An Interview with
Tony Hoare
ACM 1980 A.M. Turing Award Recipient
(Interviewer: Cliff Jones, Newcastle University)
At Tony's home in Cambridge
November 24, 2015

CJ = Cliff Jones (Interviewer)

TH = Tony Hoare, 1980 A.M. Turing Award Recipient

CJ: This is a video interview of Tony Hoare for the ACM Turing Award Winners project. Tony received the award in 1980. My name is Cliff Jones and my aim is to suggest an order that might help the audience understand Tony's long, varied, and influential career. The date today is November 24th, 2015, and we're sitting in Tony and Jill's house in Cambridge, UK.

Tony, I wonder if we could just start by clarifying your name. Initials 'C. A. R.', but always 'Tony'.

TH: My original name with which I was baptised was Charles Antony Richard Hoare. And originally my parents called me 'Charles Antony', but they abbreviated that quite quickly to 'Antony'. My family always called me 'Antony', but when I went to school, I think I moved informally to 'Tony'. And I didn't move officially to 'Tony' until I retired and I thought 'Sir Tony' would sound better than 'Sir Antony'.

CJ: Right. If you agree, I'd like to structure the discussion around the year 1980 when you got the Turing Award. I think it would be useful for the audience to understand just how much you've done since that award. So if I could, I'd like to start in 1980 and work backwards, and later on we'll come to 1980 and work in the more obvious order if that's okay.

TH: That's fine. Thank you.

CJ: So the Turing citation lists four things, not necessarily in this order – the axiomatic approach; design of algorithms, specifically Quicksort; contributions to programming languages in general; and operating systems constructs such as monitors.

Let's begin with the axiomatic approach. The key paper you wrote in 1968 I think.

TH: That's right. When I moved to Belfast as a professor.

CJ: Yes, we'll come to Belfast later on. Can you, for anybody who doesn't know, describe Hoare triples?

TH: "Hoare triples" is just a symbolic way of saying something quite simple. It's a statement about what will happen if you do something. It has three parts, as you would expect from the 'triple'. The first part is called a precondition, and that begins, 'If something or other is the case in the real world', and the second part is the program itself, which is an active verb, is that 'If you do this, then the final stage of the world after you've done it will satisfy the third component of the triple', which is called a post-condition.

CJ: Now that's what it was. Can you tell us what problem you were trying to solve when you came up with the Hoare triple?

TH: Well, I had the idea that it would be a good idea to define programming languages in a way that didn't say too much about what the computer actually did, because in those days anyway all computers were doing things slightly differently, but gave enough information to the user of a programming language to be able to predict whether the computer would do what the programmer wanted it to do. What the programmer wanted it to do was expressed as the post-condition and served as a specification for the program in the middle, but very usually the program wouldn't work in all circumstances and required to be started in a state in which the precondition also held. So what I was trying to do is to construct a formal proof system, calling on my previous acquaintance and love of logic, which would justify a formal proof, a mathematical proof that the program actually does what the programmer wanted.

CJ: Maybe you could say a bit more about the context of the work at that time. I know from this famous 1969 publication in Communications of the ACM, you make very generous acknowledgements to Floyd[1], Naur[2], van Wijngaarden[3], and so on. But could you say what other people were trying to do with language definitions at the

---

[1] Robert W (Bob) Floyd (1936 – 2001) also won a Turing Award in 1978. He was a pioneer in the field of program verification and his 1967 paper *Assigning Meanings to Programs* (*Proceedings of American Mathematical Society*, Vol. 19, pp. 19–32) was an important contribution to what later became Hoare logic.

[2] Peter Naur (1928 – 2016) was a Danish computer science pioneer and also Turing award winner.

[3] Adriaan van Wijngaarden (1916 – 1987) was a Dutch mathematician and computer scientist who was head of the Computing Department of the Mathematisch Centrum in Amsterdam. He is widely considered to be founder of computer science in the Netherlands

time you came up with your idea?

TH: Yes. There were two ideas of how to define a programming language current. One was the denotational semantics, which attempted to describe what the meaning of the program was in terms that were familiar to mathematicians – for example, using the mathematical concept of a function – and the other one was an operational semantics, which was more appealing to the programmer who likes to know how the computer's actually going to execute the program. I was out of sympathy with… I couldn't understand the first of them and I was out of sympathy with the second. [chuckles] So I came up with this third approach which is called the axiomatic approach, which has attracted quite a bit of attention.

CJ: Well, we'll draw a lot of parallels later on with your later work, but let's come to that later. Baden-bei-Wien, the formal language description languages conference, there were a lot of papers there. None of them were using the approach or hinting at the approach that you were to pioneer?

TH: I think none of them were. I remember standing up to ask a question and using it as an excuse to make a comment that I felt that one of the main advantages of a formal language description language was to be able to say as little as possible, as little as possible and as much as necessary of course, about the details of the language itself. And I gave an example of defining the modulus of a number as being… What? Sorry, I've forgotten. [chuckles] Anyway, let's leave that.

CJ: I know that you also went to the IBM Vienna Lab and heard the course, the presentations on their extremely large attempt to use an operational semantics approach to define PL/I. Were you on the ECMA standards committee or…?

TH: I was on the ECMA standards committee, and the course was being run for the benefit of that committee. It was my first introduction to the approach taken by that laboratory, which was I think primarily operational. But they were very appreciative. I actually spent the evenings during that conference writing the very first draft of the axiomatic approach paper on the notepaper of the Imperial Hotel in Vienna. [chuckles] I gave the manuscript to my colleagues in IBM and they were very appreciative of it, but I think very rightly decided that the method was not sufficiently mature shall we say to be applied immediately to PL/I.

CJ: What was your reaction to the large definition they were writing?

TH: Oh, withdrawal I think. Definitely I didn't regard, as it were, literary, suitable for literary reading.

CJ: [chuckles] Right. 1969 we've said the paper came out. I'd like to know what you feel the reaction was from the community, both short-term… I happened to be at the presentation you gave in Vienna for the WG 2.2 meeting in 1969. So did people immediately appreciate that the axiomatic approach was a good way forward? And we'll come to longer term in a minute.

TH: Right. I don't know that I was so worried about impact then as we are now. [chuckles] I think I was quite happy with the interest that people showed at these technical committee meetings.

CJ: Longer-term of course, this is one of your most-cited papers. I found 6,000 citations, more than 6,000 citations to that one paper.

TH: Oh.

CJ: Do you feel that that's an approach which is now widely followed?

TH: I think a lot of people do know about it, and it is recognised as one of the three methods of expressing the semantics of a programming language. And a lot of people who were perhaps more comfortable with the operational approach did feel the necessity of proving that it was consistent with the axiomatic approach in the sense that everything you could prove in one system would satisfy the properties that you could prove of the program in the other system.

CJ: So in working backwards, what I wanted to do was draw out some of the practical stimulus to your chosen research topics. In a paper, I guess it's the Turing Award speech, you talk about the connection between the bound checking that you built into your ALGOL compiler and the idea that they were a form of assertion. How much do you think that was an influence for you, that you…?

TH: Yes. I think I've always been attempting to make sure that the programmer had a control and understanding of what the computer was going to do when executing the program. So the motto was that whatever happened could be explained in terms of the programming language itself, and you didn't have to understand anything about the machine code or the structure of the computer in order to debug the program. I think that's really a very good principle. Which has not always been observed in subsequent languages, but the necessary condition for it is that the subscripts on all the array references must be checked every time. And indeed, modern languages are following that example, perhaps without ever having heard of it of course.

CJ: You're of course talking here about machines that were much slower. There was an overhead for checking those array bound-……that you were staying within array bounds. Your customers were prepared to pay that overhead?

TH: Maybe my customers didn't know. But since most of the customers were academics and had to use to the computer to teach students programming, I think they were quite glad of it. Many years later, the company offered the customers the option of building into the compiler an option for switching off the subscript or array checking, and they said "no." They knew how many errors were due to subscript errors.

CJ: Yes. We've not finished with the axiomatic method, but I would like to pick up on one thing which your name is always associated with, which is the Quicksort algorithm, and its connection to programming languages. So could you build the connection for us with your ability to write the program Quicksort down when you first had the idea?

TH: Not when I first had the idea. The idea first came to me when I got interested in sorting. I remember well thinking about it on my couch in my room at Moscow State University. The first idea I had for doing sorting was something like bubble sort, and then I thought it was a bit slow. I could calculate the… 'It would be n^2 in the length of the array, so there must be a faster way.' I did think explicitly, well, if I could start off by assuming that my array was split into two parts, and all the elements of one part were smaller than all of the elements in the other part, then I could tackle those two problems separately. And I sat down and used the only programming language I knew at the time, which was Mercury Autocode, and wrote the partition algorithm, the easy, non-recursive part. And then I was faced with the problem of how does one organise the calculations required to sort all the partitions that you've left behind to sort later? I couldn't figure that one out, but I thought there must be some way of doing it.

A year or two later when I was working for Elliotts, I came across the ALGOL 60 report and I read it. That was worth reading. People who have read it agree with me that it was. You learnt something about programming by reading that report. It had that wonderful sentence in it about recursion – 'Any other occurrence of the function designator inside the function body denotes a call of the function itself.' 'Recursion. Ah, that's the way.' I sort of described it and that led to publication in the Communications of the ACM of the algorithm in their algorithm section.

CJ: You describe sitting on the couch. We'll come back to Moscow in a while, but you describe sitting on the couch. Did you have pencil and paper? How were you thinking about sorting?

TH: I had pencil and paper, yes, to write the program. That was after I got the idea of course, and I don't think I ever bothered to even write out the bubble sort algorithm.

CJ: Is it true you had a financial wager about this algorithm?

TH: [laughs] When I came back to England, I was offered employment by a small British computer manufacturer, Elliott Brothers, and one of the first things that my boss gave me to do was to write a sorting algorithm. He showed me the algorithm that he wanted written. It was the now-called Shellsort, and it was quite complicated and very difficult to see how fast it was going to be. But when I'd written it out and delivered it back to my boss, I said, 'I think I know a faster way of doing that.' And he said, 'I bet you sixpence you don't.' Then I explained it to him and he implemented it for one of the Elliott machines and found indeed it was considerably faster even than his previous algorithm, which had been a merge sort.

CJ: For our audience, sixpence is how much money? [laughs]

TH: Well, about a halfpenny in present money.

CJ: [laughs] A very small wager. So we've got you at Elliotts. We've worked back to there. 1960 to 1968?

TH: That's right, yes.

CJ: After the sorting algorithm success, the next big success was the ALGOL compiler I think.

TH: Yes.

CJ: Could you say a bit about the project?

TH: It was a bit of a surprise. In those days, we wrote the programs that we wanted to write more or less with very little management instruction, and even less checking of deadlines or anything like that. I worked with Jill [nee Pym], my wife, and other members of a small team. And after about a year or so, I sort of thought maybe we could deliver it in another six months or so. So I told my boss that maybe we could deliver it, and he was quite pleased and he started selling it, and probably increased the sales of our computer quite a bit.

Oh, that was exciting. It's nice actually doing something that somebody finds useful, provided that they come back and tell you this. If you're a manufacturer however, you deliver this large chunk of paper tape embodying 10 man-years perhaps of intellectual effort, it's like publishing a book, you don't hear anything about it until much later.

CJ: So you've referred to the ALGOL description as a very valuable document. My recollection is it's a very short document as well.

TH: Indeed.

CJ: Which is even more impressive.

TH: It was about 26 pages of half-size book folio format.

CJ: But have I heard you also give credit to a course which I think was in Brighton?

TH: Yes.

CJ: Who were the instructors on that course and what was the content?

TH: The instructors were Edsger Dijkstra[4] and Peter Landin[5] and Peter Naur, Edsger and Peter of course winners of the Turing Award.

CJ: A pretty impressive team to get you up to speed on ALGOL 60.

TH: I remember not actually doing the exercise that Peter Landin had set, but writing Quicksort instead. Rather shyly I went up to the dais on which he was sitting and showed it to him. He looked at it for a bit and he looked at it again, and then he said, 'Peter, come over here.'

CJ: [laughs] Right. I'm sure they weren't grading you, but you would have got a good grade for that.

So this leads very naturally into the topic of programming languages, which is one of the things cited in the Turing Award. For those who've only programmed in high-level languages, could you describe what it was like to program for your machine, the Elliott…?

TH: 803 initially, although the main sales were on the 503, which was a faster machine which was built a little later. Programming in machine code was writing a lot of decimal and octal numbers on a piece of paper. [chuckles] What else can I say? The instruction code was relatively simple for that machine, and it was great fun to try and find the shortest sequence of instructions that would carry out my will on the computer with as short a time as possible.

CJ: How about design aids? So yes, you had to write this sequence of instructions, but did you use anything like flowcharts to develop the design?

TH: I didn't use flowcharts I don't think. There were flowchart templates that perhaps some people used. But I think on the whole the experience was that they were only used in cases that the management insisted on it. But not in my company they didn't. Our managers didn't do that.

CJ: We haven't mentioned one very important member of your team – Jill, now your wife, actually worked with you on the ALGOL project.

---

[4] Edsger Wybe Dijkstra (1930 – 2002) was a very influential Dutch computer scientist who made many contributions to both practical and theoretical aspects of the discipline.

[5] Peter John Landin (1930 - 2009) was a British computer scientist who made many important contributions to theoretical aspects of computer science. The final years of his career was spent at Queen Mary College, University of London. The computer science building there was named the Peter Landin Building in his honour.

TH: Indeed. She did nearly all the detailed programming of it. My duty was to write in ALGOL itself a sort of outline of the structure of the compiler as a whole, and I left nearly all the rest of the work to them.

CJ: So programming languages. The ALGOL 60 compiler while at Elliott, then a long series of other contributions to programming languages. Could you say a bit about ALGOL W and how that arose?

TH: Yes. In 1962 I think, I was invited to become a member of the ALGOL committee at IFIP WG 2.1. The committee spent some time working on revisions/corrections to the original ALGOL 60 report and produced a new report in 1962. Then they called for ideas to put into the next version of ALGOL, because in those days it was expected, like machine architectures, that languages would change every few years. So I made a number of language feature proposals, which were published in the ALGOL Bulletin, and that caused me to be invited. I was quite an active member at the Princeton meeting of WG 2.2, at which they discussed the features and gave to me and Niklaus Wirth the duty of writing up the agreements of the meeting in a format that would make it suitable as a definition of a new programming language.

CJ: And that did not become ALGOL 68.

TH: [laughs] Yes.

CJ: Are you prepared to tell the story about the schism and the transition from Working Group 2.1 to 2.3?

TH: Well, very briefly, the report was produced and presented at the next meeting. I think at Saint-Pierre it was, Saint-Pierre-de-Chartreuse. And the boss of the mathematical centre in Amsterdam, Aad van Wijngaarden whom you know well, during that period had discovered a new way of defining the syntax of a programming language which he wanted to try out on this new language. He spent some time explaining it. I thought it was unnecessarily complicated. But he persuaded the committee to give him a go and he was charged with producing the next draft, which he eventually did. It went through many revisions and culminated in the language ALGOL 68.

CJ: And you were not a fan of ALGOL 68.

TH: I'm afraid the final meeting in 1968 at which the committee discussed the draft and approved it, I was one of the signatories of a minority report, which in the words of Edsger Dijkstra was 'We have to regard, as a clear description of the methods of programming, that this report is a failure.' [laughs] He didn't mince his words.

CJ: And a number of you left or resigned from 2.1 and formed a new working group.

TH: Yes. I wasn't one of those who either resigned or formed a new working group. I wasn't a founding member of it. And I did stay on in the ALGOL committee to look after the interests of ALGOL 60 at a time when the committee was mainly concerned with removing – what do you call them? – ambiguities and something or other of ALGOL 68. When that task completed – it wasn't a very onerous task – that was when I resigned, and at the same time I was invited to join the WG 2.3 on programming methodology.

CJ: Yes. Also on programming languages, a very influential book, the *Structured Programming*[6] book. I fear structured programming was somewhat oversimplified by some people, but the content of that book has been very influential.

TH: Yes. The name 'structured programming' I think was taken from the people you're referring to, namely your own employers, IBM, intended to be equated with just avoiding gotos. But the book, I think we interpreted, the authors of the book interpreted it as applying much more to the overall architectural structure of a program rather than the details of the way in which a flowchart has been encoded in a linear programming language.

CJ: And a paper I love, 'Hints on Programming Language Design', which I think has also been very influential although perhaps should be even more widely read, that was for the first POPL conference I think, Principles of Programming Languages.

TH: I think it was, yes.

CJ: But it wasn't in the proceedings. Were you late delivering, or…?

TH: Oh. I don't know that proceedings were considered all that important in those days. I think it would have been late. I certainly had produced it within six months as a report of Stanford University, and that's presumably its ending, resting place.

CJ: That's the question I have, yes.

TH: Yes, yes.

CJ: And then another very big project in which I knew you were involved in early on was the Ada project from the US Department of Defense.

TH: Yes.

CJ: Could you say a bit about that?

TH: Well, I happened to be in the United States on sabbatical in the previous year I think

---

[6] *Structured Programming*: O.-J. Dahl, E.W. Dijkstra, C.A.R. Hoar; Academic Press, London, 1972.

it was, and I took on a consultancy with the Air Force to write a report on their new programming language, which was called JOVIAL, JOVIAL J-3. I wrote a report on its various features, which again I'm afraid wasn't very complimentary. [chuckles] But the report was of course ignored and so was the language. The Department of Defense decided to start work on a new language, which eventually became called Ada, and invited four teams to submit draft proposals for the language without laying down very many conditions about what the language should contain. And I was asked to serve as a consultant to one of the teams, the one that worked at… it was SRI at that time.

So I spent several trips to Menlo Park to advise them on the evolution of this language. Because like so many language designs, it starts small and evolves, and the taskmaster, the person who was masterminding the project as a whole, kept adding more features which his clients, who were of course the armed services, required in order to gain acceptance of the new language. But the SRI proposal was eventually rejected and the successful proposal still required quite a bit of development, so I served as a consultant on that as well.

CJ: You say there were not very detailed requirements on what had to be in the language, but linking back to axiomatic basis, there was one very interesting requirement on the specification of the language.

TH: I can't…

CJ: I believe I'm correct. I haven't gone back and looked this up. But I thought the iron-man requirements – have I got the right phrase? – said that any language had to be specified either in your axiomatic style or in the operational style.

TH: I don't recall that, I'm afraid. Certainly I don't think any of them were in the end. I don't think I was giving advice on how to draft an axiomatic language construction.

CJ: So back to Elliotts again, but I'd like to postpone the operating system work till when we talk about CCS later. Could you explain how you came to be working for a computing company? Because as we'll learn later on, your university degree wasn't… Well, there were no university degrees in computing then, but how did you get to your first job being at Elliotts?

TH: In 1960 when I came back from Moscow State University, just before I came back, my uncle, who was the general secretary of a British Scientific Instrument Manufacturers Association, he was organising an exhibition at which his manufacturers would exhibit their products. And he invited me to serve as interpreter to the exhibitors and promised to pay the princely fee of £40. [laughs] So I actually cut short a holiday and went to do the interpretation and found there was a computer being exhibited by Elliott Brothers, my subsequent employers. I spent most of my time actually on that stand, although I did do some other interpretation of lectures.

CJ: So perhaps you could explain to our audience how and why you knew Russian?

TH: When I finished my undergraduate degree, I got a job… sorry, I had to do national service. I applied therefore, partly based on a connection with my uncle who was a captain in the Royal Navy – in those days, these things apparently used to count – applied to join a course and learn Russian. They accepted me on the basis of my qualifications no doubt in Latin and Greek, and so I went up to Crail to study Russian in a military camp and later passed the examination to study it at the University of London, a branch of the School of Slavonic Studies.

CJ: So at that time, one was conscripted for two years to be in one of the services.

TH: Two years, that's right.

CJ: How much of your time did you actually spend in army uniform doing normal army things, and how much time did you spend learning Russian?

TH: Oh. Well, on every vacation – I think it was a month's vacation three times a year – we would spend two weeks in a camp and learn a bit of drill and learn a bit about seagoing perhaps, which maybe was just as well because part of our course in the end was to learn technical Russian to describe the parts of a ship. [chuckles]

CJ: Right. But I know from being in Saint Petersburg with you that you still speak fluent Russian.

TH: I used to go back to Russia fairly frequently to begin with to take Elliott computers to Moscow and exhibit them, and served on the stand as before to translate and generally to make things a bit easier for the exhibitors in a strange country with a strange language and so on.

CJ: So back with Elliott, initially your title was probably 'Programmer'?

TH: Yes.

CJ: And from there you progressed to…?

TH: Senior Programmer, Chief Programmer, Chief Engineer, and finally I moved out of the line of management and became a Senior Researcher I think.

CJ: How big was the research activity within Elliott at that time?

TH: Oh, it must have been quite small. Most of it was hardware research. But I met up with Mike Melliar-Smith, who was later the leader of the SRI submission for the Department of Defense language. He was my main colleague there and we were commissioned to design a new version… sorry, a new larger and faster version of a range computers which the company was manufacturing.

CJ: And then for our audience who have never heard of Elliotts, can you describe the series of takeovers that led to your departure from the company?

TH: Well, yes. The machine that we were designing never saw the light of day because the company was taken over in a very friendly way by the English Electric Company, and so I transferred my allegiance to the English Electric research group, who were working on a new design. And then English Electric were taken over by the ICL, which was a conglomerate of all the remaining computer companies in Britain.

I suppose I felt a bit sidelined, and I was offered… sorry, I was asked in the way that academics have whether I would allow my name to go forward for consideration for appointment as a chair in Manchester. I had received a similar offer in Oslo actually for the post that Dahl, also a Turing Award winner, eventually occupied. And it just tickled me because I'd always felt I wanted to be an academic, but I didn't know very much about the academic scene and I thought maybe a job with the government computer centre in Manchester would give me better contact with academic work in computing in Britain.

Was I right? No. [laughs]

CJ: [laughs] That's another issue. This is the so-called National Computing Centre…

TH: That's right.

CJ: …that was in Manchester. You didn't stay there very long though, I think.

TH: No. That was one of the more shameful episodes in my career.

CJ: No shame at all. You were offered a very…

TH: I think it was three months I was there, and half of it I spent under notice. I was the one who resigned because it occurred to me rather sensibly and rather late that maybe the best way of learning about the academic scene was to go for a few interviews for posts. So I rather tentatively drafted a letter of application and sort of wondered whether I would make it in time to catch the post. I thought, 'Well, if I can catch the post, I'll do it.' And I did. I went for an interview. To my intense surprise, I was chosen for the post.

CJ: Do you know what the competition was like at Queen's University Belfast? Were they interviewing many people or were you the only person considered good enough to be interviewed?

TH: No, they were interviewing several. In fact, I think I knew the two other… No, I knew one of the other applicants who was an academic… he was a member of the university already. No, I don't know that there was a great deal of competition.

CJ: So your first position in a university is as a full professor of…?

TH: Indeed, yes, yes. It's quite an experience coming in at the top as it were.

CJ: Can you describe the other transitions – what it was like to work in academic decision-making as opposed to working in the industrial environment?

TH: Yes. I was a bit shocked when one of the first things I had to do when I arrived in October was to decide something about the syllabuses for the next following year's courses. We never thought that far ahead in industry. The phases of industry were quite simple. At the beginning of the budgetary year, you expanded a bit, and at the end of the budgetary year, you contracted a bit, and that was as far ahead as one could possibly look. But that particular…

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 you have to understand which professors are more equal than the other ones.

CJ: [laughs] And the ways to influence decisions.

TH: Well, it was pretty unpleasant for the first two years actually because I was also director of the computing laboratory, which I took quite seriously. The manager of the computing laboratory and the professor of medical statistics, who was chairman of the computing services committee, attempted to dislodge me, which was really quite unpleasant. 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.' So I explained the problem. He said he looked into it and he came back with a right decision – I was not the director. That was a great…

CJ: A great relief.

TH: It was a great relief. And the unsuccessful applicant for the chair made a very good director after me.

CJ: You were I think in Belfast from 1968 to 1978.

TH: '77 I think.

CJ: '77, sorry. This was of course a time of troubles in Belfast, in Northern Ireland. Can you talk a bit about what effect that had on you personally and on the family?

TH: Well, 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. But 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 were so wonderful that we really enjoyed it. Our neighbours. We lived in a road a bit like Storey's Way with large houses and extremely friendly neighbours, still friends. And the only time that Jill was really worried was when – should I say this? – I was offered another post in London. Sorry, I was told that I had been appointed to another post and would I come and talk to the vice-chancellor about it? And 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.

CJ: Coming back to the axiomatic basis theme, while you were in Belfast, you wrote the FIND paper[7]. This brings neatly together your sorting thing and your axiomatic basis ideas.

TH: Yes.

CJ: That paper had an interesting history.

TH: Yes. I recounted that history at the POPL conference a little while ago, that I submitted it and had it refereed. Were you one of the referees? [chuckles]

CJ: Yes. Yes. [laughs]

TH: Being personal. Then I looked through it again to see how to put the referees' comments in and I couldn't understand it. Well, at least I was finding great difficulties in following the details, because I was trying to prove absence of overflow as well, and I thought, 'This doesn't present the use of the axiomatic method for proof in a very good light. So I'll simplify it. I'll leave out the problem of overflow.' So I rewrote it and resubmitted it and it was published all right.

One member of the audience at the POPL conference pointed out that I had been unscientific in retracting the paper merely because it was unattractive. The business of a scientist is to present it how it is. I should have kept it in. And it hadn't occurred to me that I had done any wrong and now I agree that I had.

CJ: That's an interesting insight.

TH: Yes, yes.

---

[7] Hoare, C.A.R., "Proof of a Program FIND," *Communications of the ACM*, Vol. 14, 1971, pp.39-45

CJ: To me, the transition was magic because axiomatic… the original 1969 paper is about proving programs. You made the transition in the FIND paper to a development method for programs, and that seems to me crucial for what has happened since.

TH: Yes, I suppose I did. Yes, thank you. I hadn't thought about it that way, at least not for a long time.

CJ: So I know personally you have a huge family of PhDs – children, grandchildren, great-grandchildren of your supervision. But in Belfast, you were supervising a PhD student, some PhD students without having had one [a Doctoral Degree] yourself. How did that feel?

TH: Oh, I don't think I felt the lack of it, no. I sort of feel and I still say that Quicksort was a good substitute for doing a PhD.

CJ: Tony, you next moved to Oxford. You were appointed to a chair at one of the most prestigious universities in the world. 1977. 1977?

TH: '77 is when I arrived, yes.

CJ: And we'll say later on you stayed until 1999. Before we move to the technical stuff, Oxford was your alma mater – we'll talk about that later on – but you went to Wolfson College when you went to Oxford. That's not a traditional college.

TH: That's true. So it's a graduate college, a fairly recent foundation. But as far as I was concerned, it was the right college for me because I was still somewhat in awe of the traditional colleges and the senior common rooms and so on. Wolfson was quite democratic and very friendly.

CJ: And some very interesting people there as well, people like Robin Gandy[8].

TH: Yes, indeed.

CJ: So one of the first things I'd like to pick up there is CSP, communicating…

TH: Sequential processes.

CJ: Thank you. [laughs] I didn't want to get it wrong. Perhaps again we could look at the context and switch back to Elliott. You're very frank about the operating system project at Elliott not being as successful as the ALGOL compiler.

TH: Indeed, yes.

[8] Robin Oliver Gandy (1919 –1995) was a British mathematician. Robin's PhD supervisor was Alan Turing at the University of Cambridge.

CJ: Could you say a bit more about that?

TH: Yes. We realised that the rudimentary operating systems that were available on our existing computers would not be adequate for use of a more expensive and powerful machine. So I took it upon myself I suppose – I was boss of the programming group then – to design an operating system, about which I knew nothing at all. So I read a few things, learnt about code words for example, what's now called virtual memory, and we tried our best to do something. But in the end, it turned out the system could not be delivered because it was too slow. It had used a virtual memory and caused everything to thrash. So the project was cancelled and nothing was delivered and the entire work of my department for the last two years was consigned to the bin, which was a bit depressing.

CJ: The machines at that time had tiny stores.

TH: Yes, and that machine that we had had a particularly tiny store of only 8,000 words, about four times that many bytes, and it had no capability for extending the main store beyond that limit, because that was the limit of addressing of the instruction code. Whereas other companies that got into the same trouble, including IBM I may say, were able to get around the difficulty by free gifts of hardware. We couldn't even give it away.

CJ: And this led to a long succession of contributions to how to organise concurrency, parallelism, and so on. Could you say a few words about monitors, for example?

TH: Yes. That was the result of a discovery of a way proving correctness of data representations. The monitor was just a representation of shared data, and otherwise had the same structure as an implementation of a data representation. That's I think what gave me the idea. Edsger Dijkstra was also very interested, because he had actually written a successful operating system for a computer of similar size and application in Amsterdam. So I organised in Belfast a meeting of people interested in operating systems, which led to the publication of a book called Operating System Techniques, and I wrote the introduction and one of the chapters.

We discussed… Per Brinch Hansen was there and he picked up on this idea that the updates to shared data should be all written and understood in a single place rather than being scattered around, which was the case in my previous proposal for conditional critical regions, which is also mentioned in the operating systems book. Per Brinch Hansen had the opportunity to publish the idea in the Communications of the ACM before I thought of doing so, and I'm afraid I wrote a follow-up of the same idea with very largely the same central content with a few details changed rather in the spirit of competition, I'm afraid. People for a number of a years were concerned about which of us had really invented it. Per knew exactly how it had come – we had both invented it – and he wrote a letter to me explaining exactly the order of communications and discussions that we'd had. But certainly the paper was I think somewhat influential and made me feel that I had really… that was the way I should have done it.

CJ: You mentioned Edsger Dijkstra in connection with the operating system. I was going to ask about guarded commands[9] and how much you feel the guarded command idea influenced the development of CSP as a language.

TH: Well, the guarded command itself was taken over directly, and I think it made… it turned out indeed when we formalised the semantics of CSP that was exactly the way to modularise the implicit conditional. I felt it was very important that if a process in parallel attempts to test whether an output is available for further input, it should do so with a command that at least carried the risk that the output would take place simultaneously, because I didn't want anybody testing the availability of something and then not using it when you found it was available. That seems to be a gratuitous way of introducing non-determinacy into the most critical part of a software system, which is of course the interfaces between the modules. I wanted the interfaces to be determinate, and any non-determinism should be expressed independently within the individual threads where we could manage it locally.

So [Edsger Dijkstra was] very influential I think. I got the syntax from him. I don't think I would have dared to make such a strange syntax if Edsger hadn't paved the way with his beautiful guarded command.

CJ: Well, I think you'd have dared most things, because we haven't come to the most radical departure in CSP, the complete abolition of shared state.

TH: Yes. This was at the time dictated by the structure of the implementing microprocessors, where the microprocessors were very cheap and fast but the sharing of memory between the microprocessors was expensive and slow. So one could get away with not sharing state because it fitted the architecture of the implementation - could be very fast.

The situation is somewhat reversed at the moment, as you understand, which makes shared memory more relevant. And I'm following developments and I hope I have something to contribute to the development of shared memory programming in the future. But I think the input/output will come back. People will realise the value of not sharing memory, particularly in the light of the security considerations, where shared memory is obviously offering a much broader front for attack by malicious software.

CJ: But it was still a radical departure. Did you hesitate? I mean you'd made your own

---

[9] Edsger Dijkstra introduced the concept of guarded commands as a way of making it easier to prove the correctness of programs, using Hoare logic, before the program is written in a usable programming language.

contributions to shared variable concurrency. Did you hesitate for long to say, 'There is no shared state between my processes'?

TH: No. [laughs]

CJ: [laughs] Good.

TH: That was the basis of the whole thing.

CJ: In connection with CSP, let's mention Bill Roscoe and Steve Brookes, who were two extremely important PhD students you had at that time. Can you describe the collaboration with Bill and Steve?

TH: I can describe aspects of it, I suppose. I had written a paper on CSP and published it in the Communications of the ACM, in the standard way/practice 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[10] in order to be able to redress the deficiency and make a formal model. I realised, but I was wanting to explore yet another method of defining the semantics of a programming language, which is the algebraic method. So I asked them to tell me what the algebra of this language was going to be, and they came back and said, 'Well, what do you want it to be?' [laughs] So that might have led to an impasse. But I think we realised, we must have realised that the way out of it was to do a denotational semantics of the language, and I worked with Bill Roscoe on that, and Steve was working on it too I think. Of course they've both made/done far more valuable contributions to CSP than I have now, and I'd like this opportunity of recognising that fact.

CJ: Seminal is important. Influential is another thing.

Let's move on to another major influence from CSP. There was the occam language and its realisation as the transputer, a physical chip. Can you talk a bit about how that came about?

TH: Yes. The founder of a startup company in Britain, Iann Barron, had read my paper, the first CSP paper, in the Communications, and he realised that he wanted to make a computer that would execute that programming language. For years, I'd been saying and Dijkstra had been saying that machines should be designed to implement the programming languages that make programming easy. And here was my opportunity and they offered me a consultancy, in which I was to advise on the development of the language and any hardware implications that I could think of.

CJ: And that led to a product which… I don't know how many transputers were built,

[10] Dana Scott is the recipient of the 1976 Turing Award

how many transputer chips were built, but it was a very large number.

TH: Well, I think until the ARM was produced, it was Britain's biggest-selling computer. And longest lasting. The actual architecture continued to be made for many years thereafter, and in the end it was selling something like two million a year. Which by present standards is very, very little, but by previous standards was… The computer that I spent most of my time working on for Elliotts, we only ever sold 200, and they were delivered at sort of once a month [actually once a week]. [chuckles]

CJ: And that led to one of the Queen's University industrial awards I believe.

TH: Yes. Well, that was the work done by Bill Roscoe actually in the formal verification of the hardware design for the floating-point unit. That was the first I think published case of an error detected in a hardware design. Fortunately for the company, it was detected before the chip was put into production. A much bigger company, as you know, Intel, a few years later came across a similar error after the computer had been delivered.

CJ: And cost them a great deal of money.

TH: Well, I think they put aside half a billion dollars, but I don't know that they actually spent them. A lot of people aren't terribly interested in correctness, you know. You've noticed, I think, yes. [chuckles]

CJ: Another major project from the Oxford time, which we've recently had a retrospective conference about, was the ProCoS project. Could you describe the vision of that project?

TH: The vision of the ProCoS project was set by our friends in Austin, Texas, the inventors of the ACL2 system and its predecessor. They had done a project to formally verify and to get a machine-checked verification for the correctness of the hardware and software for admittedly not an existing chip but a potentially viable chip design, which was successful. I wanted to reproduce that technology in Europe. So that was the initial inspiration, but I was most interested in the verification of the consistency of the various tools which they verified – the assembly language for the computer, the verification condition generator, as well as the hardware system and the operating system. I felt – wrongly I believe now – that the technology of Boyer and Moore's tool was not capable of doing structural proofs of that kind, so we did it all manually in the project and learnt a lot from it. But no particular deliverable product I say, except that the people who worked on it are still around and they're still contributing to the German verification efforts, at the time which more or less might otherwise have been rather diminished.

CJ: Yes. You corrected me. You corrected my omission. This was of course a European-wide project funded by the European Union with partners in Germany and…

TH: Denmark.

CJ: And Denmark, yes. Another line which began during the Oxford time was the Unifying Theories of Programming with your colleague or visitor He Jifeng. Would you like to say a few words about the objectives? I think we'll come back to it when we talk about Kleene algebras later on, but…

TH: The goal of Unifying Theories of course is one that I got from the current efforts by physicists to unify the theories of the four forces. I realised that there were more theories out in the published literature than any one person could comfortably read in a lifetime, and wanted therefore to find some way of unifying them in the scientific sense, that the unified theory would be a generalisation of the other theories but would not supersede them. One doesn't wish to create an antagonism that you're trying to supersede solutions which have been developed very often to deal with particular application areas and particular system architectures, and which are not invalidated by a general theory which shall we say is instantiated by no application and no architecture. Which is what we were looking for actually. [chuckles] It's nice to be a theoretician.

CJ: Could you say a few words about collaboration? People read your final papers and think these are such gems they must come uncut directly from your pen.

TH: No. [laughs]

CJ: I happen to know quite a few drafts.

TH: Well, I did confess to that in the Essays in Computer Science. Yes, I regard writing a specification or writing an article as the first test of a theoretical idea, that one needs to find a way of expressing it that sort of makes it seem inevitable, that there couldn't be a better way of describing this particular phenomenon, and so carry the reader with what might otherwise seem to be a series of arbitrary definitions through to the place where the punch line could be delivered. And I'm still doing it, I'm afraid.

CJ: So Oxford, major university. When you went there, the department was tiny.

TH: Yes. There was me and Joe Stoy, and two programmers.

CJ: And many practical problems. Can you talk about growing the MSc, moving the department from one building to another, and all of the things that you had to attend to as well as your research?

TH: I think you just about summarised it in the terms best appropriate. [chuckles] Yes. Setting up anything new at Oxford at that time was very difficult, and I was nearly all the time a member of the Faculty of Mathematics. I had been a member of the

Faculty of Science in Belfast and had learnt fairly quickly and exploited my knowledge of how to influence that committee to make a decision in my favour, and eventually learned how to do it pretty well so that I could predict what was going to be passed and really avoid wasting time on something that is not likely to actually pass muster.

When I got to Oxford, everything was turned on its head. In Belfast, one can make an argument based, for example, on the public perception. 'What would the public think if they knew that you were doing this sort of thing?' Or you could base it on the potential benefits for the application/exploitation of the research. These arguments carry 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. You know, that was fast enough. However, there was a predecessor. The Department of Material Science had had an even more spectacular rate of growth for a number of years and they knew how to do it, but they were in a different faculty – Natural Sciences, which was more used to this kind of thing. I was in the Faculty of Mathematics.

And then Mrs Thatcher – bless her for this at least – made an offer of money to found new posts. The first one was associated with the graduate course that we wanted to set up, and the next four were associated with an undergraduate course which I then wanted to set up, a joint degree course with mathematics. I was very pleased to be in a mathematics faculty because I knew that mathematical talent was the way to recruit good programmers, good computer scientists. And of course Bill Roscoe and Steve Brookes were a case in point. But then we got additional, slightly lesser numbers of outside money to support posts to set up new degrees, because no politician wants to support something that already exists, and therefore you need to set up a new degree if you wanted to expand.

So the number of new degrees I started in Oxford must… I don't know. The record probably still stands. Hope so, hope so. Because it's not really much fun.

CJ: And of course the college system, which is so valuable for undergraduates in Oxford, acted as a brake in the sense that you had to get the buy-in of all of the colleges.

TH: Yes. Every post that is offered by the university is a joint post, a joint appointment with a college, and the college, they're mostly fairly traditional colleges teaching fairly traditional subjects. And the only reason why the colleges were willing to accept a new subject was because Mrs Thatcher – bless her for this too – cut the funding of the universities and restricted the number of places universities were allowed to take, and each of the posts that were associated with the subsequent generosity had 10 college places associated with it. So it was just the right bribe to get the foot into the door. But there's no… [laughs] there are problems with dealing with colleges as well, as you know. Not with Wolfson but the undergraduate colleges.

CJ: And eventually 'retirement' – 'retirement' in quotes – came along from Oxford in 1999?

TH: That's right. I reached the standard age limit for retirement at the university at that time.

CJ: And we had a very nice conference to mark the end of your time in Oxford I remember. A lot of people might have stopped work at that time. You instead…

TH: I got an offer from the director of the research laboratory just being set up in Cambridge by Microsoft.

CJ: Cambridge, UK.

TH: Cambridge, UK. And the director, Roger Needham, offered me a post. He'd offered me a post two years previously, but I thought I was needed in Oxford at that time still. I think maybe I was wrong. My last two years weren't very productive after Jifeng left. So I took it. Well, I spent a half-year sabbatical up in Cambridge to test the waters and brought Jill with me of course, because she would have to agree. We both liked the place. And when I heard from the founder of the Microsoft Research Laboratory, the principles under which the laboratory was founded were to employ the best people and give them their heads, let them do the research that they felt was important. The only thing that he did require was that the recruits should have fire in their belly and want to change the world. Maybe I did.

CJ: So can you describe how you saw Microsoft? You'd been in industry in the UK early on. You now joined the largest software company in the world. Did you feel it was ripe for exploiting more formal methods? Did you feel that the methods they were using were adequate? I'm thinking of a famous paper of yours.

TH: [laughs] Well, when I wrote the axiomatic method paper, I thought that the topic of verification of programs using the axiomatic method would not be of interest to industry for a number of years. And during the time it is not of interest to the industry, it was appropriate for academic research, because industry was obviously going to have far much more money than a university to pursue the research, and therefore the sensible academic will withdraw if the industry's looking after the field. I wanted to see whether that prediction was correct. And indeed it was. Microsoft was not using formal methods, not for several years. But when they came to use it from necessity, not for the reasons that I had myself predicted – it was that in the end some error would cause loss of human life perhaps – but because of the virus, which I'd never predicted, nor had Microsoft. So they turned to an element of formal methods, the analysis of programs, as a method of countering the threat of the virus. I believe that human evolution was driven in much the same way, actually.

CJ: You've already hinted at this, but would you like to say a bit more about the research ethos, the ease of getting people with fire in their belly issuing their own ideas in an environment like an industrial research lab, versus in universities as you last worked in them or even as you know them today?

TH: Well, the thing that worked well in the universities is that the universities were able to collect teams to undertake projects which were larger than a single theorist could match. And this worked very well, very well indeed. People did pull together and produce and demonstrate ideas to the development organisation in Microsoft, many of which found their way into Microsoft products. And that sort of prospect of eventual delivery was what motivated the research and motivated the collaboration. University research is much more fragmented because the university's going to have a very small team working in any particular area of research, and the needs of teaching require that even those are diversified. Therefore most collaborations in universities at the level of staffing that we then enjoyed were between universities, which is quite an overhead.

Building teams of theorists is actually very much more difficult than teams of engineers. Much more competitive. There are no agreed criteria as to how you judge between two theories if all that you're producing is theories. You need some form of experimental use of a theory in order to make that choice, and the project that makes a theory useable, that is a tool that enables ordinary programmers to take advantage of the theories, is a multi man-year project and takes many, perhaps 15 years even to mature after the originators have put in a lot of work on it. It doesn't really recruit a productive and reactive user base for up to 15 years. So you have to be very brave to embark on a project like that.

CJ: Well, bravery's never been lacking. Can we come right up to date on your own research? And I don't expect in this interview to go through the full detail of Kleene algebras, but could you build the connections between what you are trying to do now with the algebraic approach, what you were trying to do in Unifying Theories, and what you were trying to do in axiomatic basis?

TH: Well, yes. Starting with the axiomatic basis, the first part of the axiomatic basis used an algebraic approach to illustrate how you could axiomatise a branch of arithmetic, and you could give different axiomatisations to different kinds of arithmetic, which at that time were an option even in the hardware of the computer. You could tune your axioms to describe exactly the kinds of binary arithmetic and sign plus modulus arithmetic that were fashionable at that time. And if I'd maintained that tradition, which I got by looking at standard algebra books in mathematics, I would come about with the idea of presenting the axioms as equations in an algebraic form rather than as proof rules in the form of Hoare triples.

It was only a whisker's breadth as it were. I just did not get the right idea at the right time. Even when I was writing the book on Unifying Theories, what I was doing was constructing a model of the theories using Dana Scott's method, the denotational semantics, to cover a great number of theories of how programs worked. It was again one of those chance discoveries lying on a sofa that led me to believe that one could actually present an adequate treatment, a usable treatment of the meaning of a programming language in a few algebraic axioms, which are almost identical with those that apply not just to programs but to numbers as well. Simple laws of associativity, commutativity, and distribution were exactly what you need in order to reason about programs and ensure their correctness. And I discovered a very simple proof in which I defined my own triple – or, sorry, the Hoare triple, it's not really my own – in terms of the algebraic operation of sequential composition, and derived the proof rules from the algebraic axioms by a perfectly standard style of logical justification.

So that was a surprise and I've been talking about it ever since. [chuckles]

CJ: But each of those earlier steps that you're now somewhat critical of spun off enormous amounts of other work. I can't help wondering if you'd started with Kleene algebras if any us would have understood it.

TH: [laughs] Quite. And the Kleene algebra, actually the advance was triggered by a discovery that I could do this for a new form of logic, logic of programs, a new definition of the triple that appeared recently as a result of the work of Peter O'Hearn called separation logic. I was looking at the proof rules which express the semantics of separation logic in terms of Hoare triples, and I discovered the law which enables me to treat concurrency in the same way as sequential composition. And that I think was really not only unification of theories but unification of two ideas which are now central to computing, concurrency and sequentiality, into a simple algebraic framework. And since then I've discovered that Robin Milner's operational semantics could be similarly defined in terms of the algebra of the semicolon operator, and all of his laws, his laws of operational semantics, could be derived from the algebra as well. So yes, very satisfactory. [chuckles]

CJ: And still busy?

TH: Ah, yes. Well, I'm trimming the hedges a bit and trying to go back to a denotational semantics, which is really based on the needs of people who are debugging their programs. A person who's debugging a program needs to see a comprehensible trace of the behaviour of that program together with an indication of where the fault has been detected, and with the ability to trace back in the program to all the places which might have to be changed in order to get rid of that fault. So one has a sort of graphical picture of arrows and chains of arrows leading back from a symptom to the causes to help you discover and diagnose and correct the error.

So just as the Hoare triples were designed to help people to prove programs and the Milner similar rules, the operational rules are designed to help people who are compiling and implementing the programs. My new denotational semantics based on graphs is an attempt to provide the theory which is directly applicable to the testing and correction of programs. So I'm trying to bring that particular branch of programming methodology under theoretical control as well.

CJ: I'd like to change gear. Some of our audience I'm sure would like to know more about Tony Hoare the person. You weren't actually born in the UK.

TH: I was born in Ceylon, now called Sri Lanka, in Colombo. My father was a British civil servant, among the rulers of the country. And my mother was the daughter of a tea planter, which doesn't mean somebody who plants tea but somebody who looks after a tea estate and looks after people who plant tea and collect it and dry it and manufacture it.

CJ: Do you remember things about Ceylon as it then was?

TH: I remember a few things. I went back there when I was 70. I took my family back on a holiday trip. And there are one or two things that I remember. Not as many I might have. It was mostly fairly…

CJ: I actually meant do you remember things about living there when you were a child or…?

TH: Oh yes. I remember going to school there, and the incidents going into the jungle to see elephants and tigers… sorry, leopards, and bears and buffalo. All of them pretty dangerous. The headmaster of the school took us on a school party to Yala where we stayed in the rest house and went around in this old bus to waterholes to see animals we could see. Fascinating.

CJ: And you then had to move away, still not back to the UK immediately.

TH: After… This is… We… My mother and my two brothers moved to Rhodesia during the war because of the threat of imminent invasion of Ceylon, and we spent a couple of years in Rhodesia and South Africa before going back. The school that I'm talking about was in the rather brief interval between returning to Ceylon and returning to Britain, 'returning' of course in two different senses. All the English in Ceylon regarded 'returning' as being returning to the United Kingdom.

CJ: And your first school back in the UK was…?

TH: Was the Dragon School in Oxford, a rather superior prep school where I spent just under two years. Got a scholarship to a public school in Canterbury, King's School.

CJ: Which leads on to your first university degree, which wasn't an obvious preparation for computing. Could you explain what the degree of 'Greats' is?

TH: Yes. It has quite an ancient tradition in Oxford. It consists of four subjects. Latin and Greek language and literature – well, that's four already – Latin and Greek history, and ancient and modern philosophy. So it's a four-year course with an exam in the middle, in which I did moderately well, but not sufficiently well to gain a research grant to do doctoral research in philosophy at Oxford, which is what I would otherwise have done. Fortunately…

CJ: That might have saved computing. [laughs]

TH: I think it saved me from possibly rather a career for which I was not ideally fitted.

CJ: What made you choose Greats?

TH: Well, at the public schools in those days, all the brightest students studied Latin and Greek, and history was for those who can't, and scientists, well, nobody knows what they take up for a subject. [chuckles] So I was always interested in mathematics. I got quite good marks in mathematics for as long as I was studying it, and I went on to study mathematics just for the fun of it from popular textbooks. And I acquired an interest in philosophy through the philosophy of mathematics, through reading books by Bertrand Russell for example and C. E. M. Joad, who was quite a popular philosopher in those days. And certainly it was the study of philosophy and particularly the philosophy of mathematics and the foundations of mathematics that led me into computing, take an interest in computing.

CJ: You were at Merton College I think.

TH: Merton College.

CJ: Presumably that's a very traditional college.

TH: Very traditional. It claims to be the oldest. I'm there because my father was there. [laughs]

CJ: But presumably offered you lots of scope to pursue your interest in philosophy and logic and so on. It wasn't a tightly constrained course?

TH: Well, the course was a fairly massive course, as all university courses seem to be after secondary school course. But we all had personal tutors, and the personal tutor would advise us, set us an essay subject every week in philosophy or ancient history, and so we went out to look at the literature, which he also recommended. No, I don't… I mean I studied logic in my spare time, but we did have spare time for goodness' sake. I studied it from Quine's book on mathematical logic.

CJ: And around this time, you met your first computer. The Mercury I think. Was that while you were an undergraduate, or was that in the master's course that followed?

TH: That was in the master's course. I attended a course run by Leslie Fox, who was my later head of department when I came back to Oxford as a professor.

CJ: And that was a course in statistics, not in programming as such?

TH: After my national service where I learnt Russian, I thought I better do something a little bit more practical. So I registered for a course at the Unit of Biometry just to get a diploma in statistics, a one-year course, and managed to persuade them that I knew enough mathematics to stand the pace. That enabled me… Well, I very much enjoyed that. I mean statistics is still something that I find interesting, and it's getting more interesting for computer scientists too.

CJ: Then there's the machine translation connection. Could you knit that into the story for me?

TH: Machine translation was a bit of a flash in the pan. When I was in Moscow, I got a letter from the National Physical Laboratory at Teddington offering me a post as a senior scientist to work in a team of programmers who were attempting to program an automatic translation from Russian to English on the Pilot… – no, not the Pilot ACE – the ACE computer at the National, which was, if you remember, a very primitive computer. So I took up an interest in the subject and I studied it in Russia, more or less neglecting my statistical studies, which I should have perhaps paid attention to, but were a bit beyond me. And that was how I got interested in sorting.

CJ: Yes, I was going to make sure we got that link. So large dictionaries of words needed sorting, yes?

TH: Yes, because the dictionaries were held on magnetic tape, and if the words were sorted before you started the magnetic tape whirring, then you could pick up all the words in a sentence on a single pass of the tape, which might very well take 20 minutes. And the… So how did I get… Sorry, what was the question again?

CJ: Well, just the link between machine translation and your eventual Quicksort algorithm, the design….

TH: Oh right. You were angling for that story then.

CJ: So we've already mention Jill, Jill Pym before she married you. You were married in 1982. 1962.

TH: Thank you. [laughs]

CJ: [laughs]

TH: Yes, January '62.

CJ: Children? Grandchildren?

TH: Yes, we have three children. Tom first. He's now a security expert working in the research facility of Huawei in Banbury, Oxford. My daughter Joanna is married…

Sorry, her partner is a city architect in Vienna, and she lives in Vienna and learned German, and is now working as an organiser for the Buddhist community in Europe. And my youngest son was Matthew, was a bright schoolboy, but he unfortunately succumbed to leukaemia some time ago. In, well, 1982. He was very clever, amusing, bright, an extraordinarily kind and considerate person. Real, real fun to be with. And he left us with many happy memories.

CJ: You've lived in houses, I gathered earlier, more than one in Barnet.

TH: Yes. That's North London.

CJ: North London, yes. Of course Belfast, which we have talked about. Then you lived in Oxford.

TH: Yes.

CJ: And now here. Actually ignoring for the moment the spells in the States, not too many moves in your life.

TH: No, no. Eight years for industry, nine years in Belfast, 22 years in Oxford. Wow. [laughs] I keep remembering that this is twice as long as Mrs Thatcher was Prime Minister, and that was too long.

CJ: [laughs]

TH: And now 16 years working for Microsoft in the research department.

CJ: Yes. And there were spells in America at least.

TH: Yes. The first one was six months where I was hosted by Don Knuth and wrote a number of papers, and met the Palo Alto Research Center of Xerox, which was the leading, really leading computer science laboratory in America at the time. And then a year in Austin, Texas with Edsger Dijkstra, which was wonderful.

CJ: The famous Year of Programming.

TH: The Year of Programming, yes. We organised a series of seminars which we called the Year of Programming. And I'm hoping to go back there next year to renew acquaintance and celebrate the retirement of a close friend and colleague.

CJ: Well, to move towards wrapping up, as well as the Turing Award in 1980, the enormously prestigious Kyoto Prize in the year 2000, honorary doctorates. Can you remember the first and the most recent perhaps?

TH: Yes, yes. The first was at the University of Southern California, and it was organised by Per Brinch Hansen, who was good friend of mine. He was a great man.

And the most recent were in Europe – Warsaw, Madrid, and Saint Petersburg.

CJ: And at least 10 in between those, so…

TH: Well, nearly perhaps. I don't know. [laughs]

CJ: …a lot of honorary doctorates. Fellow of the Royal Society, Fellow of the Royal Academy of Engineering, a knighthood in the year 2000. That was a good year.

TH: Yes, it was a good year. [laughs] That was my first year at Cambridge. So I met the President of China, the Mikado of Japan, and the Queen, all in the same year.

CJ: So it was actually the Queen who conferred the knighthood on you?

TH: Indeed it was, yes.

CJ: Many collaborations along the way, and in many cases those collaborations have established that person's main scientific thrust. Do you work best in collaboration do you feel?

TH: I haven't made… I work a lot by myself now. I think I do enjoy being… Well, I need somebody else to keep me on the rails. [chuckles] Niklaus Wirth filled that role for some time, He Jifeng for a very long time. Admittedly they do a quite a lot of the hard lifting and I'm very grateful to them.

CJ: Well, Tony, thank you very much. It's been a very interesting discussion and I'm sure our audience will enjoy hearing something about the way you do research and about you as a person.

TH: I hope so, but it's been very much a pleasure to meet you again and answer your questions again. Thank you.

[end of recording]