Raf Bocklandt
Korteweg-de Vries Institute for Mathematics
University of Amsterdam
r.r.j.bocklandt@uva.nl

Interview Abel Prize laureate 2021 Avi Wigderson

A look at mathematics through the lens of computation

Last January the KNAW and CWI planned to organize the 'Evening of the Abel prize' to celebrate the work of the two laureates of 2021, László Lovász and Avi Wigderson. Unfortunately, due to the arrival of a new corona variant, the event has been postponed to the spring with the hope that no further mutations will spoil the fun. In the meantime, as an appetizer, Nieuw Archief arranged an online transatlantic interview with Avi Wigderson.

Growing up in Haifa

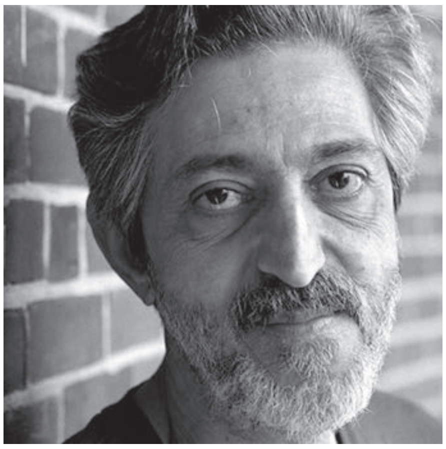
Born in 1956, Avi Wigderson grew up in Haifa, a major port in the North of Israel. He liked playing football and going to the beach with his friends, but already from a young age he developed an interest in solving mathematical problems.

"My father was an engineer who loved puzzles and riddles, and I think he transferred this love to me. His job consisted of fixing electrical problems in engines of ships for the navy. If something would stop working, he would think of it as a puzzle and he would tell me about the different diagnostic tests he would do to locate the problem. He also loved asking me riddles he got from old Russian books, which was infectious at least to me. My two brothers were interested in other things. One is a year younger than me. He was always interested in nature and he ended up as a biologist. My other brother was about six years younger. He was actually interested in engineering, so he liked it when my dad showed us how to fix things. I had no interest in the workings of physical appliances, like a washing machine or a car, but my youngest brother went for it and he became an engineer.

I went to a very good high school, the Hebrew Reali School, it was the only sort of semi-private school in Haifa and it had some very good and inspiring teachers. It also produced some other renowned mathematicians and computer scientists, including Michael Rabin and Noga Alon."

After high school Wigderson wanted to go to University but in Israel you have to go to the army first, which takes at least three years.

"The first year in the army I was in the flying course to be a pilot, but to my great joy I flunked because otherwise I would have had to serve seven years instead of three. The remaining two years I spent in a reasonably interesting office job. Nowadays, with the rise of cybersecurity and cryptography, there are a lot more possibilities to do some intellectually challenging tasks in the army than in my time. Many Israeli researchers in mathematics and computer science that you might have heard of did do interesting things."



Avi Wigderson

After his army service Wigderson returned to his hometown Haifa and there he enrolled in Technion, the Israel institute of technology. Founded in 1912, it is the oldest university of the country and it is sometimes referred to as the MIT of Israel. Despite his early interest in mathematics, he opted for a degree in computer science.

"I was interested in mathematics but my parents, who were holocaust survivors, thought that I should have a good job. They convinced me to do computer science, where I would learn some math anyway. I was not from an academic family, so the idea of becoming a university professor was never in the consciousness of our family. I didn't know about this, even when I was in college I didn't realize that my teachers do much more than teaching.

The computer science department was a rather new department. Throughout the country many mathematics departments offered computer science programs already from the sixties, sometimes they were also offered in electrical engineering, but this probably was the first independent computer science department. We were a small class of maybe forty people. Most of our teachers were already researchers who did something in computer science, such as

A view over the port of Haifa

operating systems, programming languages, databases and of course algorithms.

It was not just a theory program, we did a lot of practical things like programming classes, which in these days was still done using punch cards. Sometimes that could be very frustrating: I once wrote a program which consisted of a batch of maybe three hundred punch cards and just before I wanted to put it in the stacker, I dropped it."

From the East to the West Coast

During his studies at Technion, Wigderson met his wife Edna, a mathematics student. They got married in their final year of college and after their degree in Israel they wanted to pursue further studies in the United States. Avi ended up doing a PhD in computer science at Princeton, while Edna took a master in mathematics at Rutgers University.

"I applied to about ten universities and I was accepted by Princeton and Yale. My mentor at Technion, Shimon Even, said that Princeton is a nicer place to live, so we went to Princeton. My advisor Dick Lipton was interested in practically everything theoretical. He was interested in cryptography, complexity, algorithms, and in a variety of models of computation. I just followed him and I learned everything he was doing. I also collaborated quite a bit with my other you know classmates from graduate school. I wrote papers with classmates on different subjects and my thesis was just a collection of a few papers. It was not really a cohesive topic but my advisor was happy with it, so I was happy too. In the meantime my wife and I also had our first child, which is maybe a more impressive production than a PhD."

The young family stayed in the United States for three more years but moved to the other side of the country.

"I had three years of postdocs, two of them were in Berkeley and one at the IBM research institute in San Jose. You shouldn't think of IBM just as a company fabricating computers and software. Its research department did real pure theoretical research. At that time, the mid eighties, this institute functioned exactly like any other computer science or math department. There was a lot of freedom: nobody told anybody what to do, the staff members were top-notch theorists: Miklós Ajtai, Ron Fagin, Nicholas Pippenger, Larry Stockmeyer, … It was a phenomenal place to do a postdoc and I learned a lot there. Needless to say, my two years at Berkeley (one actually at MSRI) were fantastic for my professional development as well."

During that time he also met László Lovász, with whom he would later share the Abel Prize in 2021. The committee awarded the prize "For their foundational contributions to theoretical computer science and discrete mathematics, and their leading role in shaping them into central fields of modern mathematics".

"Combinatorics, discrete math and computer science are closely related as you know from the Abel prize. I'm very happy to have received the prize together with Laci, not only because it made a lot of sense from a mathematical point of view, but also because he is one of my heroes and a longtime friend. We spent some time in the same places. We were in Berkeley together for a year in the eighties and we were in Princeton for a year in the nineties. During these times we talked a lot and we wrote a couple of papers together."

Proofs without knowledge

One of the lines of research that Wigderson pursued in the eighties and nineties was interactive proofs, and in particular the idea of zero-knowledge proofs. This concept formalizes the problem that arises when you want to convince someone that you have a method to do something, without revealing any information about the way it works or giving away the answer to the problem.

"The idea of proving things without providing any knowledge is something paradoxical and therefore very fascinating. I cannot imagine it will not peak anybody's interest once they hear about it. The proposal was made in 1985 by Shafi Goldwasser, Silvio Micali and Charles Rackoff [7]. The motivation they had came from cryptography. It's very natural in cryptographic settings that you would like to do something that depends on a secret, and the other party wants to make sure that you did things correctly without cheating. On the other hand, you don't want to show them how you did it because part of the input is your secret.

The most basic example, one that is done daily millions of times, is when people pick a public key. This often means that you pick two primes, multiply them together and then publish the result. If I give you such a number, why should you believe me that it's a product of two primes? Maybe it's a product of three primes, maybe I didn't do it correctly. How do I convince you? Well I can give you the factors but that would reveal my secret.

Avi and Laci, Oslo 2012

That was their question: is it possible for me to convince you that I have a proof of something — and here the proof is the two prime factors — without showing you the proof. Indeed, even without giving you any information whatsoever. For another example: how do I convince you that I have a proof of the Riemann hypothesis without showing you anything that you don't already know."

While the case of the Riemann hypothesis might be out of reach for this article, we can give an example of a zero-knowledge proof inspired by the problem of factoring numbers. Let us assume Peggy (the prover) knows that a number $n = pq$ is the product of two big primes p and q. From basic number theory we know that this means she can invert the function $x \to x^e \bmod n$ for any e which is coprime with $(p-1)(q-1)$ by calculating $d = e^{-1} \bmod (p-1)(q-1)$ and taking the d-th power of $x^e \bmod n$. While finding d for a given e, is not the same task as factoring n it is hypothesized to be computationally equivalent. (This is called the RSA assumption.)

Suppose that Peggy wants to convince Victor (the verifier) that she is indeed able to invert $x \to x^e \bmod n$ without revealing the primes p, q or d, she could allow Victor to pick random x's and send her $y = x^e \bmod n$ while she returns $x = y^d \bmod n$. If this all reminds you strongly of the RSA algorithm for public key cryptography, this is not a coincidence. (To turn this into a true zero knowledge proof, one has to be a bit more careful to prevent Victor from choosing very specific x's that might reveal some information about p, q, d.)

"The mid-eighties was a period where there was an explosion in cryptography and the theory of cryptography was full of challenges of this type. There were lots of tasks that people came up with that look impossible to do, like public key encryption, how to sign a document digitally without anybody being able to forge it, or how to exchange a secret in the presence of untrusted parties. Even examples like: how do you play poker over the telephone while still following the rules, were studied. Not only did these challenges come up, but actually it turned out that they were possible in the world of cryptography, once you have a function that is easy to compute but difficult to invert, like multiplication (whose 'inverse' is factoring).

In my opinion the zero-knowledge challenge was maybe the best challenge ever because it seems totally contradictory to any human experience about what convincing is. I mean, how can I convince you of something (which you may not believe) without telling you anything that you don't already know. But it turns out to be not only possible, it's even universal. Everything that is provable at all can be proved in this way. When the idea of zero-knowledge proofs was introduced, everyone was talking about it. It was a big thing because it would be extremely useful for cryptographic protocols, but it was not clear to which extent it was possible.

At that time most of the developments in cryptography were intellectual games. It was just a theory like in normal mathematics: you have some axioms and you're trying to see what you can deduce from them. In this case: which kind of tasks can you perform privately and securely, assuming you have so-called one-way, or trap-door functions, like multiplying integers? The protocols developed for many of these tasks were theoretically efficient but it was not clear that they were directly implementable. Nobody bothered about this and for a good reason: I always believe that practice will follow theory if it's good. It's not that you have to start by taking into consideration every practical detail. Instead you first want to understand the broad picture and develop general techniques before you adapt them to specific situations. Today we all experience the great practical consequences of this theory to electronic commerce, Internet security, et cetera."

Crossing boundaries

In 1986 Wigderson went back to his home country to take up a position at the Hebrew university in Jerusalem at the computer science department. He stayed in Israel for more than a decade before returning to Princeton as a professor at the Institute for Advanced Study, a position he still holds today.

"The computer science department at Hebrew university used to be a part of the mathematics department. The physical separation happened while I was a chair, but even after the split in two, the departments remained very close. Naturally, I had most contact with the combinatorialists such as Nati Linial, Gil Kalai and other computer science theorists. All the courses I taught were open for both mathematics and computer science students, especially the graduate courses. Students of mathematics came to my courses and some of my graduate students actually were math students, so we were one big family."

Just like his teaching and supervising, Wigderson's own research is also on the boundary between mathematics and computer science. A lot of the problems that appear in complexity theory have their origin in abstract mathematics, most of which he did not encounter in his undergraduate studies.

“Most of the advanced math I needed I did not learn in an organized way. I think that’s a disadvantage. For a person working in theoretical computer science, especially today, it’s much better to get to know lots of basic math courses in school rather than later on. Since my education was not like that — I just learned the usual basics such as calculus and linear algebra — I had to learn the rest by myself, sometimes from books, sometimes from my colleagues.”

Coming from a different field often offers a fresh perspective on the material and allows one to approach problems from a different angle and ask questions that others might overlook or neglect. This can then lead to surprising new results.

“Part of the gut instinct or the DNA of a computer scientist is — not only in math but also in biology and physics — that when you face a theorem or a phenomenon, you want to know what is the process that makes it happen and how efficiently does it do that. Take for example Hilbert’s Nullstellensatz. It tells you that if some polynomials do not share a common zero then you can find a linear combination that gives you 1. As a computer scientist you immediately wonder what are these polynomial coefficients, how can I find them efficiently, and what is the complexity of doing this.

This is a fundamental question: an existence theorem of some object is nice but how can I generate it? I think this general question has been extremely fruitful and productive. There are many stories from research that illustrate this.”

Ramsey and randomness

A prime example of this comes from Ramsey theory. This branch of mathematics studies how certain nice substructures necessarily appear once a given structure gets large enough. In the case of graphs one can show that for any k there is a number $R(k,k)$ such that any graph with $n \geq R(k,k)$ nodes contains a complete subgraph of size k (a clique) or an independent set of that size (a set of nodes without any edges between them). Determining these numbers is a notoriously hard problem.

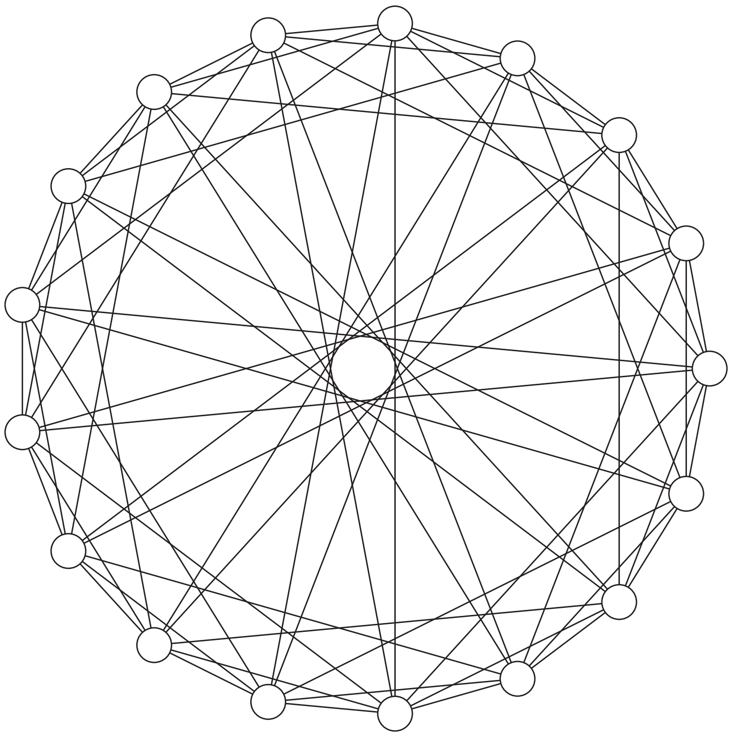
In the forties Paul Erdős [5] gave a lower bound for n using a simple probabilistic argument. If you draw a random graph with n nodes then for every choice of k vertices the probability that it is complete or independent is $2 \cdot 2^{-\binom{k}{2}}$, so if $\binom{n}{k} 2^{1-\binom{k}{2}} < 1$ then there must be graphs without either of them. Fixing n and estimating k, one deduces that there must be graphs of size n, which do not have a clique or an independent set of size $O(\log n)$.

“People tried over many decades to construct graphs that don’t have small cliques or independent sets and it was pretty pathetic what they could do. The best that were known from combinatorics [6] are of the order $k = O(\exp(\sqrt{\log n}))$. Then came a series of works from computer science that used the idea of randomness extraction. This technology gets you down to clique sizes of $O(\log n^{\log\log n})$ or maybe a little better [2,3], close to the optimal bound.”

A key notion in the study of randomness extraction is weak random sources [4]. These are random variables of binary strings with a fixed length n such that each individual string has a probability of at most 2^{-s}. The number $n-s$ is called the (min-)entropy of the source, if $n-s=0$ the random source is uniformly distributed and often called unbiased or pure.

“The study of randomness is something in which I invested a lot of my energy. It is a fascinating topic and has many sides and facets. One type of question is: how do you utilize a weak source of randomness? How do you utilize something that doesn’t look independent or unbiased, maybe stock market fluctuations or the output of some physical process like the weather. You expect that on the one hand it looks somewhat unpredictable — otherwise you could always buy and sell the right stocks — on the other hand it’s not like a completely random source.

Suppose I give you just a distribution that has some entropy in it, e.g. maybe every other bit is fixed by an adversary or I give you a distribution on n bits and $\sqrt{n}$ bits of entropy in it somewhere but you know nothing more. Can you use it in a probabilistic algorithm? It turns out that a theory developed over 25 years, the theory of randomness extraction or randomness purification, tells you how to do it, so the answer is yes. You can always purify weak random sources and use them in probabilistic algorithms, at least when you have some entropy. This is really quite surprising but we understand it very well today.”


A Ramsey graph with 17 nodes that contains no cliques or independent sets of size 4

Pure randomness is hard to come by, more often computers are restricted to use variables that look random but in reality are deterministic deep down. Surprisingly, pseudo-number generators based on computational complexity theory can be used to remove randomness from efficient probabilistic algorithms.

“Pseudo-randomness is another theory that I invested a lot of time in. This phenomenon requires a computationally hard function. Take for instance systems of quadratic equations. Nobody has any idea how to solve these efficiently. Even if you work over a finite field, we don’t know any algorithm better than exponential time. If you believe that any such natural hard problem requires at least exponential time, then you can eliminate randomness from any efficient probabilistic algorithm. That is another major understanding: that randomness is actually not as powerful as we think it is. Look at all the probabilistic algorithms that people use and that don’t seem to have any deterministic counterparts. The truth is they do and this is pretty remarkable.

This realization happened rather quickly. As a postdoc I wrote a paper with Miklós Ajtai at IBM where we did these kinds of things for a very limited class of algorithms that run very fast in parallel [1]. We already realized that any hard function for this class can serve as a basis for creating pseudo-random generators. This was also known in the cryptographic world, usually under stronger assumptions. Anyway, it was in the air that if you have a hard function you can generate pseudo-randomness from scratch. Then Noam Nisan, a graduate student at Berkeley, came to visit me for a semester at the Hebrew university and together we made this fundamental connection work for any efficient algorithm [8].

From the beginning, it was obvious to everybody that randomness is powerful, but signs that sometimes you can eliminate them began to crop up. We had all these examples and within a week or a couple of days we realized that it is very general and you can always do it. There were various parameters to make the result much cleaner and stronger. In a sense even though I was not consciously thinking it should be true, we were just pursuing the power of the paradigm that hardness can be converted into randomness.”

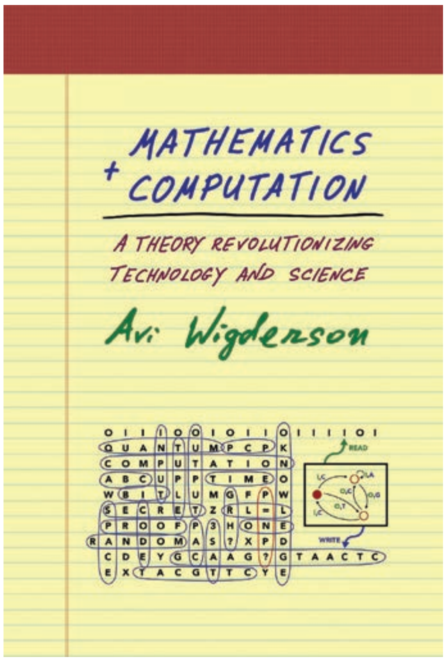
The computational lens

As illustrated by the examples above, looking at mathematics from a computational perspective can be a very fruitful endeavor and it inspired Wigderson throughout his whole career.

“I think that over forty years my understanding has evolved and improved a lot. Thirty years ago there were some glimpses, not just by me but by the whole community. We didn’t understand the full power of this computational lens. Twenty years ago it was much clearer. The algorithmic viewpoint kept improving and I collected many more examples to demonstrate.”

In 2019 Wigderson published a book [9] that offered a bird’s eye view of computational complexity theory, including all its connections and interactions with mathematics, the natural and social sciences, technology, and philosophy.

“The origin of the book is actually a funny story. I was hoping to write a popular book for the general public. I tried for several years and I didn’t get too far. I really had problems with finding the right level of presentation. I didn’t realize it was so difficult. I like giving popular lectures and I never had a problem with that, but a book



was a different challenge. So then Edna, my wife, suggested that maybe I should write a book I can finish.

I realized that I could finish a book that would be a little more technical but still accessible to the undergrads. I could write something high level, not a book with the proofs but a book about the ideas: the motivations for different parts of computational complexity, the main results and their overall consequences rather than the details.

It would not be a real textbook but it would explain many parts of computational complexity, how they relate to each other and how they connect to math, physics, biology and economics. I am very happy with the way it turned out.”

We wholeheartedly agree and recommend it to all our readers, especially those who prefer math books of a more generalist nature, that do not shy away from the bigger picture.

References

1 M. Ajtai and A. Wigderson, Deterministic simulation of probabilistic constant-depth circuits, *Proceedings of the 26th FOCS*, October 1985, pp. 11–19.

2 B. Barak, A. Rao, R. Shaltiel and A. Wigderson, 2-source dispersers for $n^{o(1)}$ entropy, and Ramsey graphs beating the Frankl–Wilson construction, *Annals of Mathematics* 176(3) (2012), 1483–1543.

3 E. Chattopadhyay and D. Zuckerman, Explicit two-source extractors and resilient functions, in *Proceedings of the 48th annual ACM symposium on Theory of Computing*, ACM, 2016, pp. 670–683.

4 A. Cohen and A. Wigderson, Dispersers, deterministic amplification, and weak random sources, *30th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1989.

5 P. Erdős, Some remarks on the theory of graphs, *Bulletin of the American Mathematical Society* 53(4) (1947), 292–294.

6 P. Frankl and R. M. Wilson, Intersection theorems with geometric consequences, *Combinatorica* 1(4) (1981), 357–368.

7 S. Goldwasser, S. Micali and C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM Journal on Computing* 18(1) (1989), 186–208.

8 N. Nisan and A. Wigderson, Hardness vs. randomness, *Journal of Computer Systems and Sciences* 49(2), (1994), 149–167.

9 A. Wigderson, *Mathematics and Computation*, Princeton University Press, 2019.