Moshe Vardi Speaks Out
on the Proof, the Whole Proof, and Nothing But the Proof

by Marianne Winslett

Moshe Vardi
http://www.cs.rice.edu/~vardi/

Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and I have here with me Moshe Vardi, who holds an endowed professorship at Rice University and is a former chair of their Computer Science Department. Before joining Rice, Moshe was a manager at IBM Almaden Research Center. Moshe is an ACM Fellow, a AAAI Fellow, a co-winner of the Goedel Prize, and a member of the U.S. National Academy of Engineering and the European Academy of Sciences. His research interests include databases, verification, complexity theory, and multi-agent systems, and his PhD is from the Hebrew University of Jerusalem. So Moshe, welcome!

Thank you very much. It is a pleasure to be here.

Moshe, after you finished your PhD, you worked on Datalog for a number of years---as did many, if not all, database theoreticians. At the time, there was vocal opposition to some aspects of this activity from some of the more practically oriented members of the database research community. In hindsight, do you think that the criticism was justified?

It surprised me that people get so emotional about certain research areas. The work on integrity constraints in the late 70s and early 80s also received scathing criticism as not being at all relevant to the practice of database systems, only to reemerge later as being of central importance. I heard recently a question that someone asked Stephen Hawking: what is the best idea in physics this year? And Hawking said that we won't know for many years. When you do an exciting piece of research, it is very hard to know whether it will be relevant to the field in the long term. This is true both for theory and for experimental work. The vast majority of theory research results will be forgotten, as will be the vast majority of experimental work. The fact that something is done experimentally does not guarantee any lasting impact.

We can look back now at Datalog, and ask what its impact was. I don't know of any database that implements Datalog per se, but SQL3 does include recursion. A case can be made that the Datalog work did contribute to realizing the importance of recursion. Georg Gottlob has a company that does Web integration using Datalog as its internal language. The Datalog concept of rules has had an impact on active databases. So yes, Datalog did have an impact. That doesn't mean that every single technical result, every PSPACE-completeness result, had much impact; but the same is true for most scientific work. Most scientific work will be forgotten.

Datalog has been very influential in the area I work in, in security.

Languages for security policies?

A very good language for security policies. Although it is not necessarily the direction that the industry is going in, perhaps they should *be going in that direction. Certainly in security research, you want to be able to reason about the correctness of what you are doing.*

What do you view as the most important open problems in database theory today?

Have you tried to use databases? They are darned hard to use. We have built a marvelous piece of middleware that is incredibly powerful and incredibly difficult to use. The challenge for the database field is how to have more usable database systems.

I use databases all the time. I use them every time I go to an ATM machine, and it is very easy. So why are you complaining that they are hard to use?

For you it is easy, but it is very expensive to make them easy for the consumer to use.

So you are not worried about ease of use, you are worried about the expense of making them easy to use.

Someone actually has to write the application. At an ATM, you don't deal directly with the database, you are interacting with an application. It is very expensive to write an application and maintain the database---I'm not going to be the first person or the last person who will say that. What we lack (and this is true both on the implementation side and the theoretical side) is a good set of abstractions that will make life easier for people who build databases and people who build and maintain applications.

The relational model overall was a wonderful abstraction, a beautiful abstract model. It cleared away a lot of bushes from the pathway, and life became easy for a while. Now the bushes are growing across the pathway again. If you talk to people in forestry, they will tell you that fires are good because they clear the brush from the forest. The relational model was a fire that cleared a lot of brush away, but we haven't had a good fire for a while.

Is there a role for the theory community there? Basically, you are talking about a revolution in the API?

Well, the API is one way to look at it, but the relational model first and foremost was a theoretical contribution. The experimental work came later to show that this theoretical contribution is realizable and can result in practically and pragmatically usable databases. In our time, we have tried another abstraction. Everybody claimed that object-oriented databases were the way things would go. But it didn't work out that way, so we can ask, why not?

There is no doubt that we are looking for new abstractions, and in that task theory plays a critical role. It is not enough to have new theory, it must also be paired with implementations and experimental work. In the database research community culture, I don't think we have a very healthy ecosystem where there is a constant interplay between theory and practice. Instead, for example, the theoreticians come and say that integrity constraints are important, and the more experimental people say that no, they are not important. After a while the theory people stop working on integrity constraints. Then the experimental people say, “No, no, we need integrity constraints!” And the theory people say, “Oh, we are not interested now, we looked at them ten years ago and now we’re not interested.” In other areas of my research, verification for example, there is a much more healthy interchange between more theoretical research and more applied research.

I think I misunderstood you earlier. You don’t want a better API, you want a new model*?*

It is not necessarily just the model; there is a whole set of abstractions that goes into building a system. The relational model was one such abstraction. Another abstraction was logic and relational algebra as a way to query. Another abstraction was the concept of keys, as a way to describe integrity constraints, and normalization. Transaction management was another kind of abstraction. So we are not tied to one single abstraction; we need a mix of abstractions. We can’t just replace tables by trees and be done. It’s more than that. But the complexity of today’s systems suggests that there is a prime opportunity for the theory community to develop and reason about new abstractions.

Do you have any proposed directions to find these new abstractions?

I don’t. If I had, maybe I wouldn’t be here! There is some work happening in the XML world, where people are trying to figure out a more abstract way of looking at XML. Maybe that will have lasting impact. Interestingly, we see theory playing a bigger role in the XML world than it did in the relational world. If XML becomes the primary way of storing data, then perhaps we will see some impact there.

In that earlier fiasco with dependency theory, I think that the integrity constraints that the theory people were looking at were not the kinds of integrity constraints that come up in practice, so that is where the mismatch happened.

I think that was part of it. At the time we theoreticians didn’t get feedback saying, “This is very nice, but here is the constraint that we really care about, and please work on it instead.” Instead, what we got was a scathing response that nobody needs anything more than keys. And then later, “Well, maybe we’ll eventually need referential integrity, but nothing else.” Today if you look at SQL, you find that there are keys, referential integrity, and assertions. Assertions are incredibly powerful and we never did develop a good theory of how to handle them well. So we don’t have a healthy interchange between theory and experimentalists. I’m perfectly okay if somebody says, “Very nice work, but if you can change it a little bit, maybe it will better fit my needs,” rather than “You’re wasting our time, you’re wasting your time, stop bothering us,” which is, I have to say, very often what we were hearing.

There is no Nobel Prize in computer science, but there is a Nobel Prize in economics. I have heard that database theoreticians have switched to working on game theory for this reason. Is that rumor true?

People work on game theory because games are a powerful paradigm. For example, I use games in my verification work. I don’t think game theory work will get a Nobel Prize. I think one should not structure one’s work for Nobel Prizes. Too few people get them; it is not a good career driver. One should strive to make impact, to have success, for other considerations. I know people who work on bioinformatics because they think they might get the Nobel Prize in medicine. I think it is a bad strategy for designing one's career.

A Nobel Prize is *a bit of a long shot, but maybe one can pick up some other prizes along the way.*

Game theory already received the Nobel Prize twice in economics, which surprised people. Some people thought after the first prize that game theory had had its time, and then there was a second prize for game theory. So I wouldn't bet on a third one.

How about bioinformatics?

Bioinformatics is a less of a long shot in some sense. Again, one would have to do something fairly dramatic to win a Nobel in that area.

You could argue whether the contribution by some people to the genome project was worthy of the Nobel Prize in medicine. I suspected that that project was so controversial that nobody from that project would get the Nobel Prize, because the Nobel Prize committee would not want to sort out the intense rivalries there. If the medicinal contributions of the genome project would materialize, which so far has not happened, then the genome project would be one of the most successful contributions of computer science to medicine.

Moshe, you were one of the main players in the area of finite model theory. Many of our readers will not be familiar with this fascinating subject. Can you tell us a little bit about it? Perhaps some of your favorite results in the field, and how they may relate to the database world?

Finite model theory emerged at the confluence of databases, complexity theory, and logic. The realization that several people had independently was that classical mathematical logic, and model theory in particular, deal with infinite structures. They deal with infinite structures because arithmetic is infinite and real numbers are infinite; logicians almost looked down on finite structures. *Finite* was almost boring; it was too trivial to study. We realized that there is motivation coming from computer science to study finite structures, and a beautiful theory emerged. Questions and issues come up that do not arise at all when you look at infinite structures.

You can ask what the implications of finite model theory are for databases and complexity theory and perhaps for other areas. The answer is that we don't know yet. So far, in my opinion, the implications of finite model theory for any other area are somewhat modest. We have a beautiful mathematical theory and there are some textbooks. We hope that the beauty can eventually lead to applications, just like in any other area of mathematics, but that can take a while.

From the finite model theory work, we learned about the limits of expressivity of first-order logic. When SQL was developed, Ted Codd and other people thought that first-order logic was as expressive as one would ever need. The work on finite model theory led us to understand the limits on expressivity of first-order logic and therefore of SQL. This contributed to the eventual inclusion of recursion in SQL, because we now had formal proof that recursion does add expressive power. This is probably the most measurable impact of finite model theory on databases.

There was some hope that finite model theory work on random models might tell us something about average query processing behavior. I don't think that has been borne out. People were hoping that finite model theory would lead to significant progress in complexity theory; that so far has not been borne out. There is some recent work that uses finite model theory to give us new techniques in query optimization, and that might become significant.

What are the hot topics in computational logic today?

I think that one of the major successes in computational logic has been the applications of logic to theoretical formal methods. (I just discovered that [the Department of Computer Science here at UIUC] has a very large group in formal methods.) The issue of how to design better systems has been one of the challenges of computer science for the last 50 years. Just in the last decade, formal methods have been coming to the fore as a major body of techniques to do better systems design, programming, debugging, and other tasks related to creating more reliable computer systems. Formal methods have had major success in industry.

I would love to see more of my database work having impact on database practice, but most of my work has been theoretical, without technology transfer. I think that some of my theoretical verification work will have an impact on industrial practice. That is where I spend most of my time: not in database research, but in trying to transfer ideas from computational logic to industrial practice. I think it is a very exciting area nowadays.

You served as department chair at Rice for many years.

Too many!

Nowadays, many departments are working hard to move up in the US News & World Report *rankings---so much so that a department has to continually improve itself just to maintain its current position. Finite model theory tells us that this cannot go on forever. So where will it all end?*

The goal of improving a department's ranking is not a realistic goal. You can say that you want to have more graduate students; that is something you can measure: how many graduate students do you admit per year? You can look at the average GRE score of the students, and say that you would like to have better graduate students. You can say that you would like to see your department getting more funding. There are all kinds of things you can measure and you can control, but you cannot control the ranking that *US News* will generate with their complex formulas. Since everybody is trying to improve, it would be nice if you could squeeze more departments into the top ten. But since only ten departments can be in the top ten, I think it is not a useful goal for departments to have. I would advise departments to focus on measurable goals and attainable goals. I never felt that the goal of ranking improvement was useful or attainable.

You can only improve your graduate student population so far, and you can only increase your funding so much. If everyone else is also trying to compete for the same pool of excellent graduate students and to compete for the same pool of research dollars, how is it all going to end?

I will tell you where it is going. Everything is going down, unfortunately. There is a biblical story that says that there were seven fat cows, and there were seven lean cows. This meant that there were seven good years, and they were followed by seven bad years. Right now, I look at the global situation and see that we are in for a tough period. Now our job is to survive in this tough period and do the best we can.

Well, what the biblical story tells us is that during the fat years, plan well for the lean years. That was the wisdom of Joseph. I don't think we did it. We enjoyed the good years and we thought they would last forever. And they didn't last forever.

What should we have done that we didn't do? Go to Congress and talk up the field?

We have not communicated effectively the contributions of our field. The public as a whole knows very little about it. When you open the newspaper, you find stories almost once a month about black holes. Somehow, the stories excite people about black holes. When you think about it, it is utterly bizarre; why would be people be so interested in black holes in the center of the galaxy? But people *are* interested in black holes. The authors made the topic interesting. We computer scientists have not told our stories well to make them interesting to the public. Maybe our topics are more difficult, more abstract; it is hard to get people excited about theoretical algorithms. But still, how many books do you know that tell exciting stories of computer science to the public? Compare that to physics, even to string theory: there are books for the public about string theory!

There are books about computer science for the public, but they often concentrate on things like startups and successful companies in computer science.

So the public does not have an appreciation for computer science. Now, how does public appreciation translate into Congressional support and funding? That is not so obvious, but when you go to Congress it does help to have background appreciation for what your discipline has done. Computer science has not built that background appreciation; we have hunkered in our own little corner. We have done great research and we built amazing infrastructure, but if you ask the public what they know about computer science, they will list the Internet, and very little other than that. They know very little about research challenges in computer science.

So does that mean we'll be seeing stories in the New York Times *about finite model theory soon?*

It is an art to tell the story in a way that people find interesting. I heard David Harel give a wonderful talk to the public about what computers cannot do. And he is one of the few people that have been able to write very cogently about theoretical computer science, about computability, about complexity theory. Now, of course, he doesn't talk about the complexity of Ehrenfeucht games. When people write about quantum theory for the public, they don't present all the gory details of Schroedinger's equations. We need to know how to tell our story. We have not developed this set of skills. I think it is more difficult for us than for other disciplines, because there is something about the natural world that people inherently find more interesting than abstract and artificial worlds such as ours.

People are starting to think about how to tell our story. We rarely publish in *Science* and *Nature*. Should we try to publish there? Is it a good idea? How do we do it? I am hearing a conversation that I haven't heard until very recently, partly driven by the current crisis.

What words of advice do you have for database groups in academia who would like to improve their group's standing?

We now have interesting tools to evaluate the success of work in the long term. Things like Citeseer and Google Scholar suddenly give us a view that we could not have had before. One thing that we discovered from these tools is that we are actually very poor in predicting the long term impact of work. There is very little correlation, for example, between the best paper awards that we give and the test-of-time awards that we give. Sometimes, miraculously, we have the same paper win the best paper and the test-of-time awards. But that is the exception rather than the rule. So I think people should focus on just doing the best work they can.

What you just said implies that the low acceptance rates at conferences now are actually a problem, because we may be pruning out those papers that would win the ten-year paper award.

I think the low acceptance rate is a terrible problem. I think that the idea that we are raising the quality is nonsense. I think that actually the quality goes down. I think we are very good at selecting about one third to one fourth of the papers; we do a pretty good job there. As we become more selective, the program committee gets larger, the whole discussion gets more balkanized, and the whole process gets more random. Nobody gets a global view. The program committee used to look at the whole set of papers and there was some consensus; people would argue only about the marginal papers. Now the whole enterprise is so large that it is effectively broken into several subcommittees, and I have no confidence that we are really selecting the best papers. I have heard other people say that we are encountering the problem that we deal with in our own research: scalability. Conferences are not scalable. They work nicely with up to roughly 300 submissions and a certain size of program committee. When you try to scale it up, very good papers get lost. It becomes more political. I think we are being held back by our own success.

What would you propose as a remedy?

We are very unique among all the sciences in how we go about publication. We have these selective conferences. (People have stopped calling them “refereed conferences.” They are not really refereed. You don’t get good referee reports.) Our conferences worked well in the world we used to inhabit. We assume that because they worked, they are scalable; but there are reasons to doubt the scalability of this model.

I don’t have a good solution to this problem. We don’t even have a good forum in which to discuss the problem. It’s not just a problem for one part of computer science, it is a problem for all of computer science. How can computer science go about changing its publication culture? Are there areas that move just as fast as we do, and have journal papers and conferences, but conferences are not the primary vehicle? I have questions about the basic model of scholarly publications. And I find it fascinating that it is difficult to have a conversation about this on a big scale, and make changes on a big scale. We are very conservative. It is interesting that computer science has been one of the slowest disciplines to move to open access publications. Other disciplines are way ahead of us in using online publications.

So you must not be thinking of ACM SIGMOD’s DiSC and Anthology?

Those are online repositories of printed publications. When you go to the SIGMOD conference, you get a monster proceedings volume. It is expensive to produce it. Do you need it?

We don’t do that anymore. Now you get a DVD, and you can buy the printed version if you want.

Right, actually, I find I cannot store DVDs. I am not organized enough to store DVDs. All I care about…

I want it on the Internet. If I have it there, I don't need a DVD. At least *books* are thick enough that you can put them back on the shelf and you can find them later. I have yet to see anybody organize DVDs in such a way that you can find a DVD when you want it. So I just want the proceedings on line. I don't know why conferences give people a DVD of the proceedings. Maybe during the conference you can use it, and during a talk you can open the paper. But long term, I don't even keep the DVDs any more.

To back up just a moment, it sounds like you are talking about moving us more towards a journal culture?

I'm raising the issues. I think we had a model that worked successfully for about 30 years, but now we see cracks in the foundations. We ought to rethink how we do it. Right now, people try to fix things incrementally by having a larger conference with a bigger program committee, a two-level PC, a three-level PC. Maybe we need to rethink the way we do scholarly communication in our discipline.

As the number of really good database researchers continues to grow exponentially in the US (because we all graduate more than one PhD student in our lifetime), while the available funds for research remain constant, what changes do you foresee in the way research is carried out?

Exponential growth is never sustainable---we know this. During the period of exponential growth, we all get very excited, and we think this will go on forever. Nothing goes on forever. Exponential growth always hits some kind of a ceiling. We have been hitting our ceiling.

So funding is going to be scarcer, and that will translate to fewer graduate students. If the graduates cannot find good jobs, that will translate to fewer graduate students. Fields go through periods of growth and then some fields decline after that. We are too young to have seen this ebb and flow of things, so we think our field is always in growth. We had a period of very heady years: we had the late 90s, which were a very atypical period. There was an Internet bubble outside, but in some sense there was also a bubble inside our discipline. We still have to come to grips with that. What is the realistic size of our discipline? That will depend on the interest by students; that will depend on the funding available. The fact that we want to have more students ultimately is not the only factor that will determine what is going to happen. There is a big world out there, and we have to learn to roll with the punches.

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

I find that the things ultimately that I have success with are the things that I find at the time just to be an enjoyable piece of research. When I did research that I really enjoyed and that I thought was beautiful, very often it became the piece of research that had long term impact. I have to admit I had some beautiful papers that only I loved and nobody else cared about; generally we are not very good at predicting the ultimate success of our own work. There is a famous story about a party at the Cavendish Lab in Cambridge around 1900, where the physicists toasted, "To the electron: may it be of no use ever!" Even technical people, scientists, are not good at predicting the ultimate impact of their own work. So do the work that you think has lasting power; some will last, and some will not, but at least you will have fun in the process.

Among all your past research, what is your favorite piece of work?

Paul Erdos, the great discrete mathematician, had a concept of *God's Book of Proofs*. He said that mathematicians produce many proofs, but once in a while there is a proof so beautiful that

God says, “Ah, this one I didn’t see coming. This one is so beautiful that I am going to put it in my own book.” Long term, the things that I appreciate are the things that are the most beautiful, the things that have the most aesthetic value---sometimes these are the things that also have the most long term impact, but not necessarily. I have a few proofs that I would consider submitting to God for his book of proofs.

My work in verification was about translating from linear temporal logic to automata; it established a new connection, and I like that work very much. The thing that drove me in the beginning was the aesthetics of it. There are some results in finite model theory that have not had as much impact, but I think they are very elegant aesthetically. I recently put on my web page two slogans. One says, “Theorems Are Forever,” and the other says, “The proof, the whole proof, and nothing but the proof.”

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

I would like to write a textbook on teaching logic in computer science.

Oh, that would be great!

There is a course I have been teaching for many years, about logic from a computer science perspective. It is very algorithmic; I make a lot of effort to convey to students why logic is important in computer science. I would love to have the time to write such a textbook.

When you say it is algorithmic, do you mean that you spend a lot of time on automated proof theory?

For example, we talk about propositional logic, and we talk about satisfiability, both from a complexity point of view and algorithmically. The course has a lot of programming, and for the course project the students write a satisfiability solver. We talk about databases and first-order logic as a query language. We spend quite a lot of time thinking about query evaluation. We don’t do SQL, but the students understand the concept of formulas as queries. There is a deep connection between logic and computer science. There are some textbooks that try to make this connection, with a focus mostly on verification, but I think the connection between logic and computer science is much deeper than that.

If you could change one thing about yourself as a computer science researcher, what would it be?

I would be better organized. I am not a very organized person. My office continually looks like a hurricane just passed through it. My time management skills are not very good. I would love to be better organized.

But a hurricane did *just pass through your office, didn’t it?*

Both Rita and Katrina went a little north of us.

Thank you very much for talking with me today.

It has been a pleasure.