degrees

By 1970 an M.Sc. and a graduate Diploma in Computer Science and Applications had been established — at first with substantial numerical analysis content. The Department also began to establish Computer Science as an undergraduate subject in its own right, and Tony's inaugural lecture, delivered three years after he took up his appointment, made it clear that he saw programming as its core discipline.

> Having surveyed the relationships of computer science with other disciplines, it remains to answer the basic questions: What is the central core of the subject? What is it that distinguishes it from the separate subjects with which it is related? What is the linking thread which gathers these disparate branches into a single discipline? My answer to these questions is simple — it is the art of programming a computer. It is the art of designing efficient and elegant methods of getting a computer to solve problems, theoretical or practical, small or large, simple or complex. It is the art of translating this design into an effective and accurate computer program.[16]

His colleague Jim Welsh has written of Tony's time at Belfast:

> Tony's first day at Queen's, 1st October 1968, was also my first day as an Assistant Lecturer. The undergraduate course offerings were only just getting under way at that point, so he inevitably had a major influence on the content, if not on the initial framework. Initially we used an interpretive language, QUBAL, (a thinking man's BASIC which I had developed as part of my PhD) for the introductory programming courses, but under Tony's influence the goal was to get to a proper structured programming language as soon as possible. So as soon as Wirth's team had the first self-compiling Pascal compiler running on the CDC machine in Zurich, Tony set me the task of bootstrapping it onto the ICL 1900 series. That was the most challenging but most rewarding project I've had in my entire career, and its success laid the foundation for the strong Pascal orientation of all teaching and research at Queen's for the next ten years.
>
> A research grant on program proving funded Maurice Clint, Peter Lauer and myself, though I dropped out of that pretty quickly when the Pascal compiler opportunity arose. The underlying specification and verification theme quickly permeated much of the department's research interests, and to a lesser extent the undergraduate curriculum, much earlier than occurred in most universities.
>
> At the undergraduate level, the Dahl/Dijkstra/Hoare/Wirth school of structured programming was incorporated early and with enthusiasm, putting us well ahead of the pack in that regard. [...] The incorporation of formal methods was more nuanced, but Tony's presence and enthusiasm ensured that every staff member took their relevance to heart.

His colleague Mike McKeag has described how some fundamental topics were approached. Tony's influence and his approach to teaching is evident here.

Practical work: A significant practical element (20% of the final assessment) featured in every course.

Computer Programming: having worked closely with Niklaus Wirth, Tony initiated the teaching of programming through the medium of Pascal, thereby encouraging a disciplined structured approach. From 1972 the seminal text on Structured Programming that he edited along with E. W. Dijkstra and O. J. Dahl [9] provided important background reading — and also helped students understand that a language used to convey computational ideas need not be the same as the language in which they are expected to program their practicals.

Compiler Construction: this covered the construction in Pascal of a complete compiler for a subset of Pascal and was based on the effective recursive descent ALGOL compiler developed

at Elliott Bothers. The practical work entailed extending the language and its compiler, and the goal was for students to learn how to work with a complete substantial program.

Parallel Programming: a departmental research project with ICL had studied the engineering of successful operating systems, and Tony himself had learned some hard lessons in this field while in industry[17]. Lessons learned from the Burroughs B5500 Multiprogramming System and from Dijkstra's THE Multiprogramming System influenced the syllabus for this course, in which a model operating system was developed in the Pascal-Plus language[45]. Its monitor construct was used to facilitate safe synchronization between parallel processes.

Theory: Tony's research interest in formal program specification and rigorous verification led to several courses on these topics appearing. Both axiomatic and denotational semantics were taught, and the connection, through Peter Lauer, with IBM Vienna eventually led to the teaching of a course on VDM.

Algorithms: given Tony's reputation as the designer of Quicksort and the developer of convincing proof techniques, as well as his interest in parallelism, it is not surprising that the results of research in sequential and parallel algorithm design also found their way into the curriculum.

Indeed, as a matter of policy, research results in all areas quickly found their way into most parts of undergraduate curriculum.

Tony's undergraduate tutorials at Oxford, in which a student prepared weekly or biweekly essays on topics of their tutor's choosing, may well have inspired the style of the *General Paper* taken in the final year. This comprised the supervised study of a topic selected by a tutor, the preparation of an essay, and its oral delivery to the Head of Department. The written part of the paper consisted of open-ended questions, challenging students to develop their writing skills and to marshal coherent and convincing arguments in support of their views on topics not covered explicitly in lecture courses. Joint degrees were developed with several departments in the Science Faculty; and, unusually for that period, Computing was also offered in the Arts Faculty for students who wished to combine it with study of the humanities.[3]

Relationships with industrial practitioners were promoted by an external education programme in which the department's course material was presented to professionals, usually as five-day residential courses.The success of this programme prefigured a more extensive programme of continuing education for software engineers that was eventually to develop in Oxford.[4]

[3] The presence of someone with a Classics degree must have been reassuring to Tony's humanities colleagues when he first explored the prospect of this with them.

[4] Computing undergraduates were also required to work in industry for a full year before their final year, and this 'sandwich scheme' eventually expanded to include employers in Europe, America, and parts of Asia. On their

10.2.4 Northern Ireland's 'Troubles'

Northern Ireland's very violent 30-year 'Troubles' began very shortly after Tony arrived at Queen's. To paraphrase his view of the effect on his family:

> Yes, of course it had quite a strong effect. To begin with it seemed rather distant and was over the other side of the province in Londonderry. But it moved to Belfast and it moved to the areas that you would expect in Belfast: the Falls Road and Shankill Road. And it did go on getting worse year by year until about 1972, and so we were always wondering whether we'd made the right choice and when we would be running for our lives.
>
> But it was such a friendly place, such a lovely place to be, and the job and my colleagues [and neighbours] were so wonderful that we really enjoyed it. The only time that Jill was really worried was when I was told that I had been appointed [without applying] to another post [in London] and [asked] would I come and talk to the vice-chancellor about it? I probably would not have gone unless I'd been invited to be the Professor. So I went for an interview and I turned them down. And Jill says that was the only time that she was really worried when I was in Belfast: that she might have to come back to London.[22]

Yet because of his burgeoning reputation, and despite the increasingly dangerous reputation of the province, he was able to attract large numbers of leading computer scientists to the department as visitors.[5]

Although he has described a few close shaves during the troubles, what sounds like one of the closest happened during an academic symposium that had attracted wide international participation. Sessions were held in Belfast's City Hall, and as one of the sessions was drawing to a close a very loud bang was heard nearby. Fortunately nobody but the locals had recognised the sound of the bomb explosion for what it was, and he was able to shepherd everyone into the bus that had been scheduled to take them on a sightseeing tour of the city, and to persuade the bus driver to take them on a less troubled and more scenic route around the province. He says that if any of the attendees had realized what had actually happened it would have been impossible to persuade any more academic visitors to come in future. Ironically there was

---

return they would give a presentation to the whole department that had to include remarks on the relevance of their undergraduate modules in their employment. Their real-world experience of the employment of informatics yielded a noticeable increase in their maturity. Their suggestions for extending or deepening the curriculum were taken very seriously.

[5] Among those who visited were Robin Milner, Peter Landin, E.W. Dijkstra, John Reynolds, Rod Burstall, Brian Randall, David Cooper, Per Brinch Hansen, Michael Jackson, Jean Ichbiah, and Michael Melliar Smith. Some, for example Niklaus Wirth, Peter Lauer, Willem de Roever, and Nissim Francez, stayed for longer periods to collaborate on research and teach short courses.

a second loud bang as the bus approached the coast, but this was a sonic boom caused by Concorde, which at the time was being flight-tested over the Irish Sea.

10.2.5 Adieu

Tony's time in Belfast coincided with a sustained period of broadening and deepening of computing as an academic discipline, and of the department he led as one of the most energetic and innovative departments in the world. It also confirmed him as a leading figure of the movement to establish programming and programming language design as theoretically coherent engineering disciplines. It would be hard to exaggerate the sense of liberation from fruitless reductive arguments felt by programmers, and teachers of programming, as they came to understand the ideas behind data abstraction, prealgorithmic specification, and what is now called *refinement* that he had played such an important part in clarifying and promulgating.

10.3 University of Oxford

10.3.1 Background

It was only with the founding of the Computing Laboratory in 1957 that Oxford finally committed itself to providing centralized support for numerical computing in the sciences.[6] This followed a campaign by distinguished mathematicians and scientists that began in the late 1940s.[7] At first Oxford hedged its bets over the extent to which electronic computers would be needed,[8] but by 1960, under the direction of the distinguished numerical analyst Leslie Fox, the laboratory was running courses and summer schools on scientific computing, and a modern digital computing service for the whole university. It had also become a leading centre for research in numerical analysis and scientific computation. By 1963 numerical computing had received recognition by the university as an academic discipline in its own right, with the appointment of Leslie Fox as Professor of Numerical Analysis.[9] It eventually took its place in the undergraduate Mathematics curriculum after the appointment of a handful of academics able to teach it in tutorials in the colleges — no easy matter in the Oxford of the mid-1960s, or since.[10]

A radical broadening of the research scope of the laboratory in the direction of non-numerical computing began in 1965, with the foundation within it of the Programming Research Group (PRG), funded with a fixed-term grant from the immediate precursor of the SRC and led by Christopher Strachey, long celebrated as a pioneering programmer and computer designer.[11]

---

[6] By way of comparison, Cambridge's Computer Laboratory was founded, as the Mathematical Laboratory, in 1937. Its staff had designed and built one of the earliest digital computers — EDSAC — and had been running a computing service on it since 1949 [46].

[7] Campaigners eventually included Charles Coulson FRS, and Dorothy Hodgkin, who was later to win the Nobel Prize for Chemistry.

[8] Tabulators were also funded, as well as hand calculating machines. Posts were established for a professorially-salaried Director, '*a suitable number of non-graduate (girl) computers*' and two graduates — one a programmer, the other a numerical analyst. The 'girl computers' must have had their hands full for a while: the first modern electronic computer, a Ferranti Mercury, was not delivered until 1959. Until then there was just an obsolete HEC computer 'the Monster in the Basement', donated by the British Tabulating Machine company, capable of sterling arithmetic but without a multiplier.

[9] Fox was also elected a Fellow of Balliol College. At the time a college fellowship was *sine qua non* for somebody expecting to be able to take academic initiatives within the university.

[10] Oxford has a federal structure, being a collegiate university composed of self-governing constituent colleges — more than 35 at the time. All students must be members of a college, and undergraduate teaching is organised around tutorials at the undergraduate colleges, which are complemented by classes, lectures, seminars, and practical work provided by university departments. So a new academic subject cannot be part of the undergraduate curriculum at Oxford until colleges can offer tutorials in it. At the time it was unthinkable that tutorials be given by someone of lesser standing than a University Lecturer (=Associate Professor) and election to a college tutorial fellowship was made simultaneously with the University appointment. The main desideratum for the University appointment was nearly always research prowess, while that for the tutorial fellowship was appetite and aptitude for teaching. That these talents could not always be found in appropriate measure in a single individual occasionally gave rise to protracted negotiations over appointments between departments and colleges.

[11] Canvassing for a college fellowship for him at the time of his appointment, The Earl of Halsbury, a leading figure in the UK scientific establishment wrote: 'If you are to assess Strachey for a Fellowship you will have to start straight

He moved from Cambridge to Oxford and was appointed Reader in Computation.[12] In 1971, when the university took over the funding of the PRG from its own resources, he was given the title Professor of Computation.

His founding vision for the PRG had been clear, and Tony would eventually quote it in his own inaugural lecture.

> It has long been my personal view that the separation of practical and theoretical work is artificial and injurious. Much of the practical work done in computing, both in software and in hardware design, is unsound and clumsy because the people who do it have not any clear understanding of the fundamental design principles of their work. Most of the abstract mathematical and theoretical work is sterile because it has no point of contact with real computing. One of the central aims of the Programming Research Group as a teaching and research group has been to set up an atmosphere in which this separation cannot happen.

By the time of Strachey's early and unexpected death in mid-1975 the PRG had acquired a leading reputation for research in computing. His pioneering work on programming language semantics, and his later collaboration with Dana Scott,[13] had been enormously influential [30–32, 34, 35, 41, 42],[14] as had the group's practical research [33, 38–40].[15]

Although it had also taught the *Theory of Programming Languages and Computation* subject stream of a Diploma in Advanced Mathematics[16] to handfuls of students a few times, the PRG had remained essentially a research-only establishment since its foundation, and had only ever provided supervision for very small numbers of doctoral students.[17] So its continuing

---

from the admission that he is an unusual kind of person who does not fit into any kind of formal classification. To say that he has not got the formal qualifications for a fellowship is merely to decline to discuss the question. Strachey has not got a higher degree for the simple reason that he has been too busy developing the pioneer phase of a completely new technique, and his work has issued in practical forms which do not lend themselves to publication as scientific papers.'

[12] The rank of Reader in Oxford was generally held by scholars with a distinguished research record. It is equivalent to full professor in the US.

[13] Scott had been appointed Professor of Mathematical Logic in the Philosophy Faculty in 1972.

[14] Lengthier tutorial treatments of the Scott-Strachey approach to semantics were published soon after Strachey's death [28, 37, 44]

[15] For example, Peter Mosses had built the prototype of a system for executing programs based on their denotational semantics [32, 33], and Strachey and Stoy had published the annotated BCPL source code of OS/6 — an operating system that anticipated some aspects of UNIX [38–40] (BCPL was a precursor of C)

[16] Joe Stoy writes: 'Esoteric fact: it couldn't be a master's course, because in those days we still adhered to the rule of the mediæval University that the top degree in each Faculty was either the Doctor or the Master, and you couldn't have both in the same Faculty. So Master of Arts, Doctor of Medicine, Master of Surgery, Doctor of Science *etc*. That finally broke down, probably in the 70s, when the Americans said they were damned if they were coming to Oxford for a one or two year graduate course and going back with just a B.Phil or a B.Litt or whatever. So we could finally have an MSc.'

[17] But Stoy writes: 'Yes, it was a small number; though we were quite pleased with ourselves when one year we produced something like 2% of the country's doctoral output with just 0.02% of the country's computing staff'

existence was by no means assured after Strachey's death. In fact it took a lengthy and concerted campaign to persuade the University to commit funds to establish a *statutory* (*i.e.* permanent) chair in computation — the only way to prevent the extinction of the group.[18]

10.3.2 Appointment

Tony's election as Professor of Computation and Fellow of Wolfson College came as no surprise to the computing community: nearly everyone had been thinking of him as the obvious successor to Christopher Strachey.[19] But he has written of being perplexed at how long it took for Oxford to act on his inclination to move there once he made them aware of it.[20] To paraphrase him:

> I applied for an advertised Chair in Oxford in the standard way, and the application was acknowledged. I kept hearing rumours that I had been appointed. [But] I think there was a two-year delay before the vice-Chancellor[21] wrote to offer me the job and an opportunity, if I wished, to talk to him. I took the opportunity, and attempted to squeeze out of him an additional lecturer post, which had sometimes been accounted as a perk of an incoming science professor. No such luck! I wanted the job too much.

Joe Stoy had been despatched from Oxford to Belfast to see him at some point.[22] He writes:

---

[18] This was not easy during the financial retrenchment then taking place in the UK University system. The prime movers were Dana Scott and Leslie Fox: they first had to overcome some parochial opposition within the Board of the Mathematics Faculty and then to persuade the General Board of the Faculties — the body that had the final say.

[19] Coincidentally the man who as Director of the Computing Laboratory would formally become Tony's head of department was Leslie Fox, who had introduced Tony to programming — in Mercury Autocode — on his return to Oxford in 1958 to study Statistics for a year. Although Fox could occasionally present as somewhat gruff, he was completely in sympathy with the aims of the PRG, in whose founding he had played a decisive role after earlier being sceptical about the prospects for non-numerical computing [11, 43].

[20] Perhaps that is partly explained by the time it had taken to secure permanent funding for the chair. Oxford was, anyway, operating in an adverse economic and financial climate: the UK was in mid-recession and there was a high (double-digit) rate of inflation. It would not have been beyond the Oxford of that period to advertise the Chair before committing the funds — in order to see if there were any distinguished applicants to bolster the case for commitment. Access to the archive of the electoral board and the General Board of the Faculties might settle this question: but they are presently inaccessible due to pandemic disruption.

[21] The (elected) highest functionary of the University, equivalent to the President of a US University, The Chancellor plays a purely ceremonial role. In recent times the vice-Chancellors of some other UK Universities have insisted on their post being called 'vice-Chancellor and President'. Apparently this started when a very grand vice-Chancellor was told by an even grander potential donor that he wanted to talk to his 'boss.'

[22] Stoy likes to joke that he was simultaneously the most senior and the most junior member of the PRG at the time. But he was also a Research Fellow of Balliol College and had been the Chair of the Mathematics Faculty's sub-Faculty of Computation — established in 1969 in response to a governmental initiative proposing that *all* undergraduates should in future be taught to appreciate the potential of computers, if not to program. The sub-Faculty was the first place that matters relating to any form of teaching of computing anywhere in the university would be discussed. Joe was both well-connected and familiar with Oxford's complicated committee structures, so Tony would sometimes turn to him for help with piloting degree legislation through (or around) the arcana, and he was once heard saying that he would have to consult 'Regulation Joe' over a particularly recalcitrant problem. The sub-faculty had accumulated

> I don't remember whether it was Tony or someone in Oxford who suggested I go; I was to give him some insight into the PRG at the time, I suppose, and hoped to talk him into letting his name be considered by the electors to the Chair.

Tony had already been working for several years on the challenges posed by concurrent programming and was attracted to the idea of moving to Oxford and working on the semantics of concurrency.

> I had written a paper on CSP and published it in the Communications of the ACM, in the standard style of the time, as an informal description illustrated by a great many simple but obviously seminal examples. But in fact one of the reasons why I wanted to move to Oxford was to learn the technology of giving a formal definition to a programming language from Joe Stoy and Dana Scott in order to be able to redress the deficiency and make a formal model. [22]

He also wanted to start a taught master's programme directed at professionals working in industry — something that would not have been feasible at Queen's.[23] But as he arrived in Oxford in October 1977 the academic staff of the PRG consisted just of himself and Joe Stoy — then a senior research officer. They were supported by two programmers, and a PA; with a 'schoolboy programmer' in post during the Cambridge summer vacations.[24] It was evident that before such a programme could be delivered it would be necessary to find more people able to teach courses, better accommodation, and much better computing facilities for those who felt they needed them.[25][26]

10.3.3 Towards a Viable M.Sc. in Computation

It is not clear how Tony eventually managed to convince the powers that be that the PRG was capable, with an official teaching staff of only two, of sustaining a master's course that could

---

many contingent responsibilities, including overseeing the *management* of the University Computing Service which had grown very large. In addition to providing the service suggested by its name, it provided the venue and the instructors for a host of courses directed at academics and students who expected to use computers in their work. Among the earliest things Tony did on arrival was to successfully argue for the sub-Faculty to divest itself of this responsibility while still keeping an eye on the academic direction of the service's courses.

[23] The Irish software industry was of negligible size, and the ongoing 'Troubles' would have made it particularly hard to attract potential students from elsewhere.

[24] This was John Hughes, who was to join the group later as a D.Phil student; and still later for a brief stint as University Lecturer in Computation and Tutorial Fellow of St. Edmund Hall

[25] Some colleagues who arrived at the PRG after Tony had been used to using powerful interactive systems for document production and programming, and were surprised by how primitive the in-house facilities had remained. Tony took their expressed needs seriously, though he himself didn't need computing facilities at all. But after 1979 the computing facilities improved dramatically; and until the late 1980s some of their features were still considered quite advanced — a beneficial outcome of Tony's encouragement of research that fused practice with theory [12].

[26] Details of the successive enhancements to the group's accommodation in Tony's time would be out of place here. So would an extensive account of how the Computing Laboratory evolved to the point where it became the Department of Computer Science in 2011; but a narrative overview can be found on the Department's website [2].

start in 1979-1980, but there is evidence that it was an uphill struggle.[27] What is clear is that every experienced teacher who came to work or study at the PRG as a student, a visitor, or a researcher for the first few years after October 1978 was invited to teach a course to the group's (by now numerous) research students and on a master's course. And who could think of saying no to such an invitation from Tony? By this means he soon gathered enough teachers, enlisting Jean-Raymond Abrial,[28] Cliff Jones,[29] Jim Kaubisch,[30] and the present author[31] in his enterprise from Oxford, as well as the eminent software industry figures Michael Jackson, John Barnes, and John Buckle. Perhaps the anticipated presence of this voluntary cadre had gone some way to reassuring the powers that be.

He set about recruiting the first student cohort. They had to be self-funding or sponsored by their employers, since no SRC grants had yet been made available for the degree. The publicity material explained the background and why experienced programmers ought to attend:

> The discovery of the mathematical basis of computer programming now permits construction of programs that are proved to meet their specification, and promises to transform programming from an arcane and error-prone craft into a modern engineering profession.
> [...]
> Most programmers now active began before [this revolutionary development] and many now stand in need of professional reorientation. The need is being met in part [by commercial organizations selling short courses based on proprietary methods] But there is still a need for a smaller number of more senior programmers to obtain a broader and deeper understanding [from] a more extended course at a University.

as well as what could be expected of them afterwards by their employers:

> A graduate [...] on return to his employer, should be able to achieve a rapid transfer of the new technology; he would select techniques most appropriate to his environment, adapt them and improve them as necessary, establish appropriate design standards [...] inspire and train his colleagues and subordinates in the observance of sound practices, and [...] keep abreast of future research and development.

[27] An early sub-faculty minute reports that disquiet about staffing levels had earlier been expressed at the University's Graduate Studies Committee and by the Science Research Council. Another minute reports Tony's frustration that priority for a new University Lectureship had been withdrawn from the PRG and transferred to Logic.

[28] Who came to Oxford as a senior SRC-funded research fellow.

[29] Who came to Oxford to take his first degree, a D.Phil, under Tony's supervision.Having taken this degree he was appointed directly to a Chair at Manchester University.

[30] His former doctoral student.

[31] The latter two both came to Oxford as SRC-funded research fellows, originally with a brief to work on the construction and publication of high-quality software.

As for the students, they:

> will obtain a practical understanding of the entire development task, from abstract, user-oriented, specification, through concrete programming, down to [...] the delivered system. Thus we hope not to perpetuate problems of inadequate software.

Formal teaching was to take place in the first six months, leading to an examination with two three-hour papers. Students who passed the examination would spend the second six months on a project and the writing of a dissertation.

A glance at the curriculum and the teachers for the first year the degree was delivered will confirm that Tony was indeed 'letting good people get on with it.' and that he was not short of further volunteers. The content and style would set the tone for the degree for many years.

Program Specification: In this course Jean-Raymond Abrial used the earliest (pre schema-calculus) variant of the evolving Z specification notation [1] to introduce logic, typed set theory, and the theory of relations, as well as material on ordering, domains, and fixed points. The course advocated constructing and reasoning about the properties of abstract specifications of a system before beginning to think about the details of its implementation, and a few case studies were presented.[32]

Functional Programming: In this course Joe Stoy used a dialect of Scheme to 'Describe the semantics of programming languages and to describe and solve other computing problems.' The lecture synopses included 'Fixed points and recursion. Lists and data structures.'

Principles of Programming: Cliff Jones taught this course, in which he introduced the rigorous approach of the Vienna Development Method expounded in his book [25]. Key aspects of the course were the abstract modelling of the state of a system, including any invariants; the prealgorithmic specification of operations by pre- and post- conditions; the notion of data refinement, and the rules for validating refinements.

Distributed Computing: Tony himself taught this, based on the synopsis: 'Processes, communication, process structures, proof techniques, localisation, sharing, nondeterminism, discrete event simulation.' His lecture notes would eventually be developed into his CSP book [19].

Programming Language Definition and Implementation: Joe Stoy and the present author taught this two-part course.[33] The first part used the methods of denotational semantics to consider the validity of proof rules and the principles for establishing the correctness of a compiler. The second part presented a couple of compilers targeted at abstract stack machines.

[32] In the second and subsequent years the present author would deliver a follow-on course consisting entirely of case studies, including a batch operating system, a reliable block storage system, and a WYSIWYG text editor.

[33] Students were encouraged, but not required, to attend Dana Scott's *Programming Language Theory* course shared with undergraduate mathematicians that presented the foundations of Denotational Semantics.

Software Management: John Buckle taught this course and its successors for several years. It was based on the synopsis: 'Project phases, tools, plans, budgets, reviews, changes, documentation, staffing, interfaces.' [7]

Miscellaneous Applications: This was an optional supplementary course. Four topics had to be chosen from: Pattern Recognition[34]; Control Systems;[35] Real-time Programming (in Ada);[36] Data Processing;[37] Numerical Algorithms Libraries.[38]

Microprocessor Software and Hardware: This was an optional supplementary course with two practically-based parts:[39] (1) High Level Language Programming — the construction of modular systems in UCSD Pascal to run on DEC LSI/11 computers. (2) Machine Level Programming — on the architecture and machine-coding of a Research Machines RM380Z, based on a Z80 processor.

The first cohort consisted of 'a communications engineer, a data-processing manager, a computer-science graduate, and two programmers with industrial experience.' The second cohort had 'four overseas and four UK students, with a remarkable spread of expertise — three being sponsored by the software house SPL.' The success of all but one of these students, and the quality of some of their projects (*e.g.* [10]) demonstrated that an M.Sc. was viable. But it was apparent quite early that a voluntaristic approach to teaching and project supervision would not be sustainable if the course was ever to be expanded further or to play an economically useful role in the UK.

10.3.4 Microprocessors to the Rescue

During the first year of the M.Sc. government funds became available for UK universities to 'Strengthen the teaching of the applications of microelectronics and microprocessors at undergraduate and postgraduate levels, particularly to engineers and scientists.' Tony seized this opportunity, and arranged for a joint bid to be forwarded by the University from the PRG and the Digital Design Group of the Engineering Science Department, for a university lectureship each.[40]

[34] Taught by Frank Harris, director of the High-Energy Physics imaging laboratory, on a variety of applications that were being pursued in the Physics Department, namely map data processing; scanning and interpretation of engineering drawings; bubble-chamber image processing.

[35] Taught by David Clarke, Reader in Engineering Science and head of the Digital Design group.

[36] Taught by John Barnes: designer of the real-time programming language RTL/2, and primary inventor of the Ada Rendezvous mechanism.

[37] Taught by Michael Jackson using his (now-classic) book: Principles of Program Design [24].

[38] Taught by Bryan Ford, managing director of the Numerical Algorithms Group.

[39] Taught by Jim Kaubisch

[40] The fact that a timely and successful bid was made is testament to Tony's organisational skill and powers of persuasion. The departments were in different faculties — Mathematics, and Engineering and Physical Sciences — both of which needed to be convinced of its worth; and a college fellowship had to be brokered in advance for each

The successful bid had committed the PRG and the engineers to teach microprocessor applications to the M.Sc. students. Tony deftly revised the publicity material accordingly

> The past decade has seen two major revolutions in computing, one in hardware and one in programming.
>
> 1) The invention of the microprocessor has reduced the unit cost of a stored-program computer by several orders of magnitude, with a corresponding increase in the range of its potential applications.
>
> 2) The discovery of the mathematical basis of computer programming now permits construction of programs that are proved to meet their specification, and promises to transform programming from an arcane and error-prone craft into a modern engineering profession.
>
> [...]
>
> Thus we hope not to perpetuate on microprocessors the problems of unreliable and inadequate software sometimes associated with the use of conventional computers.

The curriculum was now reorganised under three main headings: Programming Methodology, Microprocessor Applications, and Project Management. Tony located his CSP course *Distributed Computing* under Microprocessor Applications, with a prophetic rationale:

> In large scale applications it is expected that microprocessors will be connected in networks cooperating on a common task. Traditional methods of system design can be extended to take advantage of these possibilities.

The three Programming Methodology courses: *System Specification*, *Programming Language Principles*, and *Program Correctness and Validation* were to be revised variants of earlier courses.[41]

Students' employers were encouraged to discuss project topics with the Director of the course; though there were a couple of safeguards against employer-funded student-programmers being hauled back to do their old job in the old way

> The student will be expected to put into practice [in the project] the principles and techniques learned during the [lectures]. The project will usually involve microprocessors, of which there is a plentiful supply. The choice of project may be made in

---

of the lecturers. Eventually Peter Henderson was appointed to the PRG post — nominally as Director of the M.Sc., and to a fellowship of the all-graduate St. Cross College. Arthur Dexter was appointed to the engineering post and to a tutorial fellowship in Engineering at Worcester College.

[41] The optional courses of the earlier curriculum were replaced by optional short introductory courses in Pascal, Electronics for Microprocessors, and Microprocessor Coding offered during the first couple of weeks of term, along with a lengthier course in Functional Programming.

> consultation with the student's employer; and where convenient the latter part of the work can be carried out away from Oxford.[42]

The 1981-1982 cohort attracted by the new curriculum consisted of sixteen students — particularly good in view of the fact that all the UK students were self-funded or supported by their employers.[43]

But with several more doctoral students and research staff, the PRG had by now overflowed its allocated space. Tony warned the powers that be: 'Plans for a move to more spacious accommodation in Keble Road were made a long time ago, and have been frequently reconsidered and reconfirmed. They must not now be put into doubt. It is bad policy to accept students and house them in substandard accommodation.' Happily, the University stuck with the plans, and by late 1982 the entire Computing Laboratory had been rehoused in four huge refurbished terraced houses at 8-11 Keble Road.[44]

10.3.5 An Engineering Profession

In 1982, concluding his widely circulated 23-page manifesto *Programming is an Engineering Profession*[18] written, among other reasons, to attract students and employers to the course, Tony compared the modern profession of programming to the earlier profession of Civil Engineering:

> Computer programmers work with neither energy nor materials, but with a more intangible concept. We are concerned with the capture, storage, and processing of information. When the nature of our activities is more widely understood, both within our profession and outside, then we shall deservedly be recognised and respected as a branch of engineering. And I believe that in our branch of engineering, above all others, the academic ideals of rigour and elegance will pay the highest dividends in the practical terms of reducing costs, increasing performance, and in directing the great sources of computational power on the surface of a silicon chip to the use and convenience of man.

By 1983-1984 the M.Sc. had Specialist and Conversion streams: the former for professionals or graduates in computing (the taught material was seen as advanced by comparison with most

[42] Several employers took advantage of this offer. Some companies would later subscribe to a scheme which allowed them, for a fee, to propose projects that any M.Sc. student could pursue with a suitable supervisor, and be rewarded by a significant (£500 or so) bursary.

[43] In this period fees and living expenses for UK graduate students were funded by the SRC if they were awarded Advanced Course Studentships by their host department; departments were allocated a fixed number of studentships (sometimes zero) in advance each year by the council for courses that it approved; determined by a policy that appeared inscrutable and capricious. The department could appeal for an off-quota studentship for particular students, but the appeals were dealt with in first-come first-served order, following an even less scrutable policy.

[44] They still form part of its premises today.

computing degrees), the latter for mathematics and science graduates. As well as making it attractive to a wider range of potential students, this made it easier for the PRG to get SRC scholarships for UK students as government funding preferences oscillated between 'training' and 'retraining'; and for a while every UK student who attended the course had their fees and a living allowance paid. The programme would continue its early growth and its appeal to overseas students, and many of the large numbers of students who now attend are sponsored by employers.

10.3.6 Breaking the Ice: Towards Undergraduate Degrees

By 1981 Tony was becoming frustrated with the difficulties he was having in trying to establish an undergraduate degree:

> Do you remember when the vice-Chancellor visited 45 Banbury Road to just look around? John Barnes (of Ada fame) told him that Oxford should develop its Computer Science degrees. He turned to me and said 'That's the job for you, Tony'. After five years we had an established MSc already, but I had abandoned hope of an undergraduate course. Until [Prime Minister] Mrs. Thatcher came to the rescue!

The root of the problem was the strict limit then placed by government, its primary source of funds, on the total number of undergraduates Oxford was allowed to admit, which translated into strict quotas on numbers of undergraduates in each college. After a new degree proposal had been agreed as a matter of faculty and university policy,[45] colleges had to be found that would commit places and fellowship stipends to it. The protocol was for the University secretariat to describe it, along with any additional resources and student quota that would go with it, in an invitation for 'Expressions of Interest' (i.e., bids) sent to the undergraduate colleges. If no new resources or student quota came with the degree, then a zero-sum game was played out in each college as its governing body considered making a bid. Established subjects had the advantage in this game: existing tutorial fellows would have the voting strength to veto an innovation that they thought would pose a challenge to the college quota of students in their favoured subjects, however economically or intellectually important the innovation was. All the incentives favoured slow change if not outright stasis: a reduction in numbers could lead to the eventual loss of a tutorial position, or even of a whole subject.[46]

[45] By the General Board of the Faculties, a body elected by the Congregation ($\approx$ all tenured academics) that was responsible for the academic and educational policy of the University.

[46] The colleges are self-governing and independent of the University. The governing bodies of undergraduate colleges would have a majority of tutorial Fellows in existing subjects. Many were sceptical that computing could be treated rigorously, even if they didn't feel threatened by it. And as if this were not enough of a hazard, faculty boards and the General Board were reluctant to agree to initiatives that ran the risk of attracting no bids, and their more conservative members would use that risk as a counterargument to the innovation. Seventy five years or so earlier, Engineering had to overcome analogous hazards to gain acceptance.

Prime Minister Thatcher's unwitting rescue came in the form of the substantially funded Alvey Programme,[47] which changed the way computing research was organised and financed in the UK for a few years. The immediate benefit to Oxford was the funding of several 'new blood' University Lectureships in computation, as well as a game-changing commitment by government to expand student numbers by ten places for each college that appointed a tutorial fellow in Computation.[48] A few colleges got the message, and by 1985 eight university lecturers with tutorial fellowships were in post. This made a joint degree in Mathematics & Computation feasible.[49]

It also broadened the existing research interests and expertise of the group to include algorithms and hardware.

10.3.7 Curriculum Innovation in the Honour School of Mathematics & Computation

Tony chaired the first meeting that considered the PRG contribution to the curriculum of the *Mathematics & Computation* degree in detail. As was typical for him, everyone who might have something interesting to say was invited, not just the teaching staff. In a passionate though good-tempered discussion that everybody present knew would have big ramifications, the group considered how programming would be approached *ab initio* in the first year. It would have been easy to emulate Belfast, adopt what had become the standard pattern of a CS curriculum, and teach programming in Pascal. But most agreed that something more radical would be possible, especially in light of the expertise in functional programming and its implementation present in the group.[50]

The argument for teaching functional programming first was straightforward: the overall approach to computation was to be constructive, not reductive; about information structures, not just bits and bytes and arrays. There would be no harm at all in students with no computing experience approaching computing this way; and as for those who would arrive at Oxford

[47] The Alvey Programme was worth £350m at 1982 prices

[48] Tony says he received the news by telegram from Oxford while visiting Wollongong University during a trip to Australia. Tony's 1982 election as a Fellow of the Royal Society, and the increasing success of the PRG in working with industry may have played a catalytic role in this: industry leaders frequently talked to senior civil servants.

[49] The first colleges to appoint tutorial Fellows were Balliol (Joe Stoy), University (Bill Roscoe), and Worcester (the present author) — in 1983. Over the next two years Lady Margaret Hall (Mary Sheeran, soon to be succeeded by Jeff Sanders), Pembroke (Carroll Morgan), St. Edmund Hall (John Hughes, soon to be succeeded by Mike Reed), St. Hugh's (Ian Page), and Wadham (Bill McColl) followed — each of the latter group had partnered with another college prepared to pay half the tutorial stipend in exchange for being allowed to admit two students per year. Richard Bird, and Ib Holm Sørensen were also appointed to University Lectureships in Computation in 1983: the former to replace Peter Henderson as Director of the M.Sc. with a fellowship at St. Cross; the latter to pursue Industrial Liaison, with a fellowship at Wolfson.

[50] Richard Bird, Geraint Jones, John Hughes, Phil Wadler, and Joe Stoy were all at the PRG, as was Martín Raskovsky. Quentin Miller, and Peter Hancock were soon to join. Richard's principled approach to constructive functional programming had already been very influential[5, 6], as had Peter Henderson's pragmatic approach [13]. The present author had been using ML, CAML, and Scheme for several years in projects, and had supervised John Hughes's D.Phil.

with basic (or Basic) computing experiences from school or recreational computers, it was important to dispel the idea that university computing was much the same as they had already experienced, but on bigger and faster machines. By the end of the discussion it had been decided to teach functional programming in the first term, using a lazy functional language.

Tony enthusiastically supported the decision: he would go on to devote significant resources to the development of a new polymorphically-typed lazy functional language, Orwell, to be used for teaching. Richard Bird and Phil Wadler wrote a textbook for the course ([4]) while the language was being implemented by Quentin Miller and Martín Raskovsky. [51] [52]

When it came to imperative programming the consensus was that programming could be treated as a rigorous calculational discipline, and it was decided to delay the associated course, Software Engineering, until the start of the second year, when the students would be mathematically more mature. It would use the refinement calculus notation then being developed by Carroll Morgan.

The first year computation section of the course — a fifth of the material — would also include an elementary computer architecture course that discussed topics ranging from the structure of a simple computer, through data representations, and instruction sets, to basic (CMOS) digital circuits.[53] Other courses were shared with Mathematics, including an eight-lecture course on propositional and first-order logic. For the first couple of years the lecturer was the distinguished logician Robin Gandy, a former doctoral student of Turing.

The second year computation sections of the course — two fifths of the material — would be examined on two papers: *Software Engineering and VLSI Design*, and *Algorithm Design and Distributed Computing*, and a short section on computability was shared with the mathematicians. The VLSI Design course was based on Mead & Conway's textbook [27]. The Algorithm Design course was based on Robert Sedgewick's textbook, *Algorithms*, and the syllabus for the Distributed Computing course shows that it was based on CSP and that its practicals used the FDR refinement-checker. The Software Engineering course would have practicals in Modula II, but no formal instruction in the language. It demonstrated for some years that someone who has grasped the ideas of data abstraction and program refinement can easily pick up the specifics of a particular imperative programming language and use it to write reliable programs; but its approach to the refinement of programs from their specifications was (mistakenly) seen as too rigorous for the less calculationally inclined students, and was eventually moderated.

[51] Orwell was a precursor of Haskell. The language, indeed the whole approach, was influenced by David Turner's work, but the local functional programmers had a distinctive view about notational and semantic details.

[52] Orwell and the Bird&Wadler text were used after the course had been taught for the first time by Joe Stoy using a Scheme dialect, called T, for which an implementation and detailed tutorial materials were immediately available.

[53] Are you thinking 'So much for eschewing a reductive approach?' Interestingly, the practicals for this course required the students to simulate hardware ... in the functional language they had been taught.

The idea of starting the degree with a functional programming course would be seen as intrepid at the time, when not attacked openly as folly, by Oxford's 'competitors'.[54] But good students have usually thrived on it; because it soon shows how they can use straightforward methods of calculation and proof, analogous to those they are familiar with from school, to derive and/or verify serious and interesting programs and circuits.

10.3.8 A Delayed Inaugural

Oxford duly went about recruiting undergraduates: the first admissions interviews were held in December 1984,[55] and in the first week of the Michaelmas term of 1985 Tony delivered his long-delayed inaugural lecture, *The Mathematics of Programming*, to the first cohort of students and an assortment of colleagues and invited grandees[20]. Those present in the University Museum for the lecture still remember Tony's slow-burn prelude to the technical section, concluding:

> Our principles may be summarized under four headings.
>
> 1. Computers are mathematical machines. ...
> 2. Computer programs are mathematical expressions. ...
> 3. A programming language is a mathematical theory. ...
> 4. Programming is a mathematical activity. ...
>
> These are general philosophical and moral principles, and I hold them to be self-evident — which is just as well, because all the actual evidence is against them.
> [...]
> Many programmers of the present day have been educated in ignorance or even fear of mathematics. Of course there are many programmers who are university graduates in [mathematics]. They may have acquired a good grasp of topology, calculus, or group theory. But it never occurs to them to take advantage of their mathematical skills to define a programming problem and search for its solution.

[54] The attacks now seem quaintly rooted in earlier machinery and sensibilities: programs would be 'too inefficient', inductive reasoning and polymorphic types would be 'too hard', 'try and persuade someone in industry to program this way'. Nevertheless a year later the present author would be teaching 5-day functional programming courses twice a year to senior IBM programmers, as part of their continuing professional development. The second cohort were so enthusiastic that they adapted a mainframe IBM operating system so that Orwell could be used as one of its shell/job-control languages. Nowadays functional programming is widespread in industry and commerce.

[55] This wasn't completely straightforward! At first no college was allowed to admit more than two Mathematics & Computation students per year. Although the subject tutors would normally have interviewed candidates in their colleges and made their own admissions decisions, a cumbersome 'quota panel' was set up by the faculty board to interview them again at the PRG — nominally to ensure consistency of the college decisions. The panel had been established because some mathematicians feared that computation would not be a rigorous enough subject to attract students of comparable intellect to their mathematics students — a case of unfamiliarity breeding contempt. But Tony's powers of persuasion, helped by the very strong academic performance of many of the earliest students of the joint school, would eventually convince the doubters that this administrative albatross could be dispensed with.

> [...]
> What, then, are the laws of programming that help the programmer to control the complexity of their task? Many programmers would be hard pressed to name a single law. Those who have suffered from bad programs might claim that programmers are such an undisciplined crew that even if they knew any laws they would instantly violate them.

Thirty five years later this marvelously comprehensive and farsighted lecture is well worth re-reading.

10.3.9 From Computation to Computer Science: Two New Degrees

Once the ice had been broken in Oxford, and the high quality of the students wanting to study computing had been definitively established, two new honour schools were straightforwardly established over the next five years, and new staff appointed to support their teaching.

The increasing role played by digital systems in engineering, and the founding of an influential *Information Engineering* group[56] within the Department of Engineering Science had provided many opportunities for research collaboration, and it became obvious that the collaboration could be beneficially be extended to undergraduate teaching. Negotiations between the PRG and Engineering Science led to the founding of a new, 4 year, honour school of *Engineering and Computer Science* (ECS). A new statutory professorship was established in the laboratory in 1988 to support this school,[57] and the first students were admitted in 1987.[58]

The honour school of *Computer Science* was established a few years later. Agitation by North American colleagues, though resisted by PRG traditionalists, had led inexorably to the use of '*Computer Science*' rather than '*Computation*', or '*Computing Science*' in the names of all the honour schools in whose teaching the PRG was involved.

---

[56] Led by J.M. Brady, a polymath mathematician, computing scientist, and engineer, who has since his formal retirement been Professor of Oncological Imaging at Oxford.

[57] The first incumbent was Joseph Goguen, of SRI International. The second was Richard Brent of the Australian National University.

[58] ECS students only received their first lectures in computer science in the second year of their degree, and this hampered their capacity to use sophisticated computational thinking in later parts of that degree. The Engineering Science Department's long-established doctrine that all its undergraduates must take a common first year of instruction, with topics rooted in the material and physical aspects of engineering, including Soil Mechanics, had — despite Tony's negotiating skill — proven non-negotiable. Although it seems that tradition eventually gave way to logic, this wasn't in time to save the joint degree, which despite the brilliance of its students had proven unpopular with some unreconstructable 'old-school' engineering tutors: the Degree was quietly abolished after less than a decade.

10.4 Valediction

In 1995, four years before he retired from Oxford and went to work for Microsoft Research, Tony reflected on the original assumptions behind the programme of research and education that had been one of the mainstays of his department.

> Ten years ago, researchers into formal methods (and I was the most mistaken among them) predicted that the programming world would embrace with gratitude every assistance promised by formalisation to solve the problems of reliability that arise when programs get large and more safety-critical. Programs have now got very large and very critical — well beyond the scale which can be comfortably tackled by formal methods. There have been many problems and failures, but these have nearly always been attributable to inadequate analysis of requirements or inadequate management control. It has turned out that the world just does not suffer significantly from the kind of problem that our research was originally intended to solve. [21]

Although very carefully worded, Tony's original remark gained long-term prominence[59] and was taken by too many computing educators as a licence to drop the study of all but the dampest of formal development methods from the core computing science curriculum.[60] Reflecting on the same issue in 2006 in light of subsequent developments he described the changed situation brought about by the ubiquity of computing and importance of security:

> I thought that when I retired, it would be very interesting to see whether the positive side of my prediction would also come true, that these ideas would begin to be applied. And indeed, even since I've joined Microsoft in 1999, I've seen quite a bit of expansion in their use of assertions and other techniques for improving confidence in the reliability of programs. There are program analysis tools now that stop far short of actually proving correctness of programs, but they're very good at detecting certain kinds of errors. Some quite dangerous errors, which make the software vulnerable to intrusions and virus attacks, have been detected and removed as a result of the use of formal techniques of program analysis.
>
> The idea of verifying computer programs is still an idea for the future, although there are beginnings of using scientific technology to make the programs more reliable. The full functional verification of a computer program against formally specified requirements is still something we have to look forward to in the future.

[59] *Inter alia* by being quoted in the 'Apologies and Retractions' section of his Wikipedia entry in which, surprisingly, it has remained without further comment.

[60] Carroll Morgan has since shown very effectively how a start can be made at reversing this. [29].

> But the progress that we've made has really been quite spectacular in the last 10 years. [36]

The revolution since brought about by the increasing dependence of society on distributed applications based on Cloud infrastructure led to providers turning to formal verification on a very large scale

> Firstly many security-focused customers (*e.g.* financial services, government, pharmaceutical) are moving workloads from their own data centers to AWS. Formal verification provides customers from these industries with concrete information about how security is established in Amazon Web Services. Secondly, automatic and continuous formal verification facilitates rapid and cost-efficient development by a distributed team of developers. [3, 8]

The intellectual tools underpinning this turn have not been superseded; indeed they have become increasingly useful as tool support for them has become more effective. There really is no good reason now for serious students of computing, or students of serious computing to be denied the insights and creative leverage they provide.

So we can imagine what Tony would do now if given a free hand in the design of a modern undergraduate computing curriculum incorporating the ideas and laws he played such an important role in discovering and promulgating, as scientist and as educator.

But perhaps we should ask him!

10.5 Acknowledgements

Many thanks to Tony Hoare for sharing reminiscences of his personal and professional history with me. Jim Welsh and Mike McKeag, Tony's colleagues at Queen's, provided invaluable information about their department during Tony's time there. Joe Stoy generously recollected the prehistory of Tony's appointment to Oxford. The transcript [22] of Cliff Jones's 2016 interview with Tony [23] provides horse's-mouth witness for some anecdotes related here, and will perhaps thereby endow others with the sheen of verisimilitude. The 1969 — 1984 minute book of Oxford's sub-Faculty of Computation, and its 1985 *Examination Decrees and Regulations* provided authoritative information about Oxford organization and curriculum. Unattributed quotes are either from the above-mentioned minute book, or from my personal correspondence with Tony during 2019.

Bibliography

[1] J.-R. Abrial, S. Schuman, and B. Meyer. 1980. Specification language. *RM McKeag and AM MacNaughten, editors On the Construction of Programs: An Advanced Course*, p. 343.

[2] Anon. ABOUT THE (UNIVERSITY OF OXFORD) DEPARTMENT OF COMPUTER SCIENCE. http://www.cs.ox.ac.uk/aboutus/cshistory.html.

[3] J. Backes, P. Bolignano, B. Cook, A. Gacek, K. S. Luckow, N. Rungta, M. Schaef, C. Schlesinger, R. Tanash, C. Varming, and M. Whalen. 2019. One-click formal methods. *IEEE Software*, 36(6): 61–65. https://doi.org/10.1109/MS.2019.2930609. DOI: 10.1109/MS.2019.2930609.

[4] R. Bird and P. Wadler. 1988. *Introduction to Functional Programming*. Prentice Hall International Series in Computer Science. Prentice Hall.

[5] R. S. Bird. Oct 1986. *An Introduction to the Theory of Lists*. Technical Monograph PRG56, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[6] R. S. Bird. Oct 1988. *Lectures on Constructive Functional Programming*. Technical Monograph PRG69, Oxford University Computing Laboratory, Programming Research Group, Oxford, England. https://doi.org/10.1007/978-3-642-74884-4-5. DOI: 10.1007/978-3-642-74884-4-5.

[7] J. K. Buckle. 1977. *Managing Software Projects.* Macdonald.

[8] A. Chudnov, N. Collins, B. Cook, J. Dodds, B. Huffman, C. MacCárthaigh, S. Magill, E. Mertens, E. Mullen, S. Tasiran, A. Tomb, and W. E. 2018. Continuous formal verification of Amazon s2n. In *International Conference on Computer Aided Verification*, pp. 430–446. Springer.

[9] O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, eds. 1972. *Structured Programming*. Academic Press Ltd., England. ISBN 0122005503.

[10] E. Fielding. 1980. *The Specification of Abstract Mappings and their Implementation as B Trees*. Technical Monograph PRG18, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[11] L. Fox. 1966. *Advances in Programming & Non-Numerical Computation (Proceedings of a Summer School)*. Pergamon Press.

[12] R. Gimson and C. Morgan. 1985. *The Distributed Computing Software Project*. Technical Monograph PRG50, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[13] P. Henderson. 1982. *Functional Programming — Application and Implementation*. Prentice Hall International Series in Computer Science. Prentice Hall.

[14] C. A. R. Hoare. 1961. Algorithm 63, Partition; Algorithm 64, Quicksort; Algorithm 65, Find. *Communications of the ACM*, 4(7): 321.

[15] C. A. R. Hoare. Jan. 1962. QUICKSORT. *The Computer Journal*, 5(1): 10–16. ISSN 0010-4620. https://doi.org/10.1093/comjnl/5.1.10. DOI: 10.1093/comjnl/5.1.10.

[16] C. A. R. Hoare. 1971. *Computer Science: an inaugural lecture delivered before the Queen's University of Belfast on 10 February 1971*. Queen's University.

[17] C. A. R. Hoare. Feb. 1981. The Emperor's Old Clothes. *Communications of the ACM*, 24(2): 75–83. ISSN 0001-0782. https://doi.org/10.1145/358549.358561. DOI: 10.1145/358549. 35856.

[18] C. A. R. Hoare. 1982. *Programming is an Engineering Profession*. Technical Monograph PRG27, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[19] C. A. R. Hoare. 1985. *Communicating Sequential Processes*. Prentice Hall International Series in Computer Science. Prentice Hall, Englewood Cliffs, NJJ. ISBN 978-0-13-153271-7. http://www.usingcsp.com/cspbook.pdf.

[20] C. A. R. Hoare. 1986. *The Mathematics of Programming*. Oxford University Press.

[21] C. A. R. Hoare. 1995. Unification of theories: A challenge for computing science. In *Recent Trends in Data Type Specification*, pp. 49–57. Springer.

[22] C. A. R. Hoare and C. B. Jones, Nov 2015. Transcript of an interview with Tony Hoare, ACM 1980 Turing Award Recipient (Interviewer: Cliff Jones, Newcastle University). https://amturing.acm.org/pdf/HoareTuringTranscript.pdf.

[23] C. A. R. Hoare and C. B. Jones, November 2015. An interview with Tony Hoare, ACM 1980 Turing Award Recipient (Interviewer: Cliff Jones, Newcastle University). https://www.youtube.com/watch?v=tAl6wzDTrJA.

[24] M. A. Jackson. 1975. *Principles of Program Design*. Academic Press.

[25] C. B. Jones. 1980. *Software Development: a Rigorous Approach*. Prentice Hall International Series in Computer Science. Prentice Hall.

[26] C. B. Jones and A. W. Roscoe. 2010. Insight, inspiration and collaboration. In A. Roscoe, C. B. Jones, and K. R. Wood, eds., *Reflections on the Work of C.A.R. Hoare*, pp. 1–32. Springer London, London. ISBN 978-1-84882-912-1. https://doi.org/10.1007/978-1-84882-912-1_1. DOI: 10.1007/978-1-84882-912-1_1.

[27] C. Mead and L. Conway. 1980. *Introduction to VLSI Systems*. Addison-Wesley, Reading, Massachusetts.

[28] R. Milne and C. Strachey. 1976. *A Theory of Programming Language Semantics*. Chapman and Hall, London.

[29] C. Morgan. 2016. (In-)Formal Methods: the Lost Art. In *Engineering Trustworthy Software Systems*, pp. 1–79. Springer.

[30] P. Mosses. 1974. *Commentary on the Mathematical Semantics of Algol 60*. Technical Monograph PRG12-Commentary, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[31] P. Mosses. 1974. *The Mathematical Semantics of Algol 60*. Technical Monograph PRG12, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[32] P. D. Mosses. 1975. *Mathematical Semantics and Compiler Generation*. D.Phil Thesis, University of Oxford, Oxford, England.

[33] P. D. Mosses. 1979. SIS—semantics implementation system. *Reference Manual and user guide. Report DAIMI MD-30, Univ. Aarhus*.

[34] D. S. Scott. 1970. *Outline of a Mathematical Theory of Computation* . Technical Monograph PRG2, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[35] D. S. Scott and C. Strachey. 1971. *Toward a Mathematical Semantics for Computer Languages*. Technical Monograph PRG6, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[36] L. Shustek. 2009. An interview with CAR Hoare (Interview conducted by J.P. Bowen in late 2006). *Communications of the ACM*, 52(3): 38–41. https://doi.org/10.1145/1467247.1467261. DOI: 10.1145/1467247.1467261.

[37] J. E. Stoy. 1977. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, Cambridge, MA, USA. ISBN 0262690764.

[38] J. E. Stoy and C. Strachey. 1972. OS6—An experimental operating system for a small computer. Part 1: General principles and structure. *The Computer Journal*, 15(2): 117–124. https://doi.org/10.1093/comjnl/15.2.117. DOI: 10.1093/comjnl/15.2.117.

[39] J. E. Stoy and C. Strachey. 1972. OS6—An experimental operating system for a small computer. Part 2: Input/output and filing system. *The Computer Journal*, 15(3): 195–203. https://doi.org/10.1093/comjnl/15.3.195. DOI: 10.1093/comjnl/15.3.195.

[40] J. E. Stoy and C. Strachey. 1972. *OS/Pub (text and commentary)*. Technical Monograph PRG9, Oxford University Computing Laboratory, Programming Research Group, Oxford, England.

[41] C. Strachey. 2000. Fundamental Concepts in Programming Languages (republished 1967 lecture notes). *Higher-order and Symbolic Computation*, 13(1-2): 11–49.

[42] C. Strachey and C. P. Wadsworth. 2000. Continuations: A mathematical semantics for handling full jumps (republished 1974 monograph). *Higher-order and Symbolic Computation*, 13(1-2): 135–152.

[43] B. Sufrin, 2007. Oxford University Computing Laboratory: an informal prehistory – a lecture given at the 50th anniversary celebration of the founding of the laboratory. https://www.cs.ox.ac.uk/people/bernard.sufrin/personal/historyfortalk.pdf.

[44] R. D. Tennent. Sept. 1976. The denotational semantics of programming languages. *Communications of the ACM*, 19(8): 437–453.

[45] J. Welsh and D. Bustard. Sept. 1979. Pascal Plus - another language for modular multiprogramming. *Software: Practice and Experience*. https://doi.org/10.1002/spe.4380091109. DOI: 10.1002/spe.4380091109.

[46] M. V. Wilkes, D. J. Wheeler, and S. Gill. 1951. *The Preparation of Programs for an Electronic Digital Computer, With Special Reference to the EDSAC and the Use of a Library of Subroutines*. Addison-Wesley.

[47] N. Wirth and C. A. R. Hoare. June 1966. A contribution to the development of ALGOL. *Communications of the ACM*, 9(6): 413—-432. ISSN 0001-0782. https://doi.org/10.1145/365696.365702. DOI: 10.1145/365696.365702.

Biographical Note

Bernard Sufrin is currently an Emeritus Fellow of the Department of Computer Science and of Worcester College in Oxford – where in 2018 the senior Tutorial Fellowship in Computer Science was permanently endowed, and named after him.

Between 1982 and 2010 he was University Lecturer in Computation and Tutorial Fellow in Computation at Worcester College. Since his first approximation to a formal retirement, in 2011, he has lectured in his former department on formal proof and on concurrent programming, supervised occasional research projects, and been tutor in Computer Science at Magdalen College.

He spent 1973-74 at Bolt, Beranek, and Newman in Cambridge, Mass. where he participated in the development of the second generation Arpanet. He returned from the US to the University of Essex as Chief Research Officer, then moved to Oxford in 1978 as an SRC Research Fellow.