An Interview with Roger Bishop Jones

Tony Dale

Started 2005-01-26
Last Change Date: 2012/06/03 21:32:25
http://www.rbjones.com/rbjpub/www/papers/p006.pdf
Draft Id: p006.tex,v 1.12 2012/06/03 21:32:25 rbj Exp

Contents

Chapter 1

Introduction

The following interview was never intended for publication[1], but was undertaken as part of a research project directed by Professor Donald MacKenzie of the University of Edinburgh[2].

[1]The editing of the transcript has been done by me. In doing this I have made modest changes to make it easier to read and to correct definite errors in the transcription, removing only the most conspicuously superfluous material. These include, removal of multiple "I mean"s, substitution of "yes" for "yeah"! I have also added footnotes, sometimes to explain what I *really* meant at the time or to give a bit more relevant information, sometimes to give my present view.

[2]and was later cited in his book "Mechanising Proof"[Mac01]

Chapter 2

The Interview

Interview with Roger Jones, ICL, 8.3.94.
Interviewer: Tony Dale

2.1 Academic Background

I: So, first of all, can you tell me about your own academic background from university onwards, please?

RJ: Sure. Now, I've got a rather peculiar university thing, because I went to Cambridge from school to do engineering, but very rapidly decided that I wasn't interested in engineering. So that was long before Cambridge did computer science for undergraduates, and in fact, there was only... I think there was one university in the country that did computer science at the time.

I: What year was that? What year are we talking about?

RJ: That was 1966.

I: '66, right.

RJ: And I left at the end of the first year, and took a job with English Electric. And I worked in the computer industry for five years, and then I went back to university at Keele.

I: So, just a minute: you left Cambridge after one year, went into the computer industry. Where?

RJ: English Electric. Nelson Research Labs at Stafford.

I: And what were you doing with computers, then?

RJ: We were doing research into compilers, and compiler compilers. That was one thread. And we were also doing work on software for analysis of power transmission systems.

I: All right, so you were doing... you were working... you were actually producing software?

RJ: Yes. Well, that was my first job, and I transferred to ICL[1]after about eighteen months, something like that, and initially continued that work[2], and did more work with compilers after that, and worked with compilers for producing compilers[3], mainly, until I went back to university in 1972.

I: And you went back to university to do... ?

RJ: Well, I intended to read maths and education. I intended to do some teaching. But I went to Keele which had a four-year degree course with a foundation year first, and I was going to do a teaching certificate concurrently. But during the first year I changed my mind about what I wanted to do - which was one of the advantages of going to Keele - and so I did maths and philosophy, joint maths and philosophy.

I: Oh right, gosh. Who taught you the philosophy there?

RJ: Well, the professor was Swinburne.

I: Oh, yes, he's a colleague - ex-colleague - of mine.

RJ: And the logician was Treherne[4].

I: Treherne. I don't know him.

RJ: Well, he's still at Keele. He's transferred. He was at Edinburgh as well, actually. He was at Edinburgh before. He's been at Edinburgh; he's been at Imperial College; he's been in the philosophy department at Keele, and the last I heard he was in the computer science department at Keele. And I think he did... well, he did computer science at Keele and at Edinburgh, I think. So

[1]International Computers Limited, now absorbed into Fujitsu

[2]the work on power transmission systems analysis

[3]I wasn't working on Compiler Compilers at ICL, so I don't know what this means!

[4]Alan Treherne, still at Keele as of 2010.

he's got a similar... he's one of these people who hovers around between maths and philosophy and computer science.

I: I know the sort. And did he teach you standard first-order logic at that point?

RJ: I don't remember whether he taught the logic courses earlier on in the course, but my special subject in the final year was philosophy and mathematics, and he did that. Philosophy: a foundation course, or something like that.

I: And the maths you did was what, pure maths, or applied?

RJ: Well, a mixture.

I: Oh, you have a mixture at Keele, do you?

RJ: Yes, a mixture of pure and applied.

I: Right. So you were given a fair spread of mathematics, and presumably a fair spread of philosophy too?

RJ: Yes, though only half a degree's worth of either, so...

I: Was it concentrated more or less on the logic/philosophy of science end in the philosophy?

RJ: No, I don't think so, no. It was the standard philosophy course they do at Keele, most of which students are doing other subjects, like generally Arts subjects rather than Science subjects. I don't know what the mix is, but at the time everybody at Keele did two subjects[5], and so... I mean, I guess I've forgotten most of the stuff that I don't really use that was done there, so I don't remember all the stuff we were taught in the philosophy.[6]

I: It was quite a change from computers, which you had been doing previous years. Why did you give up the computers?

RJ: I didn't think I was getting where I wanted to be. I wanted to change, at the time.

I: And after your degree, you didn't go back into teaching, as you'd thought, but...

RJ: Well, I knew I wasn't going to do that after the first year there. I changed my mind about teaching rather rapidly.

[5]Two *principle* subjects leading to joint honours, after a foundation year of very general coverage.

[6]I specialised in mathematics in the sixth form partly because I was good at it, but even more because I have an appalling memory and wanted to avoid Chemistry. Its a miracle that I ended up with a good joint honours degree with Philosophy.

I: A wise decision!

RJ: Yes. I went to Warwick.

I: To do what?

RJ: To do a PhD.

I: To do?

RJ: I went to the computer science department at Warwick, which had - and probably still does have, though I don't hear much about them nowadays - had a strong theoretical group under David Park. The research topic that I identified and sought a suitable university for was basically about mechanisation of mathematics[7].

I: The mechanisation of maths. Had you been thinking about this topic, then, during your degree, or... ?

RJ: Yes.

2.2 Logicism and Formal Proof

I: What had led you to think about that? Was it the previous experience in the computer industry?

RJ: Well, it was a combination of the fact that I came from a computing background prior to going to university, and the fact that I was interested in the foundations of mathematics and logicism, and things like that, so it just seemed... it just seemed the right way to do maths.

I: From a logical buildup of step-by-step... you mentioned logicism itself. You were talking about a particular philosophy of maths. That particular philosophy appealed to you?

RJ: Oh yes, certainly. I'm a dyed-in-the-wool logicist to this day.

I: Are you?

RJ: Yes.

I: You actually believe that mathematics is nought but logic?

RJ: Well, I guess that's an extremely ambiguous phrase. Of course mathematicians do lots of things other than logic, but I think it's an important characteristic of mathematics that it is, effectively, reducible to logic in an appropriate sense.

[7] See Chapter A.

I: Right. The appropriate sense being?

RJ: Well, I think the key issue, I think the key thing is, that it's all... it is all a priori, and in practice you can derive all of... or you can derive all of mathematics in a suitable foundation system into a set of... I mean, we have to strengthen it when you want to take your mathematics a bit further, but in practical terms you can... not only do you have very hard criteria for truth and falsity of mathematical conjectures, but they're mechanisable, and you can build tools which will basically tell you whether you're right or not in your claims about the truth, not in the sense that they will tell you whether an arbitrary conjecture is true or false, but if you claimed to have proven something, then that's a testable claim, a mechanically-testable claim.[8]

I: So these are proof checkers you're talking about now, that you can actually run... put it in a suitable form for... ?

RJ: You're always talking in the realm of hybrids if you're talking in the realm of practicality. No practical system is purely a proof checker. You've got something that's just a proof checker, then it's not going to... you're not going to see its problems. On the other hand, you know, you're not going to get the tool to do all the proofs for you for the kind of things you want to prove. It's got to be a collaboration. Well, in terms of actually verifying that the proof is done... the thing you end up constructing together is a good proof, that's completely... that can be completely recognisable. I guess, thinking back, when I went to university I was... I've always had a problem with the notion of mathematical proof - what counts as proof and what doesn't count as a proof - and I've never felt very inclined to actually get a good understanding of what mathematicians call a proof.

I: Why?

RJ: Because I knew, from the point of view of formal logic, what a proof was. And it seemed to me that that was a viable... in the days of... in the era that I was living, you know, where in prin-

[8] As you can see from the floundering here, I was not expecting quite such tricky questions in this interview. What I should have said then, was "in the sense that you can do most mathematics with a tool like ProofPower". What I would say today is simply that is obvious (pace Quine, Boolos, ...) that mathematics is analytic (which was my belief even at that time).

ciple you could have tools which would assist you in constructing completely formal proofs and checking them, I wasn't... I really wasn't terribly interested in learning the new answers. Well, I was also rather infuriated by it, because lecturers play games with you, you know? They will go through one proof and jump huge steps to each stage, and then all of a sudden they'll start turning round and berating you, and say... and asking you to prove the obvious. And, you know, I find that a... If I weren't already annoyed by the vagueness of what constitutes a mathematics proof, I would still have been utterly infuriated by the fact that they'd changed the rules of the game at every turn.

I: These were your lecturers at Keele?

RJ: Yes, though I should say that I didn't go to very many lectures, so I'm not a very good source of information about how good lecturers are, since I think the idea... I mean, I was also diametrically opposed to the idea that you can teach mathematics in lectures, since what has turned out to be the practice is generally that the lecturer goes along and spends an hour writing his notes on the board while you spend an hour copying them down onto a piece of paper. That seems to me to be a complete waste of time. So I didn't actually go to many maths lectures at Keele.

I: Right. And yet you had this view that somehow they were cheating because they skipped important bits and would sometimes...

RJ: Not cheating, because I don't think they had a hard and fast notion of proof there against which they could be claimed to be cheating, and clearly they don't. It's a sociological thing. What's acceptable is a mathematical proof for them. But it need not be.

I: You had a formal conception of proof at this time, did you, a sequence of steps which follows by rule from the previous member or members of the sequence: I mean, that was the notion you were operating with?

RJ: Yes, I was at the time. In my first year at Keele I was working through by hand proofs of conjectures from propositional logic. I spent a certain amount of time working through in a different logical system the elementary results in Principia Mathematica, and so I definitely at that time knew what a real formal proof was.

I: That was an axiomatic proof you were using, or natural deduction?

RJ: Yes.

I: Axiomatic? You actually had your axiom starting, and you came down through hardware doing proofs?

RJ: Yes, yes. If you try and interest me in the proof of a propositional conjecture these days and I'm just not interested. I don't want to see them. A special proof tool doesn't show... tell you anything about those things. And I did do... when I was still in the realm at the beginning of the maths course when you could do these things, I did much more formal proofs than they would have done, or expected normally. But you don't get very far before that turns out to be impractical.

I: Was that because of the logic you were studying in the philosophy at the time led you to this notion of 'proof'?

RJ: No.

I: This was independently thought up?

RJ: I don't think the logic you do in the early part of the philosophy course very significant, it was prior studies of my own.

I: Right. So how did you get this notion early on?

RJ: Well, I guess by reading, reading books.[9]

I: You were reading? You were reading outside the course? People like Russell, or formalists, perhaps. Did you read any formalists? Hilbert or Carnap?

10

[9]I would add now, that it wasn't particularly "early on", since I was 24 when I went to Keele. My Logicism wasn't precocious, I was just late doing my degree.

[10]The preceding section of the interview is very muddled. Because I was a mature student I had various ideas and attitudes (and even competencies) which were unusual for a first year undergraduate. Alan is here probing what those ideas were and how I came by them. The answers are unclear so I will try a better explanation here.

Between my year at Cambridge and my arrival at Keele I had five years of work in the computer industry and a fair amount of exploration in my own time of diverse topics. My knowledge of and acquaintance with logic began very soon after leaving Cambridge. Even at Cambridge I had come across Turing, the sight of his name written on my jeans provoked the senior tutor at Churchill into reminiscences about Turing's bicycle and the unique skills required to ride it (this is the only reason I remember having known about Turing at

RJ: I've not actually read much Hilbert, no. I think most of my knowledge of formalism is second-hand.[11]

I: From introductory guides to the philosophy of maths, or Körner[12], or... ?

RJ: That sort of thing, and what people claim when they're writing philosophical papers, which is usually, even for somebody with only second-hand knowledge, rather doubtful.

I: OK. So you ... moved to Warwick, and you had this notion in mind of mechanising mathematical proofs along the lines of how a formal system would work? Is that correct?

RJ: Yes. I was certainly always dissatisfied with the notion of proof that was used by mathematicians.[13]... I don't have any recollection of ever thinking it practicable to do proofs, formal proofs, manually....

I: How optimistic were you at that stage that you could in fact mechanise mathematical proofs?

RJ: It was obvious you could. It had already been done anyway.

I: What were you aware of at the time? Were you aware?

RJ: I don't know, actually. Probably at that stage... at the

---

that time). Soon after leaving Cambridge, my acquaintance with logic and computability began with a texts by Jeffreys [Jef67] and Minsky [Min67] both published and acquired in 67/68, but I believe that my knowledge of logic at this point was quite superficial. A few of years later (1971) I read Russell's "Introduction to Mathematical Philosophy" [Rus19], which was probably my first exposure to Logicism. I had read Russell's autobiography[**?**], probably in 1968, and sometime later his "History of Western Philosophy" (probably between 1969 and 1972). In the year before going to Keele I was using the company library to retrieve classic papers by Kleene, Turing, Church which I didn't much understand, around this time (not before 1972) I also read Ayer's "Language Truth and Logic"[Aye71]. This is all small beer, I am not a scholarly or prolific reader, but I did start as an undergraduate with an unusual prejudice in favour of formal proof.

[11] And I was a logicist not a formalist. Also I did not read Carnap before going to Keele, and probably at Keele the only bit of Carnap I read was "Empiricism, Semantics and Ontology"[Car50]. It was not until much later that I discovered how much of Carnap's work was devoted to formal methods.

[12] I think he is referring here to Stephan Körner's book "The Philosophy of Mathematics" [Kör60], which probably was the first general account of the Philosophy of Mathematics which I read. However, I don't know when I read this, whether it was before I went to Keele or not.

[13] as opposed to the notion of formal proof studied by mathematical logicians

beginning of the degree there I probably wasn't aware of any mechanical proof systems that were available. But it was obvious you could. ... in principle. You could certainly code up proof checkers.

I: Right, right. Proof checkers I can see, but proof finders would be much more difficult, possibly.

RJ: Well, ... if you start from the viewpoint of Principia Mathematica where you're talking about it in fact being possible to do something pretty formal, albeit at a large amount of labour, it's natural to presume that if you've got a computer to help you, that you can achieve more. And I guess until you get further into it, you don't realise how difficult and complicated it is. Even having looked at Principia, for example, which is pretty complicated.

I: The Law of Excluded Middle comes out as 500 lines, or something, I remember.[14]

The Law of Excluded Middle's quite clearly a proof, isn't it, in the Principia system. So I mean, to actually ape such a system you virtually require a heuristics of kind, ...

RJ: ... I don't generally think that using computers to ape other systems is a good scheme. You generally find that there are better ways of doing things.

I: Can we just come back to ... what you were doing at Warwick, because we're obviously straying onto ...

RJ: OK. So ... I looked for a university. ... I had some short description ... I couldn't dig it out[15]- which was basically mechanisation of mathematics. But it's basically about trying to get together the kind of tools you needed to do mathematics properly - what I thought was properly - but mathematicians to this day still write... I ended up at Warwick to do that, but unfortunately I ran aground at Warwick for purely personal reasons: I got rather entangled in my relationships in a way which prevented me from doing any good intellectual work, and I left Warwick after the first year, and I figured I wasn't going to get any research done, so I

[14]In fact it is proposition *2.1, which appears about half way down the third page of the formal proofs. Scanning back through what is essential to get there from the axioms it looks like the proof is about 10 steps. On the other hand, the entire paperback volume "Principia Mathematica to *56"[RW70] only gets as far as the ordinal 2 in 400 pages, so there is no doubt that it is heavy going.

[15]I have dug it out, and it is in Section A.1 (page 47).

might as well get a job and do something, and losing the game. So that changed the course of my career somewhat. And it took me completely away from the area for many years.

I: So you went out and got a job independently of computing?

RJ: ... No ... I went back to computing. I rejoined ICL in 1977 after spending a year getting a bit of background, a bit more background, but not actually making any real progress in terms of research. And at that time, of course, there were proof tools around. Automath had been around for quite a while, and LCF.

I: You knew of LCF and Automath at that time, did you?

RJ: Yes, but I didn't know a lot about them.

I: Right. And how did you come across those? Did you come across them at Warwick, or just in general reading?

RJ: Yes.

I: At Warwick.[16]

RJ: Because it was a direct development of my intended research. So it's part of the background...

2.3 ICL, Formal Methods

I then spent about five years working on something completely different, which was microcoded emulation of mainframes, making 2900s[17] behave like 1900s[18] And then I got a bit fidgety with that, so I decided that I wanted to get back into doing something which was relevant to my interest in logic, and I started moving around within the company trying to find the right place for that.

I: They would allow you to do that, would they?

RJ: Well, if you could find the opportunity. ... you've got to find the jobs. So I did a bit of knowledge engineering, and I did a bit of working on relational database technology. And then I was invited to join the Defence Technology Centre, where they were forming a formal methods group. So that was the beginning of 1986. I finally found a job where you really did use logic.

I: And this was... ?

[16]Reading at Warwick, I don't think I tried them out.

[17]ICL's "New Range" of mainframe computers.

[18]One of ICL's old ranges.

RJ: This was in the formal methods group. In 1985 they set up...

(Interruption)

I: You were just going to tell where you would need logic in...

RJ: When ICL was formed - just a bit of history - ICL was engineered by the government out of lots of computer companies. At the time, in order to give it a good chance of succeeding, there was an arrangement whereby ICL was barred from entering defence, and the people who were in the defence computing business were barred from entering commercial computing. So that was a period to give us a good chance to get going in commercial computing where we were protected from the bits of the UK computer industry that hadn't been put into ICL, but we weren't allowed to do defence work.

...[19]

There was thought to be a commercial opportunity to make a vast profit in defence. We[20]set up this Defence Technology Centre, and one of the things they were doing was R&D contracts with GCHQ in the area of secure systems, and GCHQ had a requirement for formal methods to be used on these contracts. And so a formal methods group was started up.

I: At ICL?

2.4 Z and HOL

RJ: Yes. I joined the unit at the beginning of 1986, and this was to use formal methods in the development of secure systems. It was Roger Stokes who founded that unit in 1985, and he had this little motto or guiding principle that he wanted us to be engaged in using real tools on real problems, and ... the real problem we were ostensibly interested in was proving that certain systems were secure. So we did take an interest in improved tools from the beginning, and we started to use them as soon as we had the opportunity

[19]What I should have said here is that this period of agreed constraint came to an end, probably on the 20th anniversary of the formation of ICL, and so ICL began to gear up to enter defence markets.

[20]ICL.

to do so. When I arrived, Roger had already been looking at the Cambridge HOL system, and the Boyer-Moore system. GCHQ, or CESG, had already, on advice from other consultants, decided that they wanted to go with the Z language, Z specification language. So we knew that what it looked like was that we should be aiming to be able to do proofs in Z.

I: Proofs in Z, right, yes. [21]

RJ: But there weren't any tools for that, ... and we didn't want to sit around and twiddle our thumbs until Z proof tools came along.

I: Presumably from PRG.

RJ: Well, in fact from us[22], ... we decided we were going to choose the most appropriate tool and if necessary transcribe from one language to another in order to able to do some proof work.

I: Which languages are you going to and from? Sorry, Z and what?[23]

RJ: Well, at the time we were looking at those tools with a view to which tools would be most likely to be helpful in reasoning about specifications that were originally written in Z.

I: Right. Were you going to try doing the whole thing within Z, or come outside of it?

RJ: Well, we couldn't expect to do that without us having a Z proof tool, so it was a question of what was the best compromise we could achieve at the time, and we didn't have a lot of time to go around surveying all the possible tools. In any case it's enormously expensive to get to understand a proof tool properly anyway.

[21] At that time (maybe still today) many people in formal methods thought that to reason about a formal specification, you needed something else, a *logic*. Any formal language in which you can express propositions can be made into a formal logic, by codifying sound deductive rules for that language. Cambridge HOL illustrates this point the other way round really, it is "Higher Order Logic", but by some pragmatic enrichment of syntax the logic can be made into a usable specification language, so you end up with a specification language which *is* a logic, not a specification language *and* a logic. The Z specification language was rather like HOL in being rooted in a deductive system anyway (Zermelo's axioms for set theory), though in the case of Z the distance was too large for the deductive system to be obvious.

[22] The Oxford Programming Research Group (PRG) were at that time advocates of paper and pencil.

[23] The answer here is HOL, but it comes out rather tortuously.

I: To understand... ?

RJ: A proof tool.

I: To understand a proof tool.

RJ: Yes. And if you've got a nice big research grant, then you can afford to do a thorough study, but even in those circumstances people very often do shallow studies and [24] come up with all the right conclusions. So we did... we made a judgement in a fairly rapid way on the basis of a fairly modest amount of experience, and we went for the Cambridge HOL system. The two main reasons for doing that were, first of all, the actual logic supported by it was about the closest you could get to Z.[25]It's a higher-order type theory which is logically not very dissimilar to a typed set theory, which is what Z is. And also, because it was implemented using the LCF paradigm it was fairly customisable, so you could have extra layers of functionality on the top, and they were [good for] orienting towards doing things for Z. We spent several years doing work with Cambridge HOL, a couple of years maybe, before we did... and we operated at that time on the basis of supplementing Cambridge HOL with a superstructure which helped us to do proofs about specifications we did. And that was a kind of semi-manual, semi-automatic process.

I: You'd have to put some input into the machine occasionally to direct it?

RJ: Well, in terms of the actual transcription of specifications, that was the first hurdle, to get them into a form which the tool would accept. And that was essentially a manual process, but the superstructure that we had on the HOL system made it possible for us to write HOL specs in ways which looked closest to the original Z. ... And there were certain technical problems that this superstructure solved for us as well. There's a big cultural difference between what happens in the HOL community and certainly what was happening then in the Z community, where the Z community made a virtue of pencil and paper, and freedom with the language, and where the style was a rather freewheeling axiomatic style you didn't worry about consistency very much.

[24] don't

[25] In a proof tool available at that time.

I: This was before the advent of W, or whatever logic they designed for it?

RJ: Yes. It was before there was a published semantics for Z, you know, before there was much by way of definition of the language. It was sort of taught by example.[26]

I: No Spivey reference manual.

RJ: No. But Spivey's reference manual was published in '89.

I: And the semantics about the same time, I guess.

2.5 Conservative Extension

RJ: The semantics was published about a year earlier. That was his PhD thesis, in '88. So... and the HOL community was oriented around formal proof and culturally indisposed to non-conservative extensions. It was sort of frowned-upon to use axioms, and so...

I: Why was that? Why was it frowned on?

RJ: Well, I guess basically because it was unsafe. If you just ran the axioms around[27], then you'll end up reasoning in an incoherent context, and then that makes the whole thing limitless. But I don't think... I mean, that's the objective reason for it. I'm sure to some extent it's just a cultural accident, because it's not where they came from, and they got into that cultural attitude.[28]Nevertheless, I think the thing that wasn't accidental was that they were into formal proof and rigour as opposed to more freewheeling approaches to formality.[29]I think one of the contributions we've made is to show how you can integrate these two cultures together and get

[26]It wasn't quite that bad. There was some documentation which came out of an Alvey collaboration involving the PRG and SD.

[27]this doesn't sound right to me, so it probably is a transcription error, but I can't guess what I actually said.

[28]I suspect that Russell, *Principia Mathematica* and logicism have a lot to do with this. HOL is the closest modern descendant of the Theory of Types used by Russell and Whitehead in *Principia Mathematica*, and the idea that you adopt a logical foundation system and then work in it using only definitions has a lot of attraction. So at a time when logicism was supposed by philosophers to have been discredited, its practical consequences were being taken up (by some) at the theoretical end of Computing.

[29]There is a middle ground in which specifications are formal, but proofs are informal (and selective).

something good out of the combination. And I think that... the most tangible evidence of that is the reconciliation of these two different views about whether you use axioms or whether you don't use axioms. [30]

I: How do you reconcile those two seemingly opposite points of view?

RJ: Well, I guess you don't quite reconcile the two, ... what you can reconcile is the rather freewheeling axiomatic style of Z ... with conservative extension. And to do that, you need to have some slightly different principles of conservative extension, and you have to accept the discipline of conservative extension. ... The scheme is that when you do axiomatic extensions in Z, you're normally defining constants axiomatically. ... Historically, there was no constraint on what you wrote there[31], and not even much tradition of mentioning that these might be problematic[32]. Now, you can do pretty much exactly the same thing subject to the constraint that the extension you make is conservative.[33] And you can do that mechanically with the HOL system, which we originally did using the choice functions, which, however, doesn't give you quite the right semantics. That is to say, you can behind the scenes treat an axiomatic definition as saying this new object you're introducing is *the*[34]object having a defining property, which means that it would have the property if and only if the property is consistent. So it's in practice, provided that you're prepared to accept the discipline

[30]The idea of "reconciliation of cultures" is suggestive of exerting significant influence on two groups of academics in Oxford and Cambridge. I hope that is not what I intended to claim here. There are some quite simple technical ideas which make it possible to work in a way which is compatible with both of these cultures, i.e. to reason by conservative extension in HOL about specifications written in a fairly free-wheeling axiomatic style of Z, and we used these ideas and implemented software support for them. Its moot whether this exerted any influence on the academics.

[31]Apart from syntactic well-formedness and type-correctness!

[32]The axiom might be self-contradictory, or be inconsistent with the rest of the specification.

[33]Does not permit the proof of any new theorems other than those mentioning constants introduced by the extension, and hence not compromising the logical consistency of the specification.

[34]I should have said "an", an *indefinite* description, i.e. Hilbert's choice function.

of conservative extension, or even if you're prepared to accept the slightly weaker discipline that you use conservative extension as a norm, and you very explicitly depart from it where you think that's justified.

I: Can you say that again? I hope we missed it. [35]

RJ: And you depart from that rather explicitly where you think that's justified, on a rather more limited number of cases than... or you can make it hang together. I think this not very well-understood, and not very well-publicised. But it has got embodied both in the culture of proof for Z, where largely through our influence, I would claim - not only in our proof tool for Z, but also the one that CESG have funded - recognises that consistency proofs are important elements of how you develop specifications. [36]

It's also been reflected in the Cambridge HOL system, which has had a new form of conservative extension subsequent to our work in this area, which enables you more flexibility in the form of your definitions.

I: This was something that pre-existed your work?

RJ: No. Historically, the sequence is something like this: that we took the Cambridge HOL system in, and we started using it for Z. And we did that originally by mapping axiomatic definitions of Z down to ordinary definitions in HOL using the choice function, and that's slightly unsatisfactory. It differs from the originally intended scenario in two ways. One is that it requires conservative extension, which is... you know, limits you to conservative extension, which was an intended limitation; and there's a non-intended limitation which arises from using the choice function, which is that any two constants which are introduced using the same property are provable equal.

I: Are they?

RJ: So if you say, "A is the[37] object having property P," and then you say, "B is the[38] object having property P," you can prove

[35] I think he may have been concerned that it didn't get on the tape for some reason.

[36] Quite possibly we did exert a little influence on what CESG asked for in the Z proof tool they funded, but probably not in any wider context.

[37] an

[38] an

that A equals B even though there may be more than one object having property P.[39] And so what you wanted was a definitional principle which was looser.

I: So you could distinguish A and B, even if they had the same property?

RJ: Well, not that you could distinguish them, but that you couldn't prove that they were equal. And so we did a... we did actually do a slight modification to the Cambridge HOL system for the purposes of one of the contracts that we engaged in, and the changes we did there were first of all to seal off the loopholes which are present in the Cambridge HOL system, but you can't just do `mk_thm`, and make a theorem, and to make it so that... and we'd cut out the new axiom thing from it. And we also modified it so that after you'd introduced one of these definitions using the choice function that you could derive the less definite principle from it. You just searched that the object you've introduced has the required properties subject to consistency. Then you delete the original definition, which is the thing that allowed you to prove the equalities you don't want, and that threw away the extra strength. And at the same time we talked to Michael, and we suggested that they added a new principle.

I: What it might say?

RJ: Well, they did it. They did something which was equivalent to what it was. It wasn't exactly the same as we were asked for, but it... that principle is called '`new_specification`'. [40]

I: It's called?

RJ: '`new_specification`'. We only used to have definitional principles, where... I suppose the difference between a definition and a conservative extension is that a definition is usually more definite. It says, "This object is this thing," rather than "Anything satisfying this property." They're now having a thing called

[39] The inference would have been OK if it had been "the" rather than "an"

[40] I have it from Rob Arthan that my memory here is not quite right. We implemented "delete_definition" in our own version Cambridge HOL (for the OWR project) and suggested it for incorporation into the Cambridge system. There were some objections raised and "we" ended up writing a specification for *new_specification* which did prove acceptable. I suspect that the "we" was mainly Rob, since I don't remember much about it.

`new_specification`, which, subject to a proof that a property is consistent, will introduce as a conservative extension, the action which claims that a new constant has that quality. And that's... that makes quite a bit of difference to how you write specifications in HOL. I mean, it allows you to... (This is a bit hot!) describe differences to your specifications.

I: This is the work you were doing here at ICL, altering HOL to your... ?

2.6 Loopholes

RJ: Well, we did only minor modifications to HOL ourselves, and that was just in that specific area. Normally we were just sticking things on top because we didn't think it was... we didn't want to be changing the underlying HOL system. But this particular contract we were on was very high assurance, so we wanted to have the strongest confidence in the integrity of the proof tool we were using. So we sealed off the loopholes, and...

I: What loopholes were they? You mentioned loopholes before. What loopholes?

RJ: Well, there were two basic mechanisms, one of which is... just allows you to bypass all the checks, which was a... It just allowed you to make a theorem with no justification.

I: With no justification? You could have... well, it would announce, "This is a theorem," with no justification with behind it?

RJ: Yes.

I: This would seem to be a major flaw to me. What... ?

RJ: Well, I mean, it's only a flaw... it depends on your culture, it depends what you think the proof tool's doing for you. I think academics operate in an environment in which they're not pathologically suspicious of each other, and they're quite prepared to take some body's word when they come up with a machine checked proof that they haven't taken any short-cuts using `mk_thm`. We were operating in a pathologically[41] suspicious environment where you're required to prove that your system does what it's supposed

[41] *pathologically* is a bit strong there, *extremely* would have been better!

to do and where they pay people to check your claim. As well as having to check your claim they have evaluators who are there to make sure that you've really done it the way you're supposed to have done it, where you're not doing things on trust. So the tool's playing an important role there in this environment that it doesn't really play in an academic environment, which is it is sort of putting its hand on its heart and saying, "Yes, that guy really did do his proof properly." You don't have to take his word for it, you can take my word for it.

I: And so the constraints you're working on are much greater because of this extra pressure?

RJ: Well, I think the requirement in terms of assurance is...

I: I mean, I'm still worried about this notion of... in the original system you could actually just come up with anything you like as a theorem. And that I don't understand properly. How is this possible?

RJ: Well, I'd suggest you... I mean, it's certainly possible. There's no question about whether it's possible. The interesting question is why they're happy to leave it like that, and I suggest you ask that to them.

I: I wish I'd spoken to you first!

RJ: You've already been there, have you? Right.

I: I don't recall this ever being mentioned by them.

RJ: Well, I don't... yes. They don't think it's a problem, so they wouldn't have mentioned it. I mean, I don't think it's a problem for the kind of thing they're doing, but I think it is a problem... It would be a problem for us if we could not claim that it was impossible to prove this result other than by legitimate means.

I: I can see your point.

RJ: ... they also had `new_axiom` which is less problematic, for the simple reason that you have exactly the same power - you get a theorem out of it - but the new axiom is then stored in the theory hierarchy, and it's better, you can easily discover all the axioms in it, so there's no possibility of fraud with *new_axiom*. It's recorded as...

I: A new axiom.

RJ: There is still a risk, and the risk is that you've introduced

an axiom which is false, and therefore you would be able to prove anything. That's only the same as doing a slightly incomplete proof, where there are certain conjectures you haven't proved.

I: Certain conjectures that... ?

RJ: That you hadn't proved. "I've done the main body of the proof, and there are these other things we haven't proved, but they're uncontroversial, so we're not going to bother." You could be wrong.

[at this point there seems to be a gap or jump in the transcript, the conversation appears to be continuing now from a point after I have made some claim about being successful in "selling proofs" (i.e. making commercial business out of doing formal proofs)]

I: ... successful, then you...

RJ: Originally in getting... in selling proofs, really. And I think that was largely because we took the initiative and encouraged the customer. We didn't need much encouragement, you know, but I think at that time had we not been keen to do it ourselves, then it would probably have been several years longer before they would have to do.

I: This was due to your own work, your team's work at ICL?

RJ: And we produced... I think the most significant thing that we did in the early stages, the relatively early stages there, was the development of a hardware device which was certified and is still in the book as the most highly-assured secure system that's been developed.

I: And that was using your new theorem prover, or what?

RJ: Well, that was using Cambridge HOL with one or two small modifications.

I: With the modifications.

RJ: And a certain amount of... and a fair bit of additional superstructure.

I: Right. With the loopholes closed by that time? So you closed the loopholes of being able to produce any theorem.

RJ: We closed the loopholes for that project. We produced a special version for that.

I: And that still remains the most secure?

RJ: That ... is the only one registered at UKL6 ... are you familiar with ITSEC?

I: No.

RJ: Well, there are these various regulations about evaluating secure systems which are graded in terms of the level of assurance, and you have to do different things to get the higher ratings. And 6 is at the top of the scale. [42]

I: Does that mean 'absolutely secure', or does it mean... ?

RJ: No, it just means that's the highest[43] level of assurance.

I: So it could still be insecure, no matter how small one chance is, it still could be... ?

RJ: Oh yes. Insofar as the behaviour of any physical device is concerned, it may break.

I: Right. So that's the constraint put upon us by the physical world. It means things can go wrong.

RJ: But even if that [wasn't a] constraint, the models aren't perfectly precise anyway. Not for hardware. You can, in principle, get perfect models of bits of software that you do.

I: But hardware's different, because you've got to model the hardware, and that model is going to differ from the reality?

RJ: Yes.

I: Right. OK, so how far have we got, then? We've got... we're nearly up-to-date, I guess, aren't we?

RJ: Oh, let me see. No, not really.

I: Still not?

RJ: I think round about the middle of '88 we were given the opportunity to do some of these part-funded collaborative R&D things, and it seemed natural to us to do some work on the proof technology.

I: Some proof on... ?

RJ: Some work on the proof technology, because at the time there [was] no sign that anybody else was going to do any worthwhile work on Z proof technology. And it wasn't at all clear whether it was possible to produce a good proof tool for Z. At that time there wasn't any definition of [a] logic for Z, and it was clearly a

[42] ITSEC

[43] recognised

very strange language, relative to almost anything else that anybody had ever built a theorem prover for.

I: In what way was it strange? Tell me.

RJ: Well, for example, you choose something that's fairly fundamental logically. It's the only language that I know of where you can't tell what the free and bound variables in an expression are by looking at the expression. So you can introduce a whole set of variables by putting the name of the schema in there, and you have to go off and look at the definition of the schema to find out what variables you've introduced.

I: Right. So you'd inspect inside the schema, you mean?

RJ: You'd have to look inside the definition of certain objects to tell what the available schema is ... it's not very usual even for programming, ... I certainly don't know of any other logics, logical languages, in which that's the case. And it is very fundamental. Even with much simpler variable-binding behaviour, very eminent logicians have made many mistakes about the rules.

I: Indeed. It's a very tricky problem, it seems, both within logic and within the mechanisation that comes up all the time, the mistakes you could actually make with free and bound variables.

RJ: So anyway, we had these opportunities, and put together a collaboration between ourselves and Program Validation and the University of Cambridge, which was a fairly loose confederation. ... Our part in that was the development of this proof tool, which we now call ProofPower, ... which is basically a re-engineered Cambridge LCF-based HOL proof tool, with some Z built onto that.

I: Is it being used? I mean, apart from you, I mean, is it being used elsewhere for Z?

RJ: It's being used elsewhere.

(interruptions)

I: I was asking whether Proof Power was being used for Z elsewhere, outside of your own work.

RJ: Yes, well, there are other people who've got it, and we occasionally get feedback on what the're doing with it.

I: Are these commercial concerns, or academic?

RJ: Both. But I think the main use is still on our applications for our customers.

I: Applications for... ?

RJ: For our customers.

I: Your customers, right. You use them internally for your customers, and then say, "Right, we've run this through the ProofPower, and you're sure that this is OK?" I mean, is that what you'd do?

RJ: Yes. We'll deliver them the proof, in a machine re-runnable form, ... so they can run it all again if they want to, or do further development. We spent about three years on that project. It took eighteen months to get the project in place, and we spent three years on the project, and that finished at the end of 1992. During 1993 we took the thing through to being on general release as a commercial product.

I: Which one can now buy?

RJ: Yes.

I: And that brings us right up to date, does it? That's it?

RJ: Yes, more or less. Last year was our first year as a real business, in the sense that previous to last year we'd been earning external revenues in general, and also working on part-funded projects, but there'd been no requirement that we actually cover our costs or make a profit. It was fairly relaxed. But last year, we were required to start making profits, which we did, and so from there on in, we're pretty much like a business in our right.

I: Within ICL itself, a subdivision of it? And you're going to be selling more ProofPower?

RJ: Well, I hope so. The real problem is strategically, to find the ways of sustaining the investments for the funding, on the technology base. And I think we had drifted away from Cambridge quite a bit as a result partly of the fact that we're using our tool instead of their tool now, and partly the fact that this co-operation has finished, and I think we need to really vitalise our contacts.

I: And presumably, I mean, you must have differed from quite a lot if you'd patched up all these loopholes and they hadn't.

RJ: The loopholes are tightening. That's not a significant... but we had diverged insofar as we did... we have completely re-engineered the tools. So it's quite a bit different, though the basic underlying logic is the same. The basic technology is still the LCF

paradigm, so it's...

2.7 Choice of Technology

I: Tell me again why you chose LCF from all the possible routes you could have chosen?

RJ: Well, it's powerful, and it's open-ended.

I: It's the flexibility of it to build in your own... ?

RJ: Nobody knows how to build the proof tool they want yet. If you go into any application area, you're going to want to do new kinds of proof automation pertinent to the kind of theories or the kind of techniques you're using. And the LCF paradigm provides you with a safe way of extending the capabilities and a safe and powerful way of extending the capabilities of the proof system. . And, you know, I believe that eventually it will be accepted as the only real way of doing it.

(Interruption)

I: So did you seriously consider any other approaches when you were on your way to getting into Proof Power?

RJ: Well, originally, we looked at Boyer-Moore.

I: And why did you reject that?

RJ: Well, the two main factors are, first of all, the logic that Boyer-Moore uses is right at the other end of the spectrum. You know, it's a quantifier-free logic. So it was much less suitable for reasoning about things that originally came from Z. In terms of the logic it seemed less plausible that it was going to work out. And it hasn't got... you know, it's not the LCF paradigm, so it didn't appear to us to be as flexible and extendable. Those were its two main factors. Also, we have a cultural aversity to large numbers of brackets.

I: Yes, I can understand that.

RJ: Those were the only ones we looked at.[44] And I think we did look at NuPrl[45] as well, but I think that's the main... the

[44] By that I should have meant, the only tools we evaluated hands-on.

[45] NuPrl (a proof development tool for a constructive type theory) was one of many other tools which we read about, but we didn't actually try it out. I'm surprised that I mentioned it since I don't think we could have seriously

main differentiation between the HOL and the New Pearl is that New Pearl is constructive, and we weren't interested in constructive mathematics.

I: Tell me why not. Tell me why you're not interested in constructive mathematics.

RJ: Well, I'm tempted to ask you, "Why should I be?"

I: Well, it's all right.

RJ: I think it's a red herring myself. I mean, when doing mathematical models, you know, constructive mathematics is just making life difficult for you. I mean, the touted... the claim to fame of constructive mathematical systems is, you know, "We must be interested in constructive things, because we're doing computers," and therefore you want things that are going to be computable. Getting things that are computable is not a problem. Any program is computable. If you want to verify program, the last thing you have to prove is that it's computable. So that isn't a problem. Getting proofs cheap is a problem, and so anything that makes that more difficult is not a good idea.

I: Yes. Getting proofs cheaply and efficiently and quickly, less machine power and time, et cetera?

RJ: Unless you . The fact that...

I: Yes, I was going to ask you about that - the amount of human input that has to go into Proof Power - what... how much is guided by the human controller in a proof?

RJ: Well, it varies on the domain, on the kind of thing you're trying to do. It is at the HOL end of the spectrum where you're expected to have... it's intended for a professional who knows what he's doing, and that you get a higher level and more intimate degree of interaction than you would with something like Boyer-Moore. I'm not saying that Boyer-Moore isn't intended for the professionals, but I think as a research project it was aimed for high levels of automation, and not aimed at, necessarily, achieving the most cost-effective combination of human and machine input. And to some extent, it's just an accident of history how far we got in various different areas in terms of the automation, so there is linear

considered using a tool based on constructive type theory for reasoning about Z specifications.

arithmetic on natural numbers in the system, but there isn't linear arithmetic on integers. And it's only a matter of time and money before this pushes through. But there's an awful long way to go before have a really serious mathematical proof tool. Fortunately, most of computer science doesn't need you to do analysis.

I: No, no real numbers.

RJ: But, you know, that's where we have to be going.

I: I mean, there's no chance of any purely automated theorem prover?

RJ: Well, it depends what you'd expect it to be doing. It's all down to what you expect it to be doing. If you expect it to be doing... I mean, human beings don't do maths in a completely automated way, so why should we expect machines to do it? I mean, they can do all sorts of proofs automatically that would be hard for us to do, and harder for us to do for it, but mathematics is a creative activity, so you can't expect to automate mathematics. And when it comes down to automation, if you ask, "How do mathematicians learn to do mathematical proofs?" they learn off other mathematicians. You know, they don't just sit there and... they don't go into a new area and get the axioms of group theory and figure out how to group theory from the axioms, they learn techniques. If you want to make any progress in mathematics, you'd better learn everything that the guys in the field have done before, and all the techniques, and then see if you can make some movement beyond that. And it's bloody tough work, you know? So you can't expect the machines to be more successful than that, except in, you know, sub-domains where it's down to hard work and not creative input.

I: But couldn't we put the techniques into the machine?

RJ: Sure, well, that's what it going to take. You're talking about... you know, if you'd solved the machine-learning problem, then maybe you could get the machine to learn in the same way the mathematicians have. However, people haven't done that.

I: The AI venture, right.

RJ: Meanwhile, what we're basically doing and haven't yet got very far with is manually coding up the techniques. And that works. You know, you can do it. It's just expensive, and there's

an awful lot of them. And you can see that much more clearly, say, in the symbolic maths tools rather than the proof tools, where they've got much further in coding up symbolic techniques. And you can get them to do all sorts of things for you.

I: Such as? Can you give me some examples?

RJ: Well, differentiation, those sorts of things. I'm not as familiar with those tools as I'd like to be, but they've done an awful lot. And all that stuff should be within the capabilities of proof tools, and a lot more. But it's just a huge amount of work to do that. It's more work to do it properly with a proof tool. Of course, you get a different kind of capability out at the end because you get something that can actually reason, and doesn't get the thing wrong too often. And it can help you over the problems that a symbolic maths tool wouldn't be able to help you with. And as far as I can see, that's a lot of hard graft. If you approach it in that way, if you don't say, "We're going to solve the machine-learning problem in order to get these machines to learn how to do proofs," you say, "We're just going to do it the hard graft way. We're going to code up all the techniques," then you can get a lot more . And then you might, at the end of that process, have a better clue about how to the machine .

I: There was... Lenat actually claimed he'd got a program which could discover not only proofs but mathematical concepts. Do you know that work, ?

RJ: I think I've seen that.

I: Some years ago. That was '77, I think, that paper.

RJ: Yes, I'm not saying that you can't do these things on a small scale, but relatively speaking it's plausible that's it going to be more difficult to get a machine automatically to learn difficult things than it is to code up the techniques. I'm pretty sure that there's nobody out there who says he's got a machine which has learnt how to do linear arithmetic. And we've got linear arithmetic on the machine because we all went off to a textbook, and we learnt how to do the linear arithmetic ourselves, and then we coded it up. And that's bound to be a lot easier than trying to get a machine to do that.

2.8 Choice of Logic

I: So you're in favour of classical logic, quite obviously, from what you've said. So you don't see any point in having a theorem prover employing some other... ?

RJ: Well, there may well be a point, but I don't think it's particularly relevant to our field.

I: What do you want? Do you want to... ?

RJ: And I think the claims that it's important in computer science, I think, are overstated. And these claims were made by... I guess Martin-Löff's probably the guy. I thought it was rather entertaining that the Computer Science fraternity should take his word for what they needed.

I: What do you think the explanation of that is? He seems to be very influential. I mean, Martin-Löff seems to have been incredibly influential.

RJ: Well, the whole field is very academically-oriented, and in that context it's very understandable. His theories were, at the time, very interesting, complicated, and there was a rationale for their relevance to computer science. And so they're a really beautiful place to do research, you know? I mean, if you're not interested in the applications but are interested in doing research, then it's a goldmine. And one has to say also that many of the ideas do feed through and are potentially practically valuable. But the kind of ideas that are more plausible, from a practical viewpoint, are the different kinds of type structure, not the constructiveness. So dependent types are important things. Not that Martin-Löff invented them[46], but he certainly popularised [them]. And you can use those in a classical context, though even there it's still not clear whether they're better... whether you're better off having a complicated type system, a complicated undecidable type system of dependent types, or a simple decidable type system. It's still not clear. Many people think that these complex type theories are logically more expressive than simple type theories.

I: Logically more expressive. So you could actually put more

[46]Similar constructors are used in a type-free context in the earlier work on Combinatory Logic by Haskell Curry and his associates.

into them than you can if you restrict yourself to the decidable set?

RJ: Well, that's an interesting question. What they mean when they think they're more expressive... I can't think of any sense in which there is that relationship hard and fast. Certainly it isn't the case in terms of proof-theoretic strength, which is the logician's notion of expressiveness, and it's not the case either in terms of convenience of notation. It's not a sustainable case, that that's the only way to get good notation. By and large, it's still the case that these notations are more difficult to understand than plain set theory, and certainly no more expressive than plain set theory. [47]

I: So there's no advantage to using it, as far as I can see, from your description? It just complicates matters.

RJ: I don't see any advantages in the kind of stuff that we do. I don't say that this is not a good area for research that has been done at . I think it is an important area, but I think it's still not clear what the practical...

I: Yes. From a practical point of view, you can't see any benefits yet?

RJ: I suppose... there has been feed-through in rather... I mean, for example, the dependent type stuff fed through into programming language stuff about structure, like Pebble, which has got dependent types into it, and that in turn was influential, to a degree, on the design of the module facilities in standard ML, and so one can see via very tortuous routes that these ideas have input in things which are practically useful, and standard ML is certainly practically useful. We use that on lots of good things, but it's not necessarily in the ways one might imagine. And certainly in terms of directly using constructive type theories, that doesn't look to us to be practical. Mind you, we don't have a lot of choice anyway.

[47] Its important in understanding the significance of dependent types to bear in mind that these are used by Martin-Löff as part of a logical system which is based on the "propositions as types" paradigm. In this case the burdon of logical expressiveness in the system falls on the type constructors. They therefore have to be relatively complex, and the extra expressiveness delivered in this context by dependent types is essential.

By contrast, in a classical type theory such as HOL the principle role of the type system was historically to constrain the deductive system (particularly in relation to abstraction) so as to avoid the logical paradoxes. For this purpose the type system can be simple, logical strength arising by other means.

I mean, by and large our tools support languages that somebody else invented, not necessarily because they're the best languages, but because they're the languages that people want us to use. And so as soon as you start getting close to the commercial things, the technical merits of the languages aren't necessarily very significant anyway. If you get the choice, then that's great. If you get the choice to choose a nice language rather than a nasty language, then that's great, but generally speaking the market doesn't work in that way. I think HOL's nice. I mean, HOL's really... HOL's nice and does seem to have to had a reasonably good level of exposure and application. And it's nice because it really is simple, or simple in parts. I mean, it's got a great power-to-weight ratio.

2.9 Soundness of Logics and Proof Tools

I: And what about the soundness problem? Can I move up to soundness for a moment, and ask you: your theorem prover itself - have you got a proof of soundness for it, ProofPower?

RJ: Well, the logic... soundness is a concept about logic rather a tool. And the logic we use is the same as the one in... that the Cambridge HOL system uses, which is relatively uncontroversial as an extension of Church's Theory of types.

I: So it's Church's Simple Theory of types? And you don't think that a proof of soundness is important, because it's already been done in the logic?

RJ: Well, I think there is an informal proof in the Cambridge HOL [Manual], and I certainly wouldn't put formalising of a proof very high on the priority list.

I: Right. Why not?

RJ: Well, if you look at the kind of thing we're engaged in, and ask where the risks are, and what are the magnitude of the risks, the soundness of the logic is a tiny bit, a really tiny bit, and the correctness of the proof tool implementing the logic is slightly larger. It's still actually, for our , quite a small risk. And these are many orders of magnitude less than the problems you get when you start using them as tools, like: does my specification say what it ought to say? Is it a good model of the system we can talk about?

I: So compared to getting the specification matching the intention it's negligible?

RJ: Absolutely.

I: What happens if you're using your... ?

RJ: This doesn't apply to all logical systems. Some people use more controversial logical systems than this, but this is one of the least controversial logical systems around.

I: Would you be happy to use it in safety-critical areas? Are you that confident that the risks are so small that you could actually... ?

RJ: In the logic or the tool?

I: In the tool itself?

RJ: Yes, I'm not aware if there are any tools around that have got better credentials in terms of the risk of you proving something that isn't true.

I: What are those risks in general, do you think? I mean, outside your system, when you say the risk of proving something that isn't true, do you think in general with your systems and tools there is that risk, and not just a minor risk, but... ?

RJ: Well, there is the risk. It depends upon the system as to how serious a risk it is. It's certainly usual that tools... it is possible to do it with the tool. Normally people do screw up in the proof tools, so it is possible to prove false.[48] But unless somebody's actually deliberately going to exploit a flaw, it's not so... there's not so high a risk that that will actually accidentally result in your proof tool's truth. And we have ourselves found problems within the Cambridge HOL system, several of them. And they will be fixed.

I: They're fixed when you find them? You report them back, or you fix them?

RJ: Yes, or in some cases we've found, concurrently with...

I: You've found... ?

[48] I don't know that I would agree with that today. Possibly back in the 80's most tools had unsoundnesses at some stage in their development. Just as at one stage it was more common for published logical systems to be found faulty. I am possibly now out of the loop, but its very rare that I hear of soundness problems in proof tools.

RJ: In some cases, more than one party has independently found them. But all the ones that we've found we've found by thinking about the system. We've never actually done an application proof and ended up proving something we shouldn't have proved, and discovered that that was because of a flaw in the logic. I'm not sure I can remember the details, but... there were some problems with the polymorphism, which is the... one of the main features in which Cambridge HOL and ProofPower are different to Church's Simple Theory of types, and the other main feature is that it's got explicit policed rules for conservative extension, which must have interested Church. And getting the rules for conservative extension right is, in combination with the polymorphism, doesn't ... And that wasn't always right, I'll admit, but we've probably got it all right now.

I: So at some stage it was able to come out with non-theorems?

RJ: Well, it's always been able to through the known routeways, but you could say that they're not problems with the logic or the tool. People will only use those politically. And at one time we used to define... we had a method, a method of structuring proofs, which involved giving names to all the lemmas and the main lemma structure, giving names as constants in the object languages. And that's not actually sound, and we shouldn't have been able to do that.

I: Why isn't it sound?

RJ: It's not sound if the lemma that you're talking about is polymorphic, because a polymorphic lemma may have a different truth value at different type instantiations. So if you define a constant to be equal to the value of that proposition, the truth value of that proposition, by instantiating that equation at different types, you get a different value for a constant. And so when we actually figured that out when we changed the system, we had to... we couldn't put our proofs in.

I: They wouldn't go through?

RJ: The proofs using that... the proofs that used that method, in the simple way that we were using them.[49]

[49]That's a "yes". Once the definitional facilities were fixed to prevent this particular kind of non-conservative "definition" proof scripts exploiting that

I: So you had to redo them all?

RJ: We didn't redo them, no. I think that was after the completion of the project .. time. We just didn't use that [technique again]. You have to be slightly more subtle about it. If you want to give object language names to polymorphic lemmas, then you have to give them some parameters, which have got the tool labels. [50]

...[51]

With the LCF paradigm, you can give a , which is almost as good, so... I would say the same thing about... I mean, the other area which... where people... where the people often focus when they're worried about integrity of proofs is getting independent checkers of the proofs, as opposed to the tools you use to construct them.

I: Yes, run them through two things. You've got the constructor, and then run the proofs you've got through a proof checker, and that gives you more confidence.

RJ: But I think that's... I mean, in my view, that's a bit academic.

I: Why?

RJ: Well, for starters you've got that kind of structure built into an LCF system anyway. You've got a logical kernel which is checking all the proofs are constructed. And so as far as LCF's concerned, you're getting much more by doing that.

I: It's already being done, you mean?

RJ: Well, it all depends on what exactly the semantics are in-

facility would not go through.

[50] I must have been mumbling a bit here, and too many words have been lost for me to know exactly what I was trying to say. However, though we did not redo the proofs, it would not have been hard to sort them out. We were not exploiting the logical unsoundness of this kind of polymorphic definition, the technique we were using was convenient for the presentation of partial proofs, and we would have had to modify the technique in very small ways to get the proofs through. (Instead of defining a boolean constant whose value is the truth value of some unproven lemma, we would have to define a constant boolean valued function with one parameter whose domain type was the cartesian product of the type variables which occur in the lemma.

[51] There seems to be a gap in the transcript here, for the next paragraph does not seem to be on the same topic.

dependent of, you know? But I would... you know, the view I take is that an LCF system is designed to do that on the fly, and if you say, "OK, well, maybe you're... It's still better to export a proof tool verifying a checker." Well, maybe it is, but you're really talking, you know, here. This is not the area at which there is any risk, but, you know, the significant risks are completely elsewhere. That's not to say that it might still be a worthwhile bit of research for somebody to do, but in terms of the practicalities of what's significant commercially. I don't know what... 00-55 calls for this, calls for independent checkers. And I don't know how successful you'd be in arguing that an LCF system would have that.

I: But you haven't got a proof of soundness for your system, for your prover, have you? I mean, that's definitely the case?

RJ: A proof of correctness for the prover?

I: Yes.

RJ: No, we haven't got a proof. We have got... we have got some relevant proofs for it. We've got formal specifications of the logic, and we have got... and we have done some proof work in relation to the more innovative aspects of the controls in the logical kernel.

I: These are proof work towards correctness, but you haven't got the absolute correctness?

RJ: No.

I: How difficult is your prover to work, to operate? Would I need a... I mean, would one need a fair amount of training?

RJ: Well, for simple things it's easy. For simple things it just does its easy. There is a lot of it, and so you would acquire high levels of proficiency only over a long period of time. We teach it in quite short courses, and we find that people get on fine with this, so we have a sponsored student, for example, who's an undergraduate reading maths at Cambridge, and last Easter he did a one-day ProofPower course and then went on to develop various theories for it, and probably . Of course, it's easier when you're living in amongst a group of people who understand it. It's easier to...

I: In that environment.

RJ: ... employ the tools. But there's lots... there's lots of training material comes with it.

I: Presumably a proof... a prover itself will depend upon the

skill of the person operating it for its proof finding.

RJ: Well, it depends at what level you're talking about. You know, I mean, in... if it's a conjecture... I mean, the basic approach is that insofar as we can afford to do so we implement automatic proof facilities for special domains, and the guy using the tool is reducing his problem to those domains and then hitting it with the automatic proof.

I: So most operators wouldn't need to know everything that you know in order to operate in their particular domain?

RJ: Well, they certainly don't need to know everything that we do. They don't even have to know how to implement it. There's a lot of facilities there...

I: Do you think that... ?

RJ: ... because it may be being used in developing better facilities.

2.10 Surveyability of Proofs

I: What about the proofs themselves? Do you think it's important that the proofs produced by the machine should be surveyable by human beings?

RJ: Personally, I don't, no.

I: You don't know?

RJ: I don't. Personally, I do not.

I: So you don't believe that? Right. Why? A lot of people do things you could prove to be subject to survey.

RJ: Well, they are subject to survey, ...

[pause]

RJ: It's going to depend on why - what your purposes are, as to whether the survey's important. Some academics find the proofs interesting, and they want to be able to look at the proofs because they're interested in the proofs. And that's not relevant to a commercial application. At the high levels of the proof structure, then you may learn interesting things about the system by looking at the structure of the proof. And that's at a very high level relative to the nuts and bolts of the proof. In terms of wanting to reassure

yourself that the proof is correct, I think that's almost... I would say that it's a complete waste of time.

I: OK, what about 'surveyable by machine'?

RJ: What's important... what is important to surveying is 'what is it that's been proven?' What is this proposition that this thing has proven? And your scope for error is entirely... if you've got a good proof tool, if it says it's proven this conjecture, it's proven that conjecture. Your only problem is, what does it mean? Does it mean what you think it means? And that certainly does need to match your specification, and the nature of the proposition that you're claiming to have completely proven, what it means is something which has to be surveyed and understood by people. And that also applies to any residual lemmas which you've not proven. If you've got various assumptions you haven't completely discharged, then obviously the content of those assumptions is very important, because if they're actually incoherent then the whole thing's meaningless.

I: This has actually happened?

RJ: Oh yes. I mean, all it comes down to is you're trying to prove something about a system. If that thing that you're trying to prove is false, then it's still going to be possible to prove it, if you don't carry it right through to the end, because you just show that the truth of it follows from some other false claims about other parts... that part of the system. So long as your proof isn't entirely complete, then it could be entirely wrong.

(Pause)

I: Surveyability by a machine. Do you think a proof should be surveyable by a machine? I mean, if you don't think it should be necessarily surveyable...

RJ: What do you actually mean, 'surveyable'? I mean, as far as LCF-type systems are concerned, the proof will have been checked by the machine as it was constructed, but then it's thrown away.

I: Yes, but you might get to a point where a theorem prover is running through steps which you couldn't get a printout from, or anything of this kind.

RJ: Well, that's the norm.

I: Yes. So these are not surveyable?

RJ: Well, they're surveyed 'on the fly', as it were.

I: But I'm thinking about a case where you couldn't possibly get a printout, it's too much. What about such proofs being produced? "Yes, this is a theorem," and you've got no... nothing to survey at all.

RJ: One norm... well, the norm is, with the systems we operate, is if you survey anything you survey the script that produced the theorem, and parts of the log resulting from it. You never survey the detailed structure of the proof itself, and I don't think it's a problem. So I think it's not a risk they would... if you want to be sure that the tool is capable of checking proofs properly, then survey the tool, verify the tool, if you want... if you're that bothered. But this is... you don't expect engineers to hand-check the calculations that the computer does for them. Computers are good at doing these things, and the idea that a human being is going to provide a better check is a joke.

I: That's a joke? Because the length of these things would be too long, and the number of characters per line would have been, well, sometimes thousands, four thousand, right, on some of these theorem provers. So that would be a red herring altogether, the idea of having a...

RJ: Well, it might. Though... it is fairly popular to attach more significance to this than I do.

I: Why do you think that is?

RJ: Well, I can understand academics attaching more significance to it, because they don't always have the same interests and priorities, and some of them genuinely are interested in proofs, in the structure of proofs.

I: But from the point of view of soundness or correctness of the proof there's no advantage, you think?

RJ: I think that... I don't often hear... well, I don't think I've ever seen anybody make any comparative assessment of the significance, in terms of risk, of these different things. I think what people say about these things doesn't take that into account if you're talking about risk. But even people who advocate inspectability of proofs are normally doing so not from the point of view of that contributing to your confidence in the truth of the [theorem], it's

generally more in terms of it contributing to your understanding of the problem domain, and that being a major element of what you get out of doing the proofs. And that's a different thing. But you don't... to get that kind of understanding you only need to look at the gross lemma structure.[52]You don't need to look in too much detail.

I: And that lemma structure would probably convince you of the correctness of the proof by looking at the lemma structure in detail?

RJ: Well, the thing that convinces you that the truth has ... you're still not going to be able to check on the detail, and there's always going to be a risk, as far as your eyeballing is concerned, that you've not understood it properly and it's not true. getting the machine to check it really does add an assurance, there's no doubt about that, a lot more than you could ever get by looking at it yourself.

I: What about decision procedures, if you've employed those in your prover in some way, or at least your checker? Should those decision procedures themselves have a correctness proof?

RJ: Well, the decision... the kind of thing that we would point to if you were talking about decision procedures are double-checked anyway by the logical kernel, so as far as our systems are concerned, it's only the logical kernel that you have to worry about. For example, the thing which automatically proves things in linear arithmetic doesn't need to be checked, because if it gets it wrong it's only generating a proof that's going to check that logical kernel. Now, there is a lot of interest in sidestepping that obligation and using decision procedures which don't bother to use any of the proof, and then many proof tools do that. And you have got a problem there. You've got more code that you have to worry about whether it's correct or not, and ultimately one may be able to do these things using principles of reflexivity which involve you prove

[52]Its worth noting here that "gross lemma structure" means rather different things depending on whether you are talking about formal or informal proof. A lemma in a formal proof may correspond more or less to a single step in a journal level proof, so I am here saying that there is not for most purposes much value in looking at a formal proof at level of detail any smaller than one would typically see in a proof in a mathematical journal.

correct something first, although that's a lot of hard work. And in terms of the LCF paradigm, that doesn't sound to me like a very cost-effective place to put your effort. There's lots of places where you could invest in improving the automation without having to sidestep the rules.[53]

I: Again, you're more or less saying that the emphasis on correctness is not really worth it compared to the efficiency?

RJ: No, I'm not saying the emphasis on correctness isn't worth it, I'm saying here you're talking an emphasis on efficiency, on real efficiency: that is to say, you want to avoid generating the proof, because it's going to take you too long, and that creates a worry about the correctness. Now, in the general context of the capabilities of the proof tools, and the productivity of the process of producing proofs, it seems to me that there are plenty of places where you can make major steps forward without having to do that kind of thing, and therefore without entailing any risk, any additional risk. And, you know, we haven't got anywhere near the end of that line, and all the time we're working on it, the machines get bigger and faster. So the mill efficiency is not the dominant consideration, the mill efficiency is the side issue. The real issue, in terms of productivity, is 'how much automation is there in the tool, and what's the relationship between the tool and the man?'. It's not whether you can get... it's not whether you can get... how fast you can get tautologies from it, though this is sensitive to the application. So if you're talking about hardware verification, and you're trying to get very high levels of automation in hardware verification, then the picture changes a bit there. And things like resolution proof - really sort of heavyweight proof methods - tend to end up being exponential anyway. So if you do something to cut down to no cost there, it's not going to get you much further. It's not going to solve much more difficult problems for you. And the other thing is, in the LCF paradigm if you're talking about something like, say, a resolution proof, it's dominated by search. It's dominated by the time taken to search for the proof, not by the time taken to check it. So generating the proof is still not the obvious place to be... of course, if you get the generation of

[53] Without having to bypass the logical kernal.

the proof too inefficient, then it is a problem. But you can focus on... if you really think resolution proof is what's going to improve your productivity - and that's an interesting question whether... how much difference it makes, or any other kind of special area - then you can code up the proof search really efficiently without incurring any risk, provided you're prepared to just feed the proof right through the kernel when you'd found it.

I: What about hardware itself? We touched on the survey - what about the hardware? Do you think that has to have a verified design, a theorem prover, the hardware on which the theorem prover is running?

RJ: Yes. Well, if you're talking about probabilities here, you know, if you ask the question, "Somebody's sat down at a machine and developed a proof for a proposition - what's the chances of that proposition having been proved... being a false proposition which was proven only as a result of a hardware failure?" They're negligible.

I: Negligible?

RJ: Totally negligible. I mean...

I: Why is that?

RJ: These computers that we use very rarely have hardware failures at all. Most of the hardware failures is things like disks. Most of the hardware failures are immediately detected. And the chance of a hardware failure resulting in a proof going through which shouldn't otherwise have gone through is... I don't know what it is, but it's certainly not significant.[54]

I: No examples of this, obviously, from what you say.

RJ: Well, I've certainly never known that to happen. So now... that's not to say the hardware verification is a waste of time, but if that's your reason for doing it, then it's a pretty poor reason, because if you think you're going to prove something that isn't proven[55] because the hardware's not working...

[54] Of course, what's significant depends upon the application, upon just how important it is to get things right.

[55] true

2.11 The Future of Theorem Proving

I: What do you think of the future of theorem proving, the automated theorem prover? What do you think the future holds? Can you speculate or predict?

RJ: Well, my speculation is that eventually...

I: So eventually, then, the whole...

RJ: Are you talking about commercial? Are we interested in the commercial future, or just generally?

I: No, just in general, yes, your views about what direction automated theorem proving's likely to go in.

RJ: Well, I find it inconceivable that mathematicians won't eventually normally do mathematics using proof tools.

I: You really think that's going to be the case?

RJ: But I don't think that the proof tools that we have today are going to do that job, and it could be quite a long time. But why on earth out of all the scientists in the universe mathematicians should be the ones that don't use computers, that's pretty odd. And mathematical modelling is what science is all about, science and engineering is all about. That's not necessarily the kind of mathematical logic that people in formal methods are doing, but that's what it's all about. And you can do more with proof than you can do with... you know, you can do anything with proof technology, in terms of mathematics and logic, anything that can be done. The kind of tools that are out there at the moment have strictly lesser capabilities. Eventually proof tools will be able to do all these things, and that ...

I: But that's the direction it will go. Will they be completely automated, or still need the human input?

RJ: It's never[56]going to be completely automated, because it's a creative activity. So you're always... there's always going to be a human being there constructing the model, and deciding what he wants to know about the model, and there's always going to be conjectures that the man and the machine together can prove that neither can by themselves, that are really tough conjectures. You know, it's not... it wouldn't be difficult to concoct a program and a

[56]Maybe not that long, not in my lifetime.

specification of it such that the program was correct if and only if Fermat's last conjecture was true, and, you know, we may or may not have a proof of Fermat's last conjecture now, but it certainly took a hell of a long time to write.

I: We haven't got it, in fact.

RJ: It's been... ?

I: The gaps...

RJ: That's the latest view, is it?

I: Well, we think the gap is that it's a substantial gap. It really is not the trivial gap that was first thought.

RJ: So if you write a program which depends upon the proof of Fermat's Last Theorem for its correctness, then it's not going to prove it automatically.

I: No.

RJ: It's probably not going to prove it at all.

I: OK, is there anything else you wanted to add, any views you want to express before we come to the end? Anything you think I've left out that is pertinent?

RJ: Well, I'm very interested in what it takes to make the kind of cultural change that... I think the only thing to say about the use of proof tools by mathematicians is that I don't believe it's going to be a case of mathematicians are going to do what they do now in future using proof tools. I think they're going to do something different using proof tools. You know, the proof tool makes so much of a difference, especially proof tools where they're going to be good enough for mathematicians, which we don't have any yet, or at least proof tools with sufficient maths built into them that they're going to change the mathematics that mathematicians do. To some extent. There's pressure, anyway, from the fact that mathematicians are... seem to be increasingly finding that they're trying to prove things which are seriously difficult and complicated to prove, and they're losing the ability to do them without computers anyway. But it's going to change the nature of, particularly, mathematics.

I: I can see that it will do in certain areas, like, as we saw in combinatorics, or something of that area, that something like Fermat's Theorem, which depends upon some theory of elliptic

functions, or whatever - do you really foresee a time in which a computer will be used there?

RJ: Well, it all comes down to whether it's easy to do, if at all. At the moment it doesn't come down to that, even if they were good enough, because all the professional mathematicians around have grown up in a culture of doing proofs the way the way they do proofs, you know? But at some time in the future mathematicians are going to grow up and go through university using proof tools, and they're going to have an entirely different...

I: Just as they now use calculators at school, you mean? The same sort of thing? It would be, "OK, here's your proof tool," in your undergraduate class, and you can use it in your exam, or whatever?

RJ: And in my view, if you have an interest in getting mathematicians to use proof tools, the best place to get that ball rolling is not to go to people who are doing research in mathematics, it's to get undergraduates to use proof tools, because the guys who are already doing proofs are going to carry on doing proofs the way they have done already. They've learned how to do that. It's the guys who've learned to do it different that are going to do it different. Now...

I: New paradigm.

RJ: I'm not very familiar with exactly what happened, but I had this woolly perception, about fifteen years ago when... shortly after Lightfoot[57]put the clappers on AI at Edinburgh, that there was a sudden discovery that AI was the way to do psychology. Now, I don't know whether this was because everybody left computer science departments and suddenly found themselves in psychology departments...[58]

I: A diaspora, yes.

RJ: ... but that surely must have been a major shift in the way people did psychology, and it seems to me that that's more the kind of model of how proof tools would eventually affect mathe-

[57]James Lighthill (1973): "Artificial Intelligence: A General Survey" in Artificial Intelligence: a paper symposium, Science Research Council

[58]Apparently, the theoretical group in the Edinburgh "School of Artificial Intelligence" was simply renamed "theoretical psychology" at about this time.

matics than the kind of idea that they're going to carry on what they used to do, and eventually they're going to start using proof tools for it. The mathematicians... I would expect, not being a mathematician, I would expect that the kind of mathematics that gets done at the moment is, to a significant extent, conditioned by what is possible for a mathematician to do with a pencil and paper, to do with the kind of complexities he can cope with inside his head without any tools to help him. And that determines what they think is interesting problems, and what's sufficiently abstract and interesting, and the dynamics of that eventually are going to change when the tools are in place.

I: How's it going to happen? How is the change... how are the proof tools going to be introduced, say, to undergraduates, if their mentors are not using the proof tools? How's it going to happen.

RJ: I think the confusion between computer science and mathematics is helpful, because discrete mathematics is taught with computer scientists in mind nowadays, but it's kind of a bit of elementary mathematics that everybody does. And that tends to be computer science-oriented now. And I think we will certainly see tools, proof tools, moving into that patch. I think moving much beyond that will be dependent hem being able to cope with more mathematics than that. At the moment they're only good for discrete mathematics, really. But we're also seeing, at the moment, that symbolic maths tools are getting more significant. People are beginning to appreciate the merits of these, and...

I: Well, they are being used to solve differential equations. Mathematica is one such tool which some mathematicians are certainly using to ease solution of differential equations.

RJ: They're being used as educational tools as well, and eventually proof tools will be able to do that kind of thing as well, and much more as well. And it's a bit of hump to get over, you know, to get to that point. It's not at all clear how you're going to get to that point, because there's a huge amount of work involved. But it will happen, eventually. I just hope I'm still in the game!

Appendix A

Research Topics

Here are specifically the research topics I considered for a PhD dissertation.

A.1 Keele

A.1.1 Background

The following is a transcription of a note that I made, probably in early 1976, while an undergraduate at Keele University looking for a University to do research towards a PhD.

The professor of mathematics at that time gave me the advice (which now seems to me rather bad advice) that I should avoid doing an MSc and get straight into working on a PhD. So I wrote down the kinds of things I wanted to do research in and sent the note to some Universities to ask them whether I might be able to do that kind of research with them. I think it just went to Oxford and Manchester at first, then Peter Aczel at Manchester said that the Computer Science department at Warwick would be a good place to do that kind of work and suggested I talk to David Park. The copy to Oxford was addressed to Dana Scott, with a covering letter explaining that I didn't want to do the MSc but was looking to get straight into research. He replied very concisely to the effect

that he was not able to help me. Later I got a more conciliatory response from Robin Gandy, who invited me to go and talk to him, but by that time I had already agreed to go to Warwick. For more than one reason this was unfortunate, and I would have been much better doing the MSc at Oxford.

A.1.2 Possible Topics for Research in Mathematical Logic, Foundations of Mathematics, and Computer Science

High Level Logics

My most consuming ambition in Mathematical Logic concerns the development of "High Level" formal systems of logic. By "High Level" I do not mean "Higher Order", I mean something analogous to the use in "Higher Level Programming Language".

The purpose of a high level programming language might be:

(A) To facilitate the precise formulation of algorithms (possibly in some special problem area, but not necessarily)

(B) To enable people to make use of the facilities of a complex computer system without needing first to acquire detailed knowledge of the workings of the machine and its software.

The "sine-qua-non" of a programming language is translatability.

By analogy the purpose of a high level formal system of logic might be:

(A) To facilitate production of completely formal proofs of mathematical theorems (possibly in some particular branch of mathematics but not necessarily)

(B) To enable mathematicians to produce completely formal mathematical proofs without needing to master all the intricacies of mathematical logic and set theory.

The "sine-qua-non" of a formal system of logic is that there be an effective method for verifying proofs (not to mention that it should be consistent!)

The development of such languages is a prerequisite of any extensive practical use of computers for proof finding.

The Foundations of Mathematics

Concurrently with work on the above some work on the foundations of mathematics would be necessary. I could of course stick to formalising versions of the tried and tested axiomatisations of set theory, but I have traces of intuitionism lurking within me and would like to investigate other possibilities.

One possibility which interests me is that of basing mathematics on strings rather than sets. Strings however, unless infinite strings were admitted would serve only for a version of constructive analysis.

Computing with Reals

The principle source of my intuitionist leanings was the perception several years ago of the disparity between theory as embodied by real analysis and practice found in numerical analysis. When I was familiar with the way mathematicians use numbers on computers the beautiful completeness and simplicity of the real number system seemed a rather shallow pretence. In practice all mathematical computations were approximated using using a subset of the rational numbers. What seemed worse than the understandable failure of numerical analysts to use real numbers was their failure to exploit the full range of computable numbers. It seemed to me that it was just because mathematicians allowed themselves the pretence of real numbers in their theory that they failed in practice to fully exploit the computational capabilities of their machines. It is well known that a wide range of functions are computable, it would be a good thing if computers were could be organised so that any "computable" function can in practice and without much difficulty be computed within any specified degree of accuracy (space and time permitting). This is not presently the case. A typical example of

the deficiencies of the present techniques is a problem as simple as the inversion of a matrix. Any non-singular matrix with rational entries has a computable inverse and so one could reasonably expect that any substantial computing complex would have facilities to compute any such matrix to any prescribed accuracy. In fact, I doubt that there is any complex which will guarantee to compute any inverse to within 1% accuracy (subject to time and space limitations only). In practice the accuracy of the inversion will depend upon the size of the matrix, and more importantly, on how close to being singular it is. In general we can only guess at how accurate the inversion is.

It would be interesting and valuable (and not trivial) to investigate the software and hardware necessary to render practicable the computation of any "computable" function to an arbitrary number of places. On the more purely mathematical side there is the problem of sorting out which of the problems of analysis known to have solutions actually have computable solutions, and the problem of arriving at practical algorithms for obtaining those that are computable. It must be emphasised that the algorithms here referred to are quite different in character to the algorithms used in numerical analysis at present, which are generally algorithms for approximating solutions. There is no way of obtaining solutions of arbitrary and known precision without abandoning present methods of representing numbers in machine memories.

Glossary

CESG Communications and Electronic Security Group (a part of GCHQ).

class A large collection.

GCHQ Government Communications Headquarters, UK.

HOL Higher Order Logic, a specification language, a formal deductive logic, an interactive proof tool.

NuPrl A tool supporting proof development in a "New Proof/Program Refinement Logic", which is a constructive type theory.

PRG The Programming Research Group, at the University of Oxford.

set A small collection.

Bibliography

[Aye71] Alfred Ayer. *Language Truth and Logic*. Penguin Books, 1971.

[Car50] Rudolf Carnap. Empiricism, semantics and ontology. *Revue Internationale de Philosophie*, 4(11):20–40, 1950. Also in: [**?**].

[Jef67] C. Jeffreys, Richard. *Formal Logic, its Scope and Limits*. McGraw-Hill Book Company, 1967.

[Kör60] Stephan Körner. *The Philosophy of Mathematics*. Hutchinson, 1960.

[Mac01] Donald MacKenzie. *Mechanising Proof - Computing, Risk and Trust*. MIT Press, 2001.

[Min67] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, 1967.

[Rus19] Bertrand Russell. *Introduction to Mathematical Philosophy*. George Allen and Uniwin, 1919.

[RW70] Bertrand Russell and Alfred North Whitehead. *Principia Mathematica to *56*. Cambridge University Press, 1970.

Index