Ron Fagin Speaks Out on His Trajectory as a Database Theoretician

Marianne Winslett and Vanessa Braganholo

Ron Fagin
http://researcher.ibm.com/person/us-fagin

Welcome ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are in Snowbird, Utah, USA, site of the 2014 SIGMOD and PODS conference. I have here with me Ron Fagin, who has spent many years as a researcher at IBM. He is an IBM Fellow. He is a Fellow of ACM, IEEE, and the American Association for the Advancement of Science. He was elected to the National Academy of Engineering and the American Academy of Arts and Sciences. He has won the IEEE McDowell Award (the highest award of the IEEE Computer Society), the IEEE Technical Achievement Award, and the SIGMOD Edgar F. Codd Innovations Award, and he has won a bunch of Best Paper and Test-of-Time Awards. He was named Docteur Honoris Causa by the University of Paris. Most recently, he won the Gödel Prize in 2014. Ron's Ph.D. is in mathematics, from Berkeley.

So, Ron, welcome!

Thank you, Marianne.

Tell me about the work that you received the Gödel Prize for.

Well, actually it was a bit of a long story. It arose when Laura Haas knocked on my door one day and said, "Okay Mr. Database Theoretician, we've got a problem." So I said, "Laura, what's the problem?" and she said, "Well we have this middleware database system called Garlic, and it is on top of pure database systems like DB2, and it's also on top of QBIC." QBIC ("Query by Image Content") is a system where you can query by image content: you search for objects based on color, shape, or texture. She said, "The trouble is that there are mixed data types. The answer to a query in a normal database system is a set (or a bag), and the answer to these multimedia queries is a sorted list." She said, "So what do we do? How do we combine the results together?"

I thought about it and came up with a solution involving fuzzy logic (where a proposition can be not just true or false, but somewhere in between). I was very excited. I went to see Laura and said, "Laura, I got you an answer. Use fuzzy logic." She said, "That's good, Ron. But we don't have time to look at every single item in the database and assign some kind of fuzzy score to it. We need to get our answers fast. I need an efficient algorithm." So I said, "Okay, fine."

I went back to my office and a day or two later I came back and said, "Laura, good news, I got a square root of n algorithm for you (where n is the number of objects in the database)." She said, "Great! Square root of n beats linear, but you know what Ron, we database people are spoiled. We are used to log n algorithms like in B-trees." I'll never forget what she said to me next. She said, "Ron, be smarter. Go back to your office and get me a log n algorithm."

So I went back to my office and came back a day or so later and said, "Laura, I can prove square root of n is the best you can do. It's a matching upper and lower bound. " She said, "Fine; we'll take it." And it was implemented in Garlic. Then a few years later (and here's where the Gödel Prize winning work came in), I was doing some work with Moni Naor and Amnon Lotem, and we miraculously came up with a new algorithm called the Threshold Algorithm, which beat Fagin's Algorithm ("Fagin's Algorithm" is the name of the algorithm that I originally gave to Laura -- she named it that, and it appeared in papers that way). The Threshold Algorithm is optimal but in a stronger sense than Fagin's Algorithm. Fagin's Algorithm is optimal in a certain worst-case sense, which is the usual standard for optimality of an algorithm. The Threshold Algorithm is optimal not just in the worst case, or in the average case, but in every case! We called this property "instance optimality". Thus, the adversary can design his own database and his own algorithm fine-tuned to that database, and our algorithm can perform just as well on the adversary's database as the adversary's algorithm performs on the adversary's database. Even though the algorithm is only about ten lines long this paper won the Gödel Prize, which is the highest award for a paper in Theoretical Computer Science! It was hard to find that algorithm, but once you have it, it is easy to verify. Our paper is the only database paper ever to win the Gödel Prize. Our definition of instance optimality is a strong notion -- it was an exciting notion to the people in the Computer Science Community.

My goal is to convince theoreticians that they will prove better theorems and they'll do more interesting work if they just talk to practitioners.

Okay, you have won two Test-of-Time Awards from PODS and one from ICDT. What were those pieces of work about?

Well, the first Test-of-Time Award from PODS[1] was for the work that eventually won the Gödel Prize. It also won the Best Paper Award for that conference. The other two Test-of-Time Awards, the one from ICDT, which we got last year[2], and the one from PODS that we are getting this year[3], both had to do with data exchange. Data exchange deals with converting data from one format (the source) to another (the target). In data exchange, there are certain first-order logic formulas called "tuple-generating dependencies" (or TGDs) that specify a relationship between the source and the target, but do not

[1] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal Aggregation Algorithms for Middleware. In: PODS, 2001.

[2] Ronald Fagin, Phokion Kolaitis, Renee Miller, and Lucian Popa. Data Exchange: Semantics and Query Answering. In: ICDT, 2003.

[3] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. In: PODS, 2004.

completely specify the target. In the first paper, which won the Test-of-Time Award last year, we described a particular family of choices for the target (which we called "universal solutions") with a number of desirable properties. The concept of universal solutions became widely accepted.

This year's Test-of-Time Award for PODS was for composition. There, the question is "If you convert data from format A to format B and then convert from format B to format C, how do you convert directly from format A to format C?" To our amazement, it turned out that even if the methods that go from A to B and from B to C are both specified by these simple TGDs, going from A directly to C not only could take you away from TGDs but even out of first-order logic! We had to go to second-order logic. We invented something called second-order TGDs, and we proved that those were exactly the right ones for the task. Specifically, every composition where each component is specified by first-order TGDs can be specified by a second-order TGD, and for every second-order TGD, there is some sequence of compositions where each component is specified by a first-order TGD that gives that second-order TGD.

You just got named to the American Academy of Arts and Sciences too!

True! I'm really very proud of that because, for example, Wikipedia calls it one of the nation's highest honors. Something really cool about it is that it was founded during the American Revolution, and the first class included George Washington and Benjamin Franklin. Each year the Academy selects something like 7 or 8 people from each discipline. Like 7 or 8 computer scientists, 7 or 8 mathematicians, 7 or 8 physicists, and (since it's arts and sciences) they even pick people from movies and TV. So it's exciting to be invited to become a member of that select group.

What will you guys do?

Well, I think one of the main purposes of all these National Academies is to figure out whom we elect for the next year (laughs). It seems to be! (More laughs).

I know from the National Academy of Engineering that this selection process seems to be the immediate task. In my short time of my being a member, we spent a lot of time figuring out who is going to be next year's candidates. I know that an important role of the National Academies is to conduct policy studies. I have not been involved in that, and I'm not sure how much I could contribute to policy. I'm currently excited that the National Academy of Engineering is going to have a black tie inauguration. I have to actually go in a tuxedo, since they take this very seriously.

It's like the prom all over again!

Exactly like the prom!

Well, congratulations on that too. Now it must be pretty cool to have a theorem named after you. What is Fagin's Theorem?

What it does is to tie together complexity theory on the one hand with logic on the other hand. There's the important complexity class NP, and there's also something called existential second-order logic. What Fagin's Theorem says is that they are really equivalent (for example, the class of 3-colorable graphs is both in NP and expressible in existential second-order logic). On the face of it, NP and existential second-order logic look very different; they're from different disciplines (one from complexity theory and one from mathematical logic). Because of this equivalence, you can use tools from one area to help you in the other area. It's always cool when you get a connection between two very different fields, and that's what Fagin's Theorem does. There has been much follow-up work.

And that was done back in your dissertation.

Correct. Incidentally, my Ph.D. thesis seemed to be completely unnoticed for a number of years. There is probably a moral in there about keeping the faith.

I feel like much of your career was the Golden Age of Logic for Computer Science? It seems like things are switching over to the Golden Age of Statistics in Computer Science. Have you seen that too?

Yeah, I do see some of the statistics, but logic is still going strong. In fact, Jeff Ullman and I were two of the founding fathers of relational database theory, where Jeff focused on the algorithms, and I focused on the logic. Both tracks are still very active. But you're right Marianne, there are other things that are entering in the picture, but logic is an important track and is still very much there.

Okay, you've already mentioned Fagin's Algorithm. We've got Fagin's Theorem, Fagin's Algorithm, Fagin's 0-1 Law, Fagin games, Ajtai-Fagin Games, and the Fagin-inverse. It's good that you don't have a really long name I guess. So what's Fagin's 0-1 Law?

It's one of my favorite theorems ever. It says the following: take any sentence in first-order logic involving only relational symbols, and it's either going to be almost always true or almost always false in the asymptotic sense. That is, as the size of the finite structures get larger and larger, the fraction of structures that obey the sentence will converge, and it will converge to either 0 or 1. So first-order sentences are either almost always true or almost always false. That's what the 0-1 Law says.

IBM has been around for more than 100 years now, and you're not around for more than 100 years by just continuing to do what you have always done, because what you do eventually becomes obsolete and then you have to move on.

Is there an impact on the practical side of Computer Science from that?

Well, some people say they use the 0-1 Law to help them understand the average case behavior of things. I'm not quite sure if I buy into that. I mean, it's a motivation I've heard people give because there's been a lot of work on the 0-1 Law, extending it to other logics and so on. People argue that it has this practical impact. To me, it's this beauty. To me, it's just very neat that things perform in such a very simple, natural way. You'd think that either the probabilities might not converge, or they might converge to a half or two-thirds or something, but no, they always converge, and always to 0 or 1. To me, it's just mathematically beautiful. That's why I love it so much. I have to say that I love my proof too. In fact, of all of my results over the years, my proof of the 0-1 law is probably my favorite.

What about Fagin games and Ajtai-Fagin Games?

So-called "Ehrenfaucht-Fraisse games" are used to prove inexpressibility results in logic. In an Ehrenfaucht-Fraisse game, there are two players, called the Spoiler and the Duplicator, and they take turns picking points. There are two structures, and the Spoiler picks point 1 in one structure (either the first structure or the second structure), and then the Duplicator picks point 1 in the other structure. Then the Spoiler picks point 2 in one structure (again, either the first structure or the second structure), and then the Duplicator picks point 2 in the other structure. This continues for a fixed number of rounds. Consider the mapping between the two structures, where for each k, the kth point selected in the first structure maps to the kth point selected in the second structure. The Duplicator wins if this mapping is an isomorphism, and otherwise the Spoiler wins. Proving that the Duplicator has a winning strategy gives an inexpressibility result for first-order logic.

There are a lot of variations to that game. Fagin games arise when you try to prove inexpressibility results in a logic called existential monadic second-order logic. Then the rules get a little more complicated. In Fagin games, you again have the Spoiler and the Duplicator, and they first color the points. Thus, the Spoiler colors the points in the first structure and then the Duplicator colors the points in the second structure. Then they play the Ehrenfaucht-Fraisse game I described earlier, but now for the Duplicator to win, the isomorphism must respect colors. Miki Ajtai and I came up with new games (now called "Ajtai-Fagin games") in which we changed the rules of the Fagin game in an interesting way to make it much easier for the Duplicator to win, which gives a much easier proof of inexpressibility results.

Okay, and that leaves the Fagin-inverse.

That's something from data exchange. I talked earlier about converting from format A to format B, but what if you say, "I want to go back from B to A. How do I do that?" The Fagin-inverse is all about going backwards. There are a lot of very subtle issues that arise: the inverse may not exist, and even if it exists it may not be unique. So I defined this thing that is now called the Fagin-inverse and described how you take a mapping that goes from A to B and when and how you can invert it, using the Fagin-inverse, to go from B to A. It's not obvious. It's not obtained by simply reversing the arrows. Since then, there have been a number of other flavors of inverses that have been studied.

Okay, being an IBM Fellow gives you a bit of hope for evangelizing for your favorite technical causes inside IBM. How have you used that?

I'm glad you asked. My mission as an IBM Fellow has been to convince theoreticians and practitioners to work together. My goal is to convince theoreticians that they will prove better theorems and they'll do more interesting work if they just talk to practitioners.

By talking to practitioners, they will discover new exciting problems that no one else has considered before, and then other people will jump on the bandwagon. You will be creating a new field, and you'll have a real impact. The best example I can give of this is that resolving the very practical problem that Laura Haas posed to me led to the Gödel Prize.

I also have to convince practitioners they should work with theoreticians to make their products better: they'll get new algorithms, they'll get performance guarantees, and they'll have a much more solid system with features that other systems don't have. So as an IBM Fellow, my mission has been going around to IBM's worldwide research labs, giving lectures on this, talking to the young people to mentor them and to spread my gospel on applying theory to practice. I have recently expanded my mission by giving my speech on applying theory to practice at a number of major universities. My goal is to get theoreticians and practitioners to interact more with each other.

Do you think they believe you?

Well, they seem to. It is much easier for theoreticians to work only with other theoretician, to just talk to people who speak their language. It's a real effort to speak to someone outside your field. There are two ways I tell people it can happen. One way is like the way I told you with the story of Laura knocking on my door and saying, "I've got a problem." But there's another way it can happen, and this is what happened in the work on data exchange that we won the two Test-of-Time awards for. The data exchange project called Clio, also led by Laura Haas, had been going on for over a year, and because of how well things went with Laura earlier on the Garlic project, I had been regularly attending Clio meetings. Then Phokion Kolaitis, Lucian Popa, Renee Miller and I (later joined by Wang-Chiew Tan) said, "This data exchange work at IBM has been going on for a long time. But let's see how we would do data exchange if we did it from scratch. Let's see what the right way to do it is. Let's have no preconceived ideas, and just say: if we were doing data exchange and no one told us anything about it, how would we do it, using principles from database theory?" That's what we did with data exchange, and it led to a very successful body of work. In fact, our ideas were implemented in Clio. This included the use of second-order TGDs as the internal mapping language of Clio. (As I mentioned earlier, second-order TGDs arose as the result of our theory work on composition of TGDs that received the PODS Test-of-Time Award in 2014.) And because of our work, every major database conference started having special sections on data exchange. We felt good that we brought data exchange out as a discipline with interesting technical results. There has been a lot of work done on data exchange ever since.

That list of people that you gave… Most of them I'd say are more from the theory side.

True. Lucian, however, played a very key role. Lucian lives on both sides of the aisle. Lucian was heavily involved in the actual implementation side of Clio, and he also does theory. One of the great things about working with Lucian was that he was our bridge to the other world. He understood what the issues were and he would keep us honest. For example, when we discussed technical issues, he might say "Okay, guys, now we're going off into never-never land. No practitioner cares about the issue we are now discussing, so let's consider this other direction instead."

What about the finite model theory?

Finite model theory is the topic of my Ph.D. thesis, and so is really near and dear to my heart. My thesis is where Fagin's Theorem, the 0-1 Law, and Fagin games appeared. I'm happy that people consider me the founder of finite model theory. Now, lots of work is being done in the area, and finite model theory has been applied in a number of different ways. That's something I'm proud of.

That idea, did that come from talking to practitioners?

No. I was at Berkeley writing my Ph.D. thesis, and the ideas all arose in different ways. For example, my 0-1 Law arose from a huge question in finite model theory, which is closure under complement of various classes. For example, if a property can be expressed in existential second-order logic, which I showed is the class NP, is the complement also expressible in this logic? This is really close to the P vs. NP problem: it's the NP vs. Co-NP problem. While playing with the notion of closure under complement, I realized to my surprise that in ordinary first-order logic, if a property is interesting, then its complement seemed to be very uninteresting. For example, consider the conjunction of the field axioms. That is an interesting first-order sentence. But its negation (which defines the complement) is very uninteresting, since there are many ways to fail to be a field. I wondered how I might prove some theorem that says that in first-order logic, if a property is interesting, then its complement is very uninteresting. I concluded, "I can use asymptotic probabilities." Specifically, I decided to interpret "very uninteresting" to mean "almost always true". This would imply that either a first-order

sentence or its negation is almost always true. And that's what I proved, via the 0-1 Law. As for your question, I got to this without talking to any practitioners.

Okay. You've been at IBM for over 40 years. The IT companies that were big back in the mid-70s are all dead now, except for IBM. Why did IBM survive when so many others did not?

I think its adaptability. IBM has been around for more than 100 years now, and you're not around for more than 100 years by just continuing to do what you have always done, because what you do eventually becomes obsolete and then you have to move on. There are Harvard Business School studies about this issue. If your company is extremely successful at something, and you see something new coming up that is going to replace your very profitable line of business, it's hard to switch to it, because, in the short term you're going to lose a lot of money since you're suddenly pushing customers from your expensive solution that was your bread and butter to something else. But if you don't do it, someone else will, so you better do it. IBM has learned that lesson, and IBM has adapted a number of times. IBM is doing that right now, by the way.

What have they been giving up right now?

The issue isn't a matter of IBM giving up on things, but rather a matter of IBM devoting more and more of its resources to areas that are crucial for the future.

What are the new big things at IBM?

The big new things are "CAMSS": cloud, analytics, mobile, social, and security – and, of course, artificial intelligence. So IBM is moving heavily into all these areas.

You knew Ted Codd, didn't you? Tell me a story from the early days.

Oh, so let me tell you how I got involved with Ted Codd. I transferred from IBM Watson to IBM San Jose, and when I transferred, I looked around and said, "Okay, who's interesting here to work with?" There were a number of interesting people, but the guy who I thought was most interesting was Ted Codd, and I went to him and said, "I'd like to work with you." I was thrilled that he said yes. So Ted was my mentor, and he was my hero. He really helped my career.

One thing I remember (even though it's not a big deal, but to me it was huge at the time) is that he took me to a SIGMOD conference after I'd done some work with him and he put his arm around me either literally or figuratively (I'm not sure which) and he introduced everyone to me saying "This is Ron Fagin, he's a new employee at IBM and he's doing great work on relational databases." I just glowed, and I thought, "Wow the great Ted Codd, who is already the icon, is saying these nice things about me." That was even before Ted was an IBM Fellow, and before he won the Turing Award. I got into databases because of Ted Codd. He was my mentor, and he was doing relational databases, so by golly, I did relational databases.

[…] what's important for me, Marianne, is completely understanding something. Putting my arm around it, totally, deeply, completely understanding it.

So besides positive feedback, you said he had a big influence on your career.

Just talking to him – I would talk about relational databases, and he would understand it, of course, totally, deeply and that would help me understand it better. This was why I got into relational database theory. From talking with Ted, I had a good feeling about what databases were all about, what they could do, and why relational databases were different from previous ways of doing databases. I then began to understand what he was doing and how important it was, and I wanted to get involved, and I did.

Do you have any words of advice for fledgling or mid-career database researchers?

My advice I give all young people is to go to lots of talks, interact with lots of people, do different things, and open your mind. You'll never know when you will find something cool and exciting. And then follow your heart and work on what seems most important to you.

If you magically had enough extra time to do one additional thing at work that you are not doing now, what would it be?

This is kind of off the wall but believe it or not, cosmology. I am fascinated by the notion of multiple universes, and in fact I'm almost obsessed by it. Actually, I even wrote an unpublished paper about how

to calculate the probability of our own universe colliding with another universe. It sounds weird, but I based it on what I called the probability of a big bang per cubic meter per second.

How high is that probability?

It's pretty low, but I wrote this little paper on it. Then I put "colliding universes" into Google, and to my delight, it turned out that the world expert on colliding universes was a UC Santa Cruz professor whom I had met at a party! So I thought, "Okay, I'll send him my paper and see what he says." I thought he would just ignore it, but he was kind and said, "You have some nice new ideas. However, your paper violates both relativity and quantum mechanics." Oops. I'm not a physicist, so I wrote my paper from a Newtonian point of view. But I'm still fascinated by the notion of multiple universes, even though I'm a bit discouraged about my prospects for winning the Nobel Prize in physics through my cosmology work, given that it violates fundamental laws of physics.

By the way, I want to say something about laws of physics. I don't understand quantum mechanics. I admit it freely. I didn't go into physics but went into mathematics and later computer science because I just don't understand quantum mechanics. It isn't at all intuitive to me. And I felt much better, years later, when I found out that Richard Feynman said, "If you think you understand quantum mechanics, then you don't understand quantum mechanics." I thought, "Yes, it's not just me, it's everybody! If Richard Feynman, a Nobel Laureate in physics, says that, then none of us understand quantum mechanics, so it's okay that I don't understand it at all."

But does that mean that you should have gone into physics afterward?

No, because what's important for me, Marianne, is completely understanding something. Putting my arm around it, totally, deeply, completely understanding it. I feel like the reason I'm in mathematics is because I felt like I could do that. I felt like I could take that area and study it and think about it and read about it. It would be mine. I would own it. I would totally completely understand it in every way. In physics, I realized, I could never do that, because no one can. In some ways, physicists blindly follow some mysterious formalism that they don't completely understand. They may not view it that way, but I can't work like that. So I'm really glad I didn't go into physics, I wouldn't be happy just pushing equations around. I have to totally, completely understand things, and deep in my soul, I don't understand physics.

Well, it's good for computer science, I guess, that it turned out that way, but if you could change one thing about yourself as a computer science researcher, what would it be?

Actually, you know what? I'm not sure if I would change anything. I feel like I've been very lucky. Things have fallen my way, and I feel like I've made some good choices. Working with Ted Codd and getting into databases is an example. And it's gone so well, I don't think I'd change a thing. I never dreamed, by the way, of ever becoming an IBM Fellow, because the typical IBM Fellow brings like a billion dollars to IBM and I thought there's no chance that IBM would take a theoretician like me and make him an IBM Fellow. But, miraculously, they did. So things fell my way, and I'm delighted with how things have turned out. I couldn't ask for it to go any better so I wouldn't change a thing.

Okay, well thank you very much for talking with me today.

Thank you, Marianne. It was fun.