A. M. Turing Award Oral History Interview with

Leslie Gabriel Valiant

by Michael Kearns

Cambridge, Massachusetts

July 25, 2017

Kearns: Good morning. My name is Michael Kearns and it is July 25th, 2017. I am here at Harvard University with Leslie Valiant, who's sitting to my left. Leslie is a Professor of Computer Science and Applied Mathematics here at Harvard. He was also the recipient of the 2010 ACM Turing Award for his many fundamental contributions to computer science and other fields. We're here this morning to interview Les as part of the Turing Award Winners video project, which is meant to document and record some comments from the recipients about their life and their work.

So good morning, Les.

Valiant: Morning, Michael.

Kearns: Maybe it makes sense to start near the beginning or maybe even a little bit before. You could tell us a little bit about your upbringing or your family and your background.

Valiant: Yes. My father was a chemical engineer. My mother was good at languages. She worked as a translator. I was born in Budapest in Hungary, but I was brought up in England. My schooling was split between London and the North East. As far as my parents, as far as parents, I suppose the main summary would be that they were very kind of generous and giving, that I don't recall them ever giving me directions or even suggestions. They just let me do what I felt like doing. But I think they made it clear what they respected. Certainly they respected the intellectual life very much. I think my father would have wished for me to have a scientific career. I also have a sister, an older sister who did physics and has an academic life.

Kearns: Maybe tell us a little bit more about what your schools were like and what your interests were at that time and any hobbies or other activities besides school that you were fond of.

Valiant: Yes. I had lots of hobbies. I think I went from one interest to another. I had a period when I was maybe 10 or 11 when I was very much into electronics and building things, building transistor radios. I worked also with planes at some point and rockets. But I was also interested in more basic physics. Kind of things which interested me, building a thermometer from a

bottle with a little tube and a drop of water and seeing how the volume of gases was proportional to temperature. I was rather fascinated by the fact how robust this was. You couldn't spoil the experiments. You made the bottle big or small, whatever you made it, it always worked.

Kearns: These were kind of tinkering projects that you just developed on your own and gathered the materials needed and kind of just figured it out by yourself?

Valiant: Yes, it was very much all tinkering. Yes, yes, yes.

Kearns: You mentioned a certain amount of electronic tinkering, which leads to the natural question since you eventually made so many contributions to computer science, what was your knowledge of computers at the time? What was your first experience of computers? Did you know anybody who owned a computer? This sort of thing.

Valiant: No. I think I was reading there's various things you could make out of transistors, things like that, and always the last chapter there they were also being used for computers, but I usually didn't get there. As far as… One influential bit of my schooling was that right at the end I had a teacher who was rather generous to me. I finished my high school in slightly unusual order, so I had a five-month kind of gap period before I went back to school to do some exams. He found a job for me in a medical physics lab in one of the medical schools in London. The main influence on that was I was doing experimental physics, helping experimental physicists. So the main influence was that I probably decided that I didn't want to do experimental work as a career, so it was good to get that done early. But also in the last two weeks, they actually sent me to do some computer programming, so I ended up with this green punched tapes and doing some simple program to find correlations. I did that, but it didn't strike me as what I was going to do for the rest of my life.

Kearns: This was your first introduction to computers and to sort of the notion of general purpose programing in some sense?

Valiant: Yes. Yes, it was.

Kearns: What language was this program?

Valiant: It must have been ALGOL I think. I suspect it was.

Kearns: So you weren't flipping switches on a circuit board?

Valiant: No, no.

Kearns: This brings us to your time at Cambridge. Maybe you could tell us a little bit about what you studied at Cambridge and what the environment was like there and the influences on you at the time.

Valiant: Yes. I did mathematics at Cambridge. In Cambridge, quite a wide variety of students do mathematics who don't necessarily want to do it professionally. Many go on to do other things, some physics, economics, and whatever. It's a fairly broad thing and one does with pure and applied mathematics. I still had at the back of my mind possibly turning to theoretical physics, but in the end I never did that. I just did mathematics throughout. Again, I didn't think much about computing. There was one small course in applied mathematics where you did some simulations of haystacks using computers, which I found quite interesting.

Kearns: I want to follow up on one thing that you mentioned which I think is a little bit unusual certainly in modern times but maybe even at other universities at that time, which is this notion of mathematics as sort of a general education for people who then go onto work in other quantitative areas, which I think in my experience is not a view that's widely taken of math today for example. I'm wondering if you could comment a little bit about the culture at King's of math students at the time.

Valiant: Yes. Well, I think this is a long tradition in Cambridge. I think it's had very good effects in making other fields more mathematical. Like physics was very strong at Cambridge, economics was very strong, so a lot of non-mathematicians benefit from this. I supposed mathematics just had a high prestige, so to prove that you were clever, you did mathematics even if that wasn't going to be your career. I think that was the…

Kearns: And I mean, were you aware at the time you were at King's and Cambridge that Alan Turing, who would be not just the namesake of an important award you would receive many years later but have such a fundamental influence on your work, did you feel his legacy there or were you aware of him, or…?

Valiant: Well, amazingly no. I was at King's, the same college which Turing had been both as a student and as a fellow for a long time. But amazingly I never heard of him there. Then later on when I went into computer science, I got to know a lot about Turing machines. I never knew much about Turing himself. Somehow he wasn't talked about, who this person was. Then once many years later in a library, I just chanced on his mother's book about Turing, Sara Turing's work, and there on the first page it said he went to King's. I was shocked that I didn't know this.

Kearns: That's interesting. What about after Cambridge? How did you find your way to graduate school and how did you make the transition from

kind of pure math into theoretical computer science and early computational complexity?

Valiant: In Cambridge, I think I thought quite a lot about what I wanted to do after Cambridge. I thought I wanted to do research and probably mathematical research, so I talked to lots of people in Cambridge, mostly in the applied math/physics area. But I couldn't quite find anything to commit to. Then I went to a one-year so-called diploma course at Imperial College London on computing, which was a very sort of generalist course. It wasn't theoretical at all. There I had time to think more, then again at there had to decide where to apply for PhD.

Then I started reading papers of various people who were in Britain. The paper which did influence me on my direction was reading a paper of Mike Paterson who'd done a PhD in Cambridge earlier on computability under the direction of David Park. Reading his paper and some background, I kind of discovered basically the depth of Turing's computability theory, that it was a rather particularly novel and staggering comment on the limits of our intellectual capabilities. I thought this was something totally different, something that wasn't obviously understandable, mysterious. So then I went to Warwick where David Park was and Mike Paterson was about to come.

Kearns: Tell us a little bit about the first problem that you worked on in your first publication.

Valiant: Well, I spent two years at Warwick. Mike Paterson became my advisor. He was interested in algorithms and complexity, but he'd also had this background in computability theory. [0:10:00] He suggested to me two problems in automata theory which were unresolved about whether things were computable. The one I worked on most, which was what my PhD thesis on, whether two machines with pushdown stacks, so these abstractions which are widely used in compilers, whether they're the same or not. This was one of these problems which is still one of the most wide-open problems in that it's not just… the question was still open whether you could compute this or not or whether there was an algorithm. I never resolved this. I proved special cases. [chuckles]

But it was probably a good problem to work on as an intellectual challenge, just because it was difficult. It was a new kind of problem, you had absolutely no idea what techniques to bring to it. There weren't all these techniques to apply. You had to develop your own. So I think it was maybe a good background for computer science in the point of view that problems abound but there isn't a pile of techniques which you know should be applied to these problems.

Kearns: But you're still at this point thinking about problems that are kind of in the outer reaches of the sort of complexity hierarchy and

worrying about just sort of problems of being primitive recursive or even being computable at all.

Valiant: Well, yes. But I mean I was fully aware of analysis of algorithms and worrying about things at every level. In terms of for this particular problem, there was so little known that we just had no way of proving anything more efficient.

Kearns: I mean to move onto another very influential piece of work that you did I think shortly after your PhD which has a strikingly different character, which was your work on context-free grammar recognition in sub-cubic time. That's quite striking, just sort of exploring the boundaries of what's computable to giving a very precise upper bound that I think to this day many people find surprising. Maybe you could tell us about how you made that transition and how you came to be interested in the problem and whether it required in your view very different ways of thinking than the work you'd been doing at Warwick.

Valiant: Well, the work I'd been doing at Warwick did involve complexity at this practical level — *n*-squared, *n*-cubed — and I was familiar with the problem of these grammars. But then independent of this, I got to know Volker Strassen's work on matrix multiplication, which has influenced me a lot ever since, which is about multiplying matrices fast using identities which aren't traditional mathematical identities but arbitrary ones invented for the purpose. I think this was very influential on me and I think others also because it showed that algorithms aren't just an afterthought of mathematics. It's not that you find relationships among things and hope that some are constructive but that you can create new mathematics for the purpose of finding algorithms.

So I was fascinated by that and knew something about context-free grammars and I somehow… I don't know the process by which I brought it together so it worked out, but I do have vivid memories of actually getting the results. Maybe it was… It was the first result where there was some pleasure in it, in knowing that other people will be amused by it or that want to know about it. It's not that you have to persuade them about it but… Although I'm not one of these people who talks about the beauty of mathematics, in fact after the result, I do have the insight, this thought that "This is like poetry, but it's more constrained," that you have to create some new image but it's very constrained.

That was one thought. Another thought was that there was an accidental element to it. I had drawn a diagram just in the right way. Maybe if I hadn't drawn it from the right perspective, I wouldn't have got the idea or maybe would have got it the next day. Then another thought I had was that in fact the first time you discover something, you don't understand it very well. So for several days afterwards, I hesitated, "Is it right? Is it wrong?" So many people report that experience of discovery. I had multiple aspects of that.

Kearns: Les, I wanted to ask next about the period of time kind of in the mid and late ’70s and into the early ’80s where you made a lot of fundamental contributions to circuit complexity and gave lower bounds there and also started to work on the algebraic problems like permanent that led to all of your work in #P and enumeration problems. I was wondering if you could just tell us your own view about that period.

I also wanted to see if I could tease out any thoughts you have about the fact that up until that time, a lot of your early work had been on solving sort of very specific problems that were extant already, and certainly with #P, it seems like a time in which you kind of discovered an entire world that then went on to be populated by many other people as well and really was not just solving a problem but really discovering an entire universe that had not been discovered before and its relationship to the other complexity classes. I’m wondering if you could maybe talk a little bit about how that felt and how you came to that.

Valiant: Yes. In some sense, the difference between working on technical problems and things which have somehow broader meaning, for me that’s not a very sharp distinction because I think most of the things I find interesting need… both of the problems need to have some broader meaning, but need to make some technical contribution to bring things forward from current intuitions.

Maybe one example, which I suppose we’re going to well come to, is one on parallel computing on routing where the problem is that if you want to route information in a network, and have lots of packets but only one packet can go down the wire at any time and you want to make sure the last packet arrives in time, well, what do you do? The ultimate solution was that you randomize in space. You send things to random places then to the right place. The point is that once said, this idea’s so obvious that the question is “Why doesn’t someone think of it and then try to prove it?” I didn’t think of it either like this and it came about by some more tortuous way of small technical steps and finally this is what one found. So I think new broad conceptual things arise from trying to solve particular technical problems.

Kearns: Was that the case also with sort of the discovery of #P? Maybe you could just briefly take us through perhaps the more technical specific steps that kind of eventually opened onto this much, much bigger world.

Valiant: Right. I think the specific question I had then and still have is that the most impressive algorithm we have, which seemingly can do an exponential number of different things, is linear algebra, which I learned in high school. The question is why can’t linear algebra solve NP-complete problems? This would seem even a simpler, simpler problem than the general problem of what is the

connection between P and NP? Basically I think this is what I was trying to find out. So early on I found this permanent function, which was a pretty obscure function at the time which not much was written about. Combinatorialists knew about it.

Kearns: Do you remember where you first became exposed to it?

Valiant: Actually I don't. I presume I kind of invented it and then realized it was well known and that I came across it somehow. But I remember, so maybe it was 1976, I met Dominic Welsh at a workshop and he was a combinatorialist. We had a discussion and at that point I didn't know which way I was going, what's easy, what's hard. I mentioned to him this permanent and determinant and his comment was "If I can reduce the permanent to determinant efficiently so I can compute the permanent fast, that would be a publishable result." [chuckles] Okay? It turned out that I had to go… I went the other way. But the specific problem I was working on, there's no reason even now for believing that you cannot do NP-complete problems with linear algebra.

Kearns: Not just in linear time but in fact in linear time with linear algebra.

Valiant: Right. It doesn't have to be linear time, but linear algebra in a certain amount of time. So in some sense, the #P-completeness came out of a failure to get positive results.

Kearns: I see.

Valiant: Yeah. So first I got the result about the permanent using some very technical reduction and then I started thinking about other problems which are counting problems [0:20:00] and where they lie. So lots of problems where you compute probabilities, reliability of a network breaking down, and counting, all kinds of stuff. Yeah, so this was kind of a… Yeah, so the surprising part was that at the time people thought that NP-completeness is one explanation of difficulty. The idea of finding NP-complete problems where actually finding solutions is easy but just counting them is hard was something which wasn't seriously addressed, and so I remember once to a well-known person describing this on a train to a conference and they said, "Oh, something must be wrong here."

Kearns: I see. Were you surprised by not just the richness of the world that you discovered but the sort of subsequent centrality it had in many other complexity theoretic questions and just things like Toda's theorem and just even to much more modern times, the sort of central role it plays in many problems in learning and inference, which are essentially counting problems of one kind or another?

Valiant: No, I think I understood that counting is a pretty fundamental… Counting integration, adding up. I understood that so much of calculus… Calculus is all #P-complete in high enough dimensions. So I realized that in some sense in high school we'd been misled, that we thought these were things that were easy to compute and we weren't told that in high dimensions, everything is hard.

But as far as the generality, yeah, I mean there are various clues. So one particular recollection is that… I will say I had a student, Carl Sturtivant, who also had a Cambridge kind of scientific background, and so we discussed quite a bit about the permanent as a function in physics. In physics, the permanent is the wave function of a set of bosons. It is a function that describes…

Kearns: So the theoretical physicists' world where have been…?

Valiant: Right. It's the best-known description of a physical situation. Then obviously the question was "Well, what does this mean, that there's a physical quantity which is worse than NP hard, it's #P-complete?" Of course we couldn't… This was before quantum computing, before Feynman's paper, so there was no discussion of quantum computing. But of course we couldn't find any way of making a concrete computer out of this, couldn't think of a way in which you could configure a situation and you measure something and there was this permanent.

Kearns: But you contemplated it at the time?

Valiant: Yeah, yeah, yeah. We certainly contemplated it. Yeah, so we didn't know what it meant.

Kearns: I see. The thought occurred to you that we have this problem that classically is quite difficult, but on the other hand nature somehow does this, and if only you could harness that then you might be able to kind of circumvent traditional complexity limitations.

Valiant: Yes. Just wondering what it meant, really.

Kearns: And it seems that that series of works also laid down a longstanding foundation of interest in algebraic complexity that actually you've been revisiting in many ways recently with holographic computing and accidental algorithms for instance. Can you just say a little bit about that kind of substrate of your work and that long-standing interest and the sort of modern variations on it?

Valiant: Yes. That I think I continue to be very interested in. After the #P-completeness papers, I had a paper where I looked at a completely algebraic version of it. Why this I think continues to interest to me, I supposed the simple

motivation of that was to cast these computational problems in a mathematical way, so that maybe techniques from classical mathematics could be applied. But thinking about it more, I think there's some deep philosophical issue there, so clearly this NP-completeness phenomenon was discovered in the computer science context, it was discovered by computer scientists, and maybe the most spectacular consequence of P equals NP would be that mathematical proofs could be checked efficiently.

On the other hand, and I think what this algebraic line of work shows, is that this P-equals-NP phenomena is part of a much broader set of phenomena which are purely mathematical. Mathematicians could have discovered this 200 years ago, but didn't. There's nothing computational about it. It's something much deeper. So we don't know what branch of mathematics will resolve which ways these questions go, but it shows that computation is very deeply embedded into trying to do its mathematics. There's no distinction basically.

[Recorder was paused here briefly because of a clock chime]

Kearns: You mentioned a while ago in passing your work on randomized routing, which was an early instance of what seems to be a very longstanding thread of fascination with parallelism, whether it be in computers or in the brain. I'm wondering if you could talk a little bit about that interest and how it's evolved over time.

Valiant: Yes. Parallel computing is a basic question in computer science I think from the beginning, just because in sequential computing, they're so satisfied that the Turning machine–von Neumann dualities has been so successful. The question is whether one can do parallel computing as successfully in a general purpose way and as easily. This has always been around.

I suppose there are two main stages in my work on this. The first one was to do with the communication, that the main problem… the first problem we have to overcome is one of communication, that computation you can do locally, but if you want to communicate among lots of processes, that's the bottleneck. Actually on this point, that's clearly also a bottleneck in the brain. Communication is a bottleneck in any kind of parallel computing.

I worked on this problem of routing, so it was related to queuing theories. Queuing theories existed where you got lots of people in queues, but queuing theory concentrated on just the average, what the average waiting time of a single person, whereas here the issue is that you're trying to route a large number of packets and what's important is when the last one arrives, because that's when you can start the next step. So then this problem had a positive solution. I think I thought of publishing a paper called "Parallel computing is

possible," but then in fact the routing was somehow more general communication, so I gave it a more general title.

This was one phase. This was about 1980. Then after that I built up a picture of what a parallel computer would be, which would be have some big network, having some good general purpose routing thing, and that maybe at a high level you simulate a nice programming language, like a PRAM is an abstraction where the programmer doesn't have to worry about locality. But there's still something missing there, because even if you could do all this somehow, it didn't lead to paradise. Somehow things were still difficult.

As far as recollections, I remember going to a meeting most probably in late 1988 in California on parallel computing, and in parallel computing always everyone comes and describes their own angle, but there's never any summary. I remember thinking, "Yeah, of the things said here, I think all the important things are already known and have been said, but somehow we don't understand it, it hasn't been encapsulated."

Then at the same time I'd volunteered to write an article for this handbook of computer science, which was edited by van Leeuwen. I did one on parallel computing. Although normally I hate writing these summary things, I thought this would be good for me, because I knew that this field was totally in chaos in my own mind, so I thought it would be good for me to organize it in my own mind.

Also there was a plethora of models of parallel computation at the time. The idea that one needs a new one, that seemed the worst idea, so no one thought you needed a new one because there was so many already. But this bulk synchronous model actually came out again kind of by accident. I was sitting at a desk and I was literally trying to do reductions between various models, so "This model reduces this model efficiently." Then I did a few of these and there was this thing in the middle, [0:30:00] and that was the bulk synchronous model. The one mere aspect was that this wasn't one of the thousand models already known, it was a bit different, so I thought maybe it meant something. That's how I came to that.

What it is, is… I call it a bridging model, so one thought I had at the time was that… I was about 40 years old and I was a computer scientist, but I'd never thought carefully about the difference between a Turing machine and the von Neumann model. I thought this is shocking that… how come… So I did think about it and then the interpretation I came to is that the Turing machine is some pure mathematical model, it's here to stay, it's perfect for what it is, while the von Neumann model is more kind of a pragmatic thing which is influenced by physics. The amazing thing about the von Neumann model is that even as technology changes, it implemented efficiently over the decades. Then I thought, "This is the paradigm to which parallel computing should aspire." Part of the bulk synchronous model is a pragmatism which is possible in physics.

Kearns: I see. Interesting. Les, I want to move on now and talk about the phenomenally influential work you did in the theory of machine learning and your development of the probably approximately correct or PAC learning model. First of all, when did you first become interested in learning as either a natural phenomenon or a computational one?

Valiant: I can't quite place a date on that. I think it came slowly, but I became kind of intensely interested probably in the year or two before that paper was published. I can't quite trace the origins.

But I suppose the philosophy was that I was interested in cognition, that cognition is such a powerful phenomenon in the world that it had to have a theory. It couldn't just be some heuristic where you got such a reliable phenomenon. There are many ways of asking the question, but one idea I remember having is that if one looks at a conference proceedings of an AI conference at the time, there was various topics – planning, search, reasoning, whatever, machine learning. Then I thought that all these things have meaning, but it's unlikely that all of them are the fundamental theoretical building blocks of AI. So the question is "Which are these is fundamental?" When asked that question, I think it became obvious to me that it was learning.

Kearns: What did the field of machine learning look like from your perspective at roughly that time?

Valiant: Well, I think I was still thinking of the problem more broadly as cognition rather than machine learning. Theories of Newell and Simon were quite dominant. Learning was part of their theory, but for them, learning was not statistical at all. There was no generalization in the sense we understand it today. It was learning more meant that you had some experience and the next time you came across the same problem, you compile your previous experience and do…

Kearns: So something closer to rote learning or…?

Valiant: Yeah, but doing it better next, some sort of optimization. You use reasoning to understand why something worked last time and do it differently this time. But it was compiling knowledge. It's a field called explanation-based learning where you try to find explanations of…

So this is what AI looked like for me. I was interested in learning, so I did… from undergraduate days, I think I did eight lectures on statistical inference once, and so our first question was statisticians talk about inference, so that must be something to do with this, but it's clearly not an explanation of cognition because if it were, we'd know about it by now. I think more technically how I phrased the

question was “Why isn’t statistics enough?” And I read some stuff, and the kind of stuff I read were people thinking about probability and cognition.

I read this book on probability theory by John Maynard Keynes. Of course he was an economist but he also wrote an interesting book on probability in his younger days, which again had some [cognitive? collective?] influences of people, try to find properties of why you believe things. Also, so Carnap, was a philosopher who had a more probabilistic reasoning kind of explanation of cognition. The thought came that, well, what’s missing in these is any kind of quantitative explanation why it is. When do you have enough data? When do you have enough computation to do what you do?” I thought clearly this is where computer science and complexity theory had something to offer.

Kearns: I wanted to talk about two other remarkable aspects of that early work and learning, one that’s sort of specific to your own line of work and one just about theoretical computer science more broadly. On the first point, up until that time, although all of your papers had a very strong conceptual basis and were trying to get, even when they were solving specific problems, to some bigger issue that you thought was fundamental, they had rather difficult proofs in them and there were a lot of arguments that would have been extremely hard for anyone who wasn’t working in theoretical computer science to really follow, whereas the first PAC learning paper has technical results in it, but they’re of a much more elementary nature, to the extent that even people outside of the theory community could kind of appreciate what was being said and see things like the combination of a union bound and a Chernoff bound in a simple, provably correct learning algorithm. So it feels like in that paper you’re really trying to lay out an entire model or framework that you intend or hope for other people to go and populate, which of course succeeded spectacularly in the decades since.

But I’m wondering if you thought differently about that paper both in terms of what your goals for it were going forward and in terms of the technical content relative to earlier papers you’d written, which were in some sense much harder to follow.

Valiant: Right. Well, I think maybe the simplest answer is that in some papers, the interest is in the… in my mathematical papers, the interest is in the proof, but in some of these, the interest is in the definitions. This one clearly, what the paper is about is a definition, is defining what you want to achieve when you learn. So the main thing is a definition, the definition should be simple, and so certainly when writing paper, I was hoping that it would have a broader audience. The aim wasn’t to hide the main content with incomprehensible stuff, so I think it was fairly deliberate that… I was hoping that people would understand the definition and also that some of the things which followed were striking and simple, that sometimes deep phenomena have simple explanations.

The fact that for certain learning mechanisms there were simple proofs that they really learned, so I was struck by the fact that it was possible at all because I hadn't seen that before, that having a proof without assumptions, that something generalizes.

Because certainly in our community, the idea of having any kind of rigorous theory of learning was thought of as absurd. People thought of learning as being some phenomenon which had no scientific explanation. It was ill defined. I also remember some personal reactions. People basically wondered, was I serious? Kind of a basic memory I have is going to maybe the conference where I presented the paper, the basic paper, and then someone was kind of friendly to me and sympathetic to me. Kind of we're talking like this, but then we're standing, but then he came up next to me to show that he was really on my side, and he said, "Are you serious?" [laughs] I think that was kind of the reception, because this was trying to move computer science in area where people didn't think it could move.

Kearns: Yeah. I think you're getting the second point I wanted to discuss a little bit, which is to my knowledge this was one of the very earliest if not really the first attempt to firmly bring theoretical computer science, complexity theory, and the analysis of algorithms to bear on a natural phenomenon. These days anybody in the field of computer science has heard of things like the computational viewpoint or the computational lens, and this sort of expansionist sense [0:40:00] that maybe computer science and theoretical computer science are the right tools for understanding a lot of natural and even social and economic phenomena. I think the PAC learning model was maybe the first sort of pure attempt from the theory community to do something like that.

So I'm wondering if you can say more about how it was received both within the theory community and within the more AI/machine learning community in the sort of years… I mean I think in hindsight of course, it succeeded wildly on both sides, but what was it like in the early days?

Valiant: Well, as far as being the first, I think my first thought is that it's probably at least the second, because I think Turing's computability was of the nature you describe. That's before Turing, people would think of computing as some sort of human phenomenon, some human operation. The idea that you could formalize it mathematically was probably regarded as absurd, and obviously its incredible success is shown by the fact that now we forget even what "computing" meant before his time and we identify the word with his definition. So obviously I thought that learning was also ripe for it.

So what was the…? Well, in the theoretical computer science community, I think there was a few people whom you know who were very good people, good scientists, and they made progress. I think it was important that there was a

group of people who worked on it. I think in theoretical computer science, I think there were no issues.

AI is a combination of different kinds of people, and as you expect, there's a spectrum of people that are more theoretical, less theoretical. I think one meets the usual spectrum you meet in any field, that humans are spread in spectrum of engineering orientation, mathematical orientation, and everywhere in between. So theories influence people depending on where they are on the spectrum.

Kearns: On that point, obviously the field of machine learning as a whole has evolved tremendously since the early '80s. As you know, it's now to the point where you can't… not only can you read about machine learning in *The New York Times*, it's hard to avoid it if you read *The New York Times* for example. Obviously there have been huge empirical strides made by that field, especially in very recent years. I'm wondering if any of that has caused you to rethink or change your views on the fundamental aspect of nature, of learning and its computational aspects. What are your thoughts on deep learning and the apparent success of learning sort of extremely complicated, difficult function classes that complexity theory might tell us in principle are hard?

[Recorder was paused here]

Kearns: So that's the next thing I wanted to ask you about, is there's obviously, since the introduction of the PAC model, been tremendous strides in the field of machine learning, both scientifically and empirically, to the point where it's difficult to avoid reading about machine learning in *The New York Times* these days. Especially in recent years with phenomena like deep learning, there have been sort of notable experimental and practical successes. I'm wondering, has any of that caused you to think differently about the nature of learning and its computational aspects or any aspects of the PAC model, particularly in regards to situations in which the classes that the PAC model might tell us are intractable to learn? You at least have specific, perhaps favorable cases where extremely complicated functions are in fact being learned well.

Valiant: Well, also in general, yes, in the situation with machine learning, the public perception has totally transformed. Even 10 years ago, if one gave a talk on machine learning, it was regarded as something a bit esoteric, something theoretical and esoteric which probably didn't work. Whereas now, as you say, everyone believes in it and talks about it.

The PAC model is about what you want to achieve when you learn and it leaves open how you achieve it and also what's the reason why you achieve it, the proof. The recent progress is clearly in the direction of algorithms, which is very

large amounts of data and they do a large amount of computation. So present evidence suggests that the best algorithms when you've got large budgets of both data and computation is different from the algorithms people looked at maybe 10 years ago, where had people had less data and less computation. There's a bigger set of algorithms people look at, but still I think what these algorithms are good at is exactly supervised learning in the statistical sense of PAC learning. Theoretically, it does open up some good theoretical questions like "Why exactly do different algorithms work?"

But on the other hand, the development is very similar to development in any area of computing where there's potentially exponential costs. The result is that people use heuristics, like in combinatorial optimization, and so use heuristics, people don't understand when they work and why they work, but they work, but you still understand the problem or what you're trying to achieve in a different way. So I think…

Okay. I mean, but my… Okay. My other thought is that's what's also a consequence of this recent success, is that just because people think, yes, machine learning is a success, supervised learning is successful and it's a reasonable goal to achieve, so the question brought up is one which I've been looking at for a long time, is "Where do you go next?" Because even though this kind of supervised PAC learning seems fundamental to cognition, it's not all of it. A theoretical question is whether there's some elegant theoretical definition of something broader. The obvious thing you want drawn up is learning with some sort of reasoning. I think this is still the first attempt at doing this, but this is still an open problem. So I hope that the success of supervised learning will shed light on a big, open problem for artificial intelligence.

Kearns: I mean this seems like a natural moment to also ask about your longstanding interest in models for the brain, which at least sort of chronologically came shortly after some of your early PAC contributions. Maybe you could tell us a little bit about sort of that progression and whether sort of PAC learning was centrally on your mind when you started thinking about the brain, or not.

Valiant: Yes. Again, the motivation for this kind of work is a question of that there must be some computational explanation of what the brain does, which seemed to be lacking. Yes, I did start this soon after I did the PAC work and I'm sure at the time I was thinking that that work would become relevant and would be key. But in fact what I found pretty fast, that things which are big, open questions for the brain are even simpler, that even if the brain does generalization in a very simple way like perceptrons, which I think is quite possible, we still don't understand very simple things about the brain as far as how information is stored, how it's added to, how it's retrieved, and simple memory. No one knows whether your memory of what you had for lunch

yesterday, whether that's stored in 10 neurons or a million neurons, and how this memory came about.

So my work in neuroscience veered off in a different direction. I'm trying to find out whether you can make models of the brain which address aspects of the computation of the brain where enough is known about the brain that models are meaningful. What this has meant is that they mainly relate to limitations in communication, that long-distance communication in the brain is very stylized, spikes going along these axons, whereas the extra local computation in neurons are probably incredibly diverse. It's known that even synapses have incredible diversity, not only neurons, so modeling each neuron, each synapse, is difficult, but exploiting the limitation of communication is something which there's some hope of doing.

Kearns: In your work on models, computational models of the brain and in particular in your book *Circuits of the Mind*, you have a couple of striking examples of cases in which obvious human capabilities, let's say in memorization, sort of conjunctive memorization of arbitrary pairs of words to attach to a new memory or concept, might imply very strong structural limitations or requirements on the actual topology of our neural networks. You give some very elegant, interesting calculations in that regard that are broadly consistent with the brain. In the time since you wrote that book, there has emerged the field of connectomics within neuroscience, which my outsider's understanding is that the basic hypothesis that perhaps very, very detailed, almost circuit diagrams of the brain might shed serious light on how computations and basic things like memory and learning take place in the brain. I'm wondering what your thoughts are on that field [0:50:00] and on that hypothesis and on what fruit it might bear and what you think about that general direction and trend in neuroscience.

Valiant: Just going back to one comment of yours, my approach is that one, we're born with some network which is very general purpose and whatever information the world throws at us, we'll be able to represent it, but it's a very general purpose machine. But for any kind of quantitative theory, including mine, what's really missing are basic numerical parameters, numbers of…the strength of synapses, something which is difficult to measure, even numbers of connections, number of connections for each neuron. I think what I'm hoping for from connectomics is that even before they get us the exact wiring diagram of the brain, which in fact may not give that much information, it'll give us statistics of the basic parameters. There's obviously some complexity in the brain, there are different layers, connection between different layers may have different properties, so the basic statistical facts are things which are essential to get any quantitative theory, and it seems that these would be very difficult to get without connectomics. So if connectomics is broadly defined and the charter[?] gets all

kinds of information, massive information of the brain, I think that'll be very important.

Kearns: Maybe let's move on to talk about your relatively recent work in evolution and evolvability in particular and evolution as a form of learning. What is the intellectual thread that took you there and to what extent did it go through learning or through the brain or by a separate route?

Valiant: It came through learning. The idea that evolution is somehow connected to learning I think is an old one. But I think what computer science has to offer is that you can ask, is evolution stronger than learning or weaker than learning, for example, or are they comparable? I think I had been thinking about this for quite a while and evolution looked like a very weak kind of learning, almost implausibly weak. I think that's kind of the big intellectual question, is exhibiting evolution as a learning phenomenon which is strong enough that we can understand how it happens. Certainly at the level of mutations in DNA, it's a very concrete phenomenon. You got mutations and it's very a concrete kind of chemical phenomenon. So understanding what the basic steps are. Mutation is some randomized process, in computer science we study randomized processes, and different ways of doing mutations would probably yield different randomized processes. Understanding which one Nature actually does would be of some interest.

In some sense, this is a detail that Darwin left out. He talked about variation in selection and now we're in a much better position to understand variation, because we understand the basic physical building blocks. What computer science has to offer is that it can describe the possible set of randomized algorithms which may be happening.

Kearns: In both the book *Probably Approximately Correct*, which is to a good extent about not just learning but about evolution as a form of learning, you start off that book and also your lecture when you received the Turing Award with some basic facts about life on Earth and the age of the Earth and sort of the constraints on evolution. Don't let me put words in your mouth, but it seems like you have a mild rebuke in some sense for sort of the literature and history of evolutionary biology. Sort of you express some incredulity that this has not… sort of how it is that the complexity of life that has evolved on Earth around us is possible in the amount of time given, sort of a very complexity theoretic view – "How did this computation take place on these machines in this amount of time?" It seems like you're surprised that this hasn't been *the* central question in many ways of evolution.

I'm wondering if you could say a little bit more about that mild rebuke and also what has been the reaction to people in evolutionary biology, maybe

not particularly about the model that you suggest but just about that sort of sense, suggestion that maybe they've been studying the wrong things?

Valiant: Well, obviously at one level, there's no dispute that evolution happened in the sense that living things are related. There's DNA evidence and the fossil evidence is overwhelming. What the question is, is whether we believe that we have a sufficient explanation. What's an explanation? In some sense, I'm not sure whether maybe some people interpret what I've said as a rebuke. It wasn't intended entirely as that. The questions I'm raising, evolution in terms of mechanism, informally of course they're ancient because I think all the early debates about evolution in Europe and around the world were all really about complexity, that it was implausible that complex things could by chance evolve. So the question I'm asking informally is ancient and the only news is that I think we've got some language in which to express the problem and maybe solve it.

Also as far as history, this issue of what was the previous theory of evolution? The theories which people regard as theories of evolution which started in the early 20th century based on statistics like population dynamics, they really were statistical theories of competition. Once you've got some things of various qualities, what happens when they compete? But in what way something is better or more fit isn't part of the model being analyzed, and this is what computer science has to offer. So I'm criticizing people who didn't think of a problem which was rather difficult to formulate.

But I think one way I've been trying to persuade people that there is a problem is that if you understand a phenomenon, even if you can't find a mathematical theoretic explanation, you should at least be able to simulate it on a computer. Almost every other science does computer simulations to demonstrate that they understand what they're doing and in evolution, it's elusive. There may be various excuses for it being elusive, like "We don't know the conditions under which evolution evolved." So this difficulty of not having good simulations isn't an absolute argument, but it's something one should think about.

Kearns: Do you feel that… I mean since broadly speaking we don't really understand either evolution at the granularity that you're suggesting we need to or the brain and you're sort of proposing that one route might be simulations that are broadly consistent with the facts we see around us, do you feel that there's been more success in simulation approaches in neuroscience than there has been in evolution? I know I'm comparing apples and oranges, but just in your view.

Valiant: I don't think so. No, no.

Kearns: Yeah.

Valiant: No.

Kearns: And of course in both of those fields, I mean they have theoretical components, but they're also largely empirical fields. I mean in the case of evolution, it's difficult to run experiments but you can ask about consistency with the fossil record or other historical data. In neuroscience, you can run certain types of experiments and certainly gather large amounts of passive data. What do you think it'll take to kind of bridge the gap between the detailed computational theories you're suggesting and the empirical branches of those fields, to the extent that concrete experiments with outcomes predicted and either verified or refuted might arise from your theories [1:00:00] or similar theories and actually be tested in some sense?

Valiant: I think in both areas, the issue isn't that there's a single experiment which will resolve everything, but that if you ask questions from a… if you have some computational theory and then with systematic experiments, you could verify whether it's a plausible one. In neuroscience, I know my approach does suggest various experiments which are very difficult to do at the moment but potentially are easier and very, very broadly. Traditionally people think … a traditional model is something like Hebb rule, which is that if you've got a single neuron, what happens if you stimulate it in this and that way? Whereas it's likely that in the brain, it's much more kind of a systems-level phenomenon and what you want to find out is some systems-level primitives, a bit like the Hebb rule, which you want to verify experimentally. So you need lots of probes testing what's happening and verifying whether like a systems-level Hebb rule is possible on arbitrary sets of neurons.

So if you've got a theory of what algorithms are run on the brain and if you can do invasive-enough experiments, then there's no reason why you shouldn't verify them. And I think these things are on the horizon of being feasible.

Kearns: And what about in evolution?

Valiant: Well, in evolution, I think we're clearly… what's been done already, is in understanding mutations, understanding binding, understanding why if you do a single mutation in DNA, what are the effects on the binding of various proteins. Is it possible that the space or binding strength is such that it's smooth enough or whatever that some algorithm can run to make things better? At that level, there's lots you can do experimentally.

Kearns: So Les, I just thought I would close with a very open-ended, broad question, which is you've had a remarkable career in which you've both made very, very core, fundamental contributions to computer science and just the nature of computation itself, but you've also tackled really areas of everyday interest to ordinary people like the brain or evolution or learning. I'm wondering if there is another such very, very large topic out

there that in your view has some sort of natural phenomenon that either requires or would be amenable to computational viewpoints and algorithmic analysis and sort of the study of how the laws of computation apply to it.

Valiant: Yeah. All these things for me are rather related, so they're not so distinct questions. Yes, I think as far as asking what other natural phenomena are amenable to computational approaches, I think that's the right question. As far as what I spend time on, I think the only other one I would mention is one I already mentioned — you mentioned learning, the brain and evolution — is what I say is cognition. In what way you can extend learning to kind of AI-ish theory of what you need to achieve or what is being achieved by cognition. I separate… Neuroscience is a different thing, because I think there you're trying to explain more simpler, low-level things. The question is whether beyond learning there's something fundamental to cognition or whether cognition is just learning with various heuristics added. We don't know. I think that's a good question.

Kearns: Interesting. Well, thank you for your time and congratulations on an astounding career. Thank you.

Valiant: Thank you.

[1:04:07]
[end of recording]