DOI:10.1145/1646353.1646369

Dennis Shasha

Interview

An Interview with Michael Rabin

Michael O. Rabin, co-recipient of the 1976 ACM A.M. Turing Award, discusses his innovative algorithmic work with Dennis Shasha.

I first encountered Michael Rabin in 1980 when I was a first-year Ph.D. student at Harvard University. It was four years after Rabin and Dana Scott won the ACM A.M. Turing award for their foundational work on deterministic and nondeterministic finite state automata. By 1980, however, Rabin's interests were all about randomized algorithms. His algorithms course was a challenge to all of us. Sometimes he presented a result he had worked out only two weeks earlier. When I had difficulty understanding, I would make puzzles for myself. As a consequence, when I published my first puzzle book *The Puzzling Adventures of Dr. Ecco*, I dedicated the book to Michael as one of my three most influential teachers. At Harvard, I enjoyed every encounter with Michael—in seminars and at parties. He was a great raconteur and a great joker. In 1994, journalist Cathy Lazere and I embarked on the writing of *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists*. The goal was to interview great living seminal thinkers of our field: Knuth, Rabin, Dijkstra, among others. Each thinker was associated with an epithet: Michael's was "the possibilities of chance."

—*Dennis Shasha,*
Courant Institute, New York University,

PHOTOGRAPH COURTESY OF MICHAEL RABIN

Shasha: I'm going to try to get to a mixture of personal and technical recollections, let's start in Israel. You finished high school, and what were you thinking about doing after high school?

Rabin: I've been intensely interested in mathematics ever since I was 11 years old, when I was kicked out of class and told to stand in the hall. I encountered two ninth-graders who were working on geometry problems and asked them what they were doing. They said they had to prove such-and-such a statement, and then condescendingly said to me, "You think you can do it?" Even though I never studied geometry, I was able to solve that problem. The fact that you can by pure thought prove statements about the real world of lines, circles, and distances impressed me so deeply that I decided I wanted to study mathematics.

My father sent me to the best high school in Haifa, where I grew up, and in 10th grade I had the good luck to encounter a real mathematician, Elisha Netanyahu (the uncle of Benjamin Netanyahu) who was at the time a high school teacher. Later he became a professor at the Haifa Institute of Technology. He conducted once a week a so-called "mathematical circle," where he taught a selected group of students number theory, combinatorics, and advanced algebra. Netanyahu lent me advanced mathematics books. Starting at age 15, I read Hardy and Wright's book *An Introduction to the Theory of Numbers*, the first volume in German of Landau's *Zahlentheorie*. I studied a wonderful book by G.H. Hardy called *Pure Mathematics*, which was mainly analyses, two volumes of Knopp's *Functions of a Complex Variable*, A. Speiser's book *Gruppentheorie*, and so on.

I finished high school at age 16½ because the war of independence broke out in Israel, and everybody in my class was drafted into the army. I continued to study mathematics on my own while in the army. Then I got in touch with Abraham Fraenkel, who was a professor of mathematics in Jerusalem and whose book on set theory I had studied. That was maybe in September of 1949. Fraenkel met and tested me on group theory, number theory, set theory, and decided that it would be quite worthwhile if I came to the university. He got in touch with army authorities, and I was discharged from the army to go and study at the university.

The system was that you went directly to a master's degree. I studied a lot of mathematics: set theory, algebra, functions of a complex variable, linear spaces, differential equations, probability as well as physics and applied mathematics. I was mainly interested in algebra. So in 1952 I wrote a master's thesis on the algebra of commutative rings, and solved an open problem due to Emmy Noether, giving a necessary and sufficient condition on a commutative ring for having the property that every ideal is a finite intersection of primary ideals. That was one of my first papers. It appeared in the *Comptes Rendus* (the proceedings) of the French Academy of Sciences.

Princeton Days and String Automata Theory

I went to Princeton University to study with Alonzo Church. At the time, only 13 Ph.D. students were admitted every year to the Mathematics Department. I found out the committee that was considering me assigned somebody to look at the master's thesis paper, and this is what led to my admission. My Ph.D. thesis was on computational (recursive) unsolvability of group theoretical problems, thus combining my interests in algebra and computability.

In 1957, IBM decided to go into research in a big way. They sent people to recruit promising students from top universities and top mathematics departments. Dana Scott and I were invited to spend the summer at IBM Research. Watson Labs did not exist at that time, and research was located at the Lamb Estate in Westchester County. This had previously been an insane asylum where, rumor had it, rich families would forcibly confine troublesome members.

At the Lamb Estate, Dana and I wrote the paper "Finite Automata and Their Decision Problems." Automata theory had started with a study by Walter Pitts and Warren McCulloch of what would now be called neural networks. They assumed that those neural networks essentially embodied a finite number of states, and talked about what is recognizable by such finite state neural networks, but they didn't have a complete characterization. Later, in 1956, S.C. Kleene invented regular languages: sets of strings that are obtained from finite sets by certain simple operations. Kleene showed that finite neural nets compute or recognize exactly the regular languages.

We decided to completely abstract away from neural nets and consider a finite-state machine that gets symbol inputs and undergoes state transitions. If upon being fed a string of symbols it passes from an initial state to one of a number of accepting states, then the string is accepted by the finite-state machine. The finite-state machine had no output except for saying yes/no at the end.

I set myself the goal of finding more and more applications of randomization in mathematics and computer science.

Dana and I asked ourselves in what ways could we expand the finite-state machine model. One of our extensions was to consider nondeterministic machines. We really did not have any deep philosophical reason for considering nondeterminism, even though as we now know nondeterminism is at the center of the P = NP question, a problem of immense practical and theoretical importance. For us, it was just one of the variants. We stipulated that the automaton when in state S and upon input symbol sigma can go into any one of a number of states S', S",..., comprising a certain subset of the set of states. The nondeterministic finite-state machine accepts a string if there is a possible computation, a possible sequence of transitions, leading to an accepting state. Then we proved that finite-state nondeterministic automata are exactly equal in power to deterministic automaton computations. [On a practical level, this means that the wildcard searches of say grep, perl, or python can be expressed as nondeterministic finite state automata and then can be translated into deterministic finite state automata.]

The corresponding question whether nondeterministic polynomial time Turing machine computations are equal in power to deterministic polynomial time Turing machine computations is the famous P = NP problem.

Employing nondeterministic automata, we were able to re-prove Kleene's result that finite state machines exactly accept regular languages. The proof became very easy. We also introduced and studied other extensions of finite automata such as two-way automata, multi-tape and linearly bounded automata. The latter construct appeared in our research report but did not find its way into the published paper.

Origins of Complexity Theory

The next summer I again went to the Lamb Estate. At that time there was a methodologically misled widely held view that computing, and what later became computer science, was a sub-field of information theory in the Shannon sense. This was really ill-conceived because Shannon was dealing with the information content of messages. If you perform a lengthy calculation on a small input then the information content in the Shannon sense of the outcome is still small. You can take a 100-bit number and raise it to the power 100, so you get a 10,000-bit number. But the information content of those 10,000 bits is no larger than that of the original 100 bits.

John McCarthy posed to me a puzzle about spies, guards, and password. Spies must present, upon returning from enemy territory, some kind of secret password to avoid being shot by their own border guards. But the guards cannot be trusted to keep a secret. So, if you give them the password, the enemy may learn the password and safely send over his own spies. Here is a solution. You randomly create, say, a 100-digit number x and square it, and give the guards the middle 100 digits of x2. John von Neumann had suggested the middle square function for generating pseudo-random numbers. You give the number x to the spy. Upon being challenged, the spy presents x. The guard computes x2 and compares the middle square to the value he has. Every password x is used only once. The whole point is that presumably it is easy to calculate the middle square, but it is difficult, given the middle square, to find one of the numbers having that value as a middle square. So even if the guards divulge the middle square, nobody else can figure out the number the spy knows.

But how do you even define that difficulty of computing? And even more so, how do you prove it? I then set myself to study that problem. I wrote an article called "Degree of Difficulty of Computing a Function and Hierarchy of Recursive Sets." In that article, I wasn't able to solve the problem of showing that the von Neumann function is difficult to invert. This is really a special case of the P = NP problem. It hasn't been settled to this day. But I was able to show that for every computable function there exists another computable function that is more difficult to compute than the first one, regardless of the algorithm or programming language one chooses. It's similar to the minimum energy required for performing a physical task. If this phone is on the floor and I have to raise it, there is a minimum amount of work. I can do it by pulling it up, by putting a small amount of explosive, blowing it up here, but there is a certain inherent amount of work. This is what I was studying for computations.

The next significant application of my work on randomization was to cryptography.

I think this paper, no less than the Rabin/Scott paper, was a reason for my Turing Award. The ACM announcement of the Turing Award for Dana and for me mentioned the work on finite automata and other work we did and also suggested that I was the first person to study what is now called complexity of computations.

Randomized Algorithms: A New Departure

I went back to Jerusalem. I divided my research between working on logic, mainly model theory, and working on the foundations of what is now computer science. I was an associate professor and the head of the Institute of Mathematics at 29 years old and a full professor by 33, but that was completely on the merit of my work in logic and in algebra. There was absolutely no appreciation of the work on the issues of computing. Mathematicians did not recognize the emerging new field.

In 1960, I was invited by E.F. Moore to work at Bell Labs, where I introduced the construct of probabilistic automata. These are automata that employ coin tosses in order to decide which state transitions to take. I showed examples of regular languages that required a very large number of states, but for which you get an exponential reduction of the number of states if you go over to probabilistic automata.

Shasha: And with some kind of error bound?

Rabin: Yes, yes, that's right. In other words, you get the answer, but depending upon how many times you run the probabilistic automaton, you have a very small probability of error. That paper eventually got published in 1963 in *Information and Control*.

In 1975, I finished my tenure as Rector (academic head) of the Hebrew University of Jerusalem and came to MIT as a visiting professor. Gary Miller was there and had his polynomial time test for primality based on the extended Riemann hypothesis. [Given an integer *n*, a test for primality determines whether *n* is prime.] That test was deterministic, but it depended on an unproven assumption. With the idea of using probability and allowing the possibility of error, I took his test and made it into what's now called a randomized algorithm, which today is the most efficient test for primality. I published first, but found out that Robert Solovay and Volker Strassen were somewhat ahead with a different test. My test is about eight times faster than theirs and is what is now universally being used. In the paper I also introduced the distinction between what are now called Monte-Carlo and Las-Vegas algorithms.

In early 1976 I was invited by Joe Traub for a meeting at CMU and gave a talk presenting the primality test. After I gave that lecture, people were standing around me, and saying, "This is really very beautiful, but the new idea of doing something with a probability of error, however exponentially small, is very specialized. This business of witnesses for compostiness you have introduced is only useful for the primality test. It will never really be a widely used method." Only Joe Traub said, "No, no, this is revolutionary, and it's going to become very important."

From Trick to Fundamental Technique

From then on, I set myself the goal of finding more and more applications of randomization in mathematics and computer science. For example, in 1977 in my MIT class, I presented an algorithm for efficiently expressing an integer as the sum of four squares. Lagrange had proved in 1770 that every integer can be so expressed, but there was no efficient algorithm for doing it. In 1977 I found an efficient randomized algorithm for doing that computation. That algorithm later appeared in a joint paper with Jeff Shallit in 1986 together with additional applications of randomization to number theoretical algorithms. Later, I turned my attention to distributed algorithms and found an approach using a random shared bit for solving Byzantine Agreement far more efficiently than previous approaches. Still later I applied randomization to asynchronous fault-tolerant parallel computations in collaboration with Jonatan Aumann, Zvi Kedem, and Krishna Palem.

Right now, if you look at STOC and FOCS [the major conferences in theoretical computer science] maybe a third to half of the papers are built around randomized algorithms. And of course you've got the wonderful book *Randomized Algorithms* by Rajeev Motwani and Prabhaker Raghavan.

Shasha: Let me back up and talk a little bit about the general uses of randomness in computer science. There seem to be at least three streams of use of randomness—yours, the use in communication (for example, exponential back off in the Ethernet protocol), and the use in genetic algorithms where random mutations and random recombination sometimes lead to good solutions of combinatorial problems. Do you see any unified theme among all those three?

Rabin: I would say the following: The use in the Ethernet protocol is in some sense like the use in Byzantine agreement. In Byzantine agreement, the parties want to reach agreement against improper or malicious opponents. In the Ethernet protocol, the participants want to avoid clashes, conflicts. That's somewhat similar. I don't know enough about genetic algorithms, but they are of the same nature as the general randomized algorithms. I must admit that after many years of work in this area, the efficacy of randomness for so many algorithmic problems is absolutely mysterious to me. It is efficient, it works; but why and how is absolutely mysterious.

More recently, I have become interested in protecting the privacy and secrecy of auctions.

It is also mysterious in another way because we cannot really prove that any process, even let's say radioactive decay, is truly random. Einstein rejected the basic tenets of Quantum Theory and said that God does not play dice with the universe. Randomized algorithms, in their pure form, must use a physical source of randomness. So it is cooperation between us as computer scientists and nature as a source of randomness. This is really quite unique and touches on deep questions in physics and philosophy.

Tree Automata

Let me return to a chapter of my work that I skipped before. After the work on finite automata by Dana Scott and me, two mathematicians, Richard Buchi and Calvin Elgot, discovered how the theory of finite automata could be used to solve decision problems in mathematical logic. They showed that the so-called Pressburger arithmetic decision problem could be solved by use of finite automata. Then Buchi went on to generalize finite automata on finite sequences to finite automata on infinite sequences, a very brilliant piece of work that he presented at the Congress on Logic, Philosophy, and Methodology in Science in 1960. In that paper, he showed that the so-called monadic second-order theory of one successor function is decidable. Let me explain very briefly what that means.

We have the integers, 0, 1, 2, 3. The successor function is defined as $S(x) = x+1$. The classical decision problems pertain to problems formulated within a predicate logic—the logic of relations and functions with the quantifiers of "exists" x and "for all" x, and the logical connectives. Monadic second-order theory means that you quantify over sets. Buchi demonstrated that the monadic second-order theory of one successor function—where you are allowed to quantify over arbitrary subsets of integers—is decidable. His was the first result of that nature.

Buchi posed an open problem: is the monadic second-order theory decidable if you have two successor functions. For example, one successor function being 2x (double), and the other one being 2x+1. I don't know if he realized how powerful the monadic theory of two successor functions would turn out to be. I realized that this is an incredibly powerful theory. If you find a decision procedure for that theory, then many other logical theories can be demonstrated to be decidable.

I set myself to work on this problem. My formulation was a generalization from infinite strings to infinite binary trees. You consider a tree where you have a root, and the root has two children, a child on the left and a right child, and each of these has again two children, a left child and a right child. The tree branches out ad infinitum, forming an infinite binary tree. Consider that tree with the two successor functions, left child, right child, and study the logical theory of the tree with these two functions and quantification over arbitrary subsets of nodes of the tree.

In 1966, I came as visitor to IBM research at Yorktown Heights, and one of my goals to find an appropriate theory of automata on infinite binary trees and prove the decidability of the same problems that Dana Scott and I had shown to be decidable for finite automata on finite strings. I created the appropriate theory of automata on these infinite trees and showed that it was decidable. I consider this to be the most difficult research I have ever done.

A remarkable feature of that original proof is that even though we are dealing with finite automata, and with trees that are infinite but countable, the proof and all subsequent variants employ transfinite induction up to the first uncountable ordinal. Thus the proof is a strange marriage between the finite and countable with the uncountable.

This theory led to decision algorithms for many logical theories. That included decidability of nonstandard logics like modal logics and the temporal logics that are a tool for program verification, especially in the work of Amir Pnueli, Moshe Vardi, Orna Kupferman, and many others.

Great teaching and great science really flow together and are not mutually contradictory or exclusive of each other.

Keeping Secrets

The next significant application of my work on randomization was to cryptography. Ueli Mauer suggested a model of a cryptographic system that is based on what he called the bounded storage assumption. Namely, you have a very intense public source of random bits that everybody can observe, say beamed down from a satellite. The sender and receiver have a short common key that they establish say by meeting, and they use that key in order to select the same random bits out of the public source of randomness. Out of those bits, they construct so-called one-time pads. If you assume that the intensity of the random source is so large that no adversary can store more than, let us say, two-thirds of its bits, then the one-time pad is really completely random to the adversary and can be used for provably unbreakable encryption.

Mauer initially proved the unbreakability result under the assumption that the adversary stores original bits from the random source. However, there remained an open question: suppose your adversary could do some operations on the original bits and then store fewer bits than the number of source bits. Jonatan Aumann, Yan Zong Ding, and I showed that even if the adversary is computationally unbounded, one can still obtain unbreakable codes provided the adversary cannot store all his computed bits. This work created quite a stir and was widely discussed, even in the popular press.

Nowadays however, the bounded storage assumption may not be compelling because the capacity of storage has increased so dramatically. So I posited a new Limited Access Model. Suppose each of 10,000 participants independently store physically random pages. The sender and receiver use a common small random key to randomly select the same 30 page server nodes and from each node download the same randomly selected page. Sender and receiver now XOR those 30 pages to create a one-time pad they employ to encrypt messages. Assume that an adversary cannot listen to or subvert more than say 2,000 of the 10,000 page server nodes. Consequently, his probability of having obtained any particular random page is no more than 1/5, and, since the pages are XORed in groups of 30, his probability of having all of those pages is 1/5 to the power of 30. If the adversary is missing even one of those pages, then the one-time pad is completely random with respect to him, and consequently, if used to encrypt by XORing with the message, the encrypted message is also completely random for that adversary.

Privacy for Pirates and Bidders

In the late 1990s, [Dennis Shasha] came to me and suggested that we work together on devising a system for preventing piracy, to begin with, of software, but later on also music, videos, and so on—any kind of digitized intellectual property. We started by reinventing variants of existing methods, such as the use of safe co-processors and other methods that were actually current at the time and which, by the way, have all either been defeated or impose excessive constraints on use. We then invented a new solution. Our design protects privacy of legitimate purchasers and even of pirates while at the same time preventing piracy of protected digitized content. For example, an engineering firm wanting to win a contest to build a bridge can purchase software for bridge design without identifying itself. Thus competitors cannot find out that the firm is competing for the project. The purchaser of digital content obtains a tag permitting use of the content. The tag is not tied to a machine identifier so that the tag and use of the content can be moved from machine to machine. Coupled with robust content-identifying software, use of protected content absent a corresponding tag is stopped. The deployment of this robust solution to the piracy problem requires the cooperation of system vendors.

More recently, I have become interested in protecting the privacy and secrecy of auctions. Working with Stuart Shieber, David Parks, and Chris Thorpe, we have a methodology that employs cryptography to ensure honest privacy-preserving auctions. Our protocols enable an auctioneer to conduct the auction in a way that is clearly honest, prevents various subversions of the auction process itself, and later on, when the auction is completed and the auctioneer has determined the winner or winners and how much they pay and how much they get of whatever is being auctioned in multi-item auctions, the auctioneer can publish a privacy-preserving proof for the correctness of the result. In the initial papers, we used the tool of homomorphic encryption. Later on, I had an idea that I then eventually implemented with Chris Thorpe and Rocco Servedio of an entirely new approach to zero knowledge proofs, which is computationally very efficient, does not use heavy-handed and computationally expensive encryption, and achieves everything very efficiently by use of just computationally efficient hash functions.

Teachers, Teaching, and Research

In the life of every creative scientist, you will find outstanding teachers that have influenced him or her and directly or indirectly played a role in their success. My first mentor was a very eminent logician and set theorist, Abraham Halevi Fraenkel. I wrote a master's thesis under the direction of the wonderful algebraist Jacob Levitski. He was a student of maybe the greatest woman mathematician ever, Emmy Noether, the creator of modern abstract algebra as we know it. In Jerusalem, I learned applied mathematics from Menachem Schiffer, a great mathematician and a great teacher.

Powerful algorithms are enabling tools for every computer innovation and application.

Shasha: Let's talk a little bit about the relationship that you've mentioned to me often between teaching and research, because you've won many awards for teaching as well as for science. How do you think academics should view their teaching responsibilities?

Rabin: This is really a very important question. There is this misconception that there is a conflict and maybe even a contradiction between great teaching and being able to do great science. I think this is completely incorrect, and that wonderful teaching, such as Schiffer's teaching, flows from a deep understanding of the subject matter. This is what enables the person also to correctly select what to teach. After all, the world of knowledge, even in specialized subjects, is almost infinite. So one great contribution is to select the topics, and the other great contribution is to really understand the essence, the main motifs, of each particular topic that the person presents, and to show it to the class in a way that the class gets these essential ideas. Great teaching and great science really flow together and are not mutually contradictory or exclusive of each other.

The Future

Shasha: You have contributed to so many areas of computer science. Talk about one or many, and where do you think it is going?

Rabin: Computer science research has undergone a number of evolutions during my research career, and because of the quick pace, one can even say revolutions. To begin with, there was Alan Turing's model of a computing machine, leading to the stored-program computer. Next there was a great emphasis on the study of various mathematical machines. Finite automata are models for sequential circuits. Nondeterministic and deterministic so-called pushdown automata play a pivotal role in the syntactic compilation of programming languages. For about 20 years or so, there was a great emphasis in research on automata, programming languages, and formal languages, also in connection of course with linguistics. There was also a considerable emphasis on efficient algorithms of various kinds. The book by Alfred Aho, John Hopcroft, and Jeff Ullman, and Donald Knuth' classical books are examples of this strong interest in algorithms. The study of algorithms will always remain centrally important. Powerful algorithms are enabling tools for every computer innovation and application.

Then emphasis started to shift toward issues of networking and communication. Even the people who created the Internet, email, and other forms of connectivity perhaps didn't fully realize where all that was going to lead. This evolution, revolution, explosion, started to accelerate, and over the past 10 years we went into a phase where the main use of computers is in the world of information creation, sharing, and dissemination. We have search engines, Wikipedia, Facebook, blogs, and many other information resources, available to everyone literally at their fingertips.

We are only at the beginning of this revolution. One can predict that there is going to be an all-encompassing worldwide network of knowledge where people are going to create, share, and use information in ways that never existed before, and which we can't fully foresee now. These developments are giving rise to a torrent of research in information organization and retrieval, in machine learning, in computerized language understanding, in image understanding, in cryptography security and privacy protection, in multi-agent systems, to name just a few fields. Computer science research will continue to be rich, dynamic, exciting, and centrally important for decades to come.