Georg Gottlob Speaks Out

on How Computer Science is a Continuation of Logic by Other Means, How Change Makes You Live Longer, How to Improve the Status of Computer Science, How to Interest Practical People in Complexity Theory and Database Theory, and More

by Marianne Winslett

Georg Gottlob
http://web.comlab.ox.ac.uk/oucl/people/georg.gottlob.html
(see also http://benner.dbai.tuwien.ac.at/staff/gottlob/)

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are at the SIGMOD 2006 conference in Chicago, Illinois. I have here with me Georg Gottlob, who is a professor of computing science at Oxford University. Before that he was a professor at the Technical University of Vienna for many years. His research interests lie in database theory, logic, AI, and complexity. Georg is a co-founder of Lixto Corporation, and until recently he was the editor in chief of AI Communications*. Georg's PhD is from the Technical University of Vienna. So, Georg, welcome!*

Georg, as a young man, you worked as a night train conductor, you wrote a booklet on the catacombs of St Stephen's Cathedral in Vienna, and you were very interested in psychology. How did you end up in computer science?

I was interested in psychology and I considered studying medicine, but out of laziness I started to study mathematics. In high school, mathematics didn't take much time to learn. It was very easy for me, so I thought, why shouldn't I continue and have a nicer life? For medicine, you had to study very big books and do other difficult things. I started mathematics, and I liked analysis, but was more interested in algebra, logic, and discrete mathematics. Eventually I saw that there is more of this kind of mathematics in computer science, such as automata theory, formal languages, and complexity theory. Eventually, after having done almost all the mathematics exams, I switched to computer science.

You typically change your main research focus every few years, moving between databases, AI, complexity, and logic. Would you recommend this form of research nomadism to others?

This form of nomadism is very nice for *me*, at least. I could recommend it to others; it makes life more interesting in some sense. You switch between topics, but you come back, of course, to the same set of research communities. You learn a lot more by switching. When you have been in a field, and you come back two years later, many things have happened there, so it is not boring. You can bring ideas from one area to another, so it is actually rather nice. There is a drawback too: people will ask you to review papers from three different fields, and people will ask you to write letters of recommendation in multiple areas, so that is a drawback.

You are probably as well known in the AI and computational logic communities as in the database theory community. What are the connections and differences between these communities and fields?

The differences are probably the specific goals, the narrow goals. But I see a lot of connections, and the basic thing is that they are all connected through logic. For me, logic is very important, and for me, the entire field of computer science is a continuation of logic by other means. Logic is a very important component of databases. If you answer a query, the answer should be a logical consequence of the database in some sense. And in AI planning, your actions should be in some sense logically related to what you know, to your knowledge. So you need forms of reasoning in both areas, and they are not forms of reasoning that correspond to classical logic. For example, we have the closed-world assumption in databases.

How does chairing IJCAI compare to chairing PODS?

IJCAI is a huge conference, PODS is a smaller conference. In IJCAI we have more than 1,000 submissions. So it's unthinkable that you read all of the very good papers as when you are an IJCAI program chair. As an IJCAI program chair, you have to be a very good organizer. I'm not such a good organizer, but still when I chaired IJCAI in 2003, I tried to do my best and fortunately I succeeded. You must know people, you must organize your program committee, you must try to get many people to cover areas that you don't know, and so on. For PODS, you don't have to organize so much; the community is much smaller, so it is easier to pick a good program committee.

Why isn't computer science considered on a par with the "old" disciplines like math, physics, and chemistry?

I think there are several reasons for that. One reason is certainly that computer science is a very new discipline, compared to disciplines that are two thousand years old. Of course, there is also the grant money. Physicists are very strong in getting lots of funding and they don't want computer science to take that funding away from them. On the other hand, the popular view of computer science, of computing, is not necessarily a view of computing *research*. The icons of computing are not scientists, but are the big entrepreneurs. That is not so in physics or in chemistry. But I see a slight change, and one symptom or sign of the change is that Academies of Science, who never in the past considered computer science to be one of their own disciplines, are starting to include it. For instance, the German Academy of Science Leopoldina starts to have its own section on information processing. And the European Academy of Science is starting to have its own group on computing. So there are good signs, and I think eventually computer science will end up being on par with the other sciences.

Is there some way that we could leverage the fact that we have these icons that are so well known?

We computer scientists have to promote our scientific ideas a little bit more. I think we have to be a little bit more oriented toward the popular audience in presenting our ideas. If you look at journals like *Science* and *Nature* and magazines such as *New Scientist or Scientific American*, you find a lot of articles on physics and biology and so on, but very few featured articles on computing. I think more popularization is needed so that people understand that computer science is not just entrepreneurship, and has a real science behind it.

You were a professor at the Technical University of Vienna for 18 years, with a well-endowed chair there. Many people thought you would stay there forever, so what led you to move to Oxford?

There are several reasons for this. First, Oxford is a very nice place. It is a beautiful place, and it is a great university in my opinion. The students are fantastic. Actually, I have to say that the best students in Oxford are probably at the same level as the best students in Vienna. But the average student in Oxford is so much better than in Vienna, and that is due to the very strict and severe selection they have at Oxford. Of course, we also have fewer students at Oxford. In Vienna, we had classes of 500 students. You probably cannot imagine this as an American.

I certainly can--- we have computer science classes of 700 students at the University of Illinois!

Do you? So you know what it means---teaching a programming class to 500 students. After the class, you are tired, but the students don't let you out of the room, they want you to explain the errors in their programs and so on. At Oxford, we don't have all that. We have small classes, very bright students.

And there are other reasons why I moved to Oxford. I like Vienna very much (in particular, the Technical University, where I now have an adjunct position), but I also like the change. After 18 years, you feel it is time to change. As I get older, time is passing much faster. There is a psychological reason for this, namely, your life is becoming more monotonous. As a child, every year, every month, there are so many changes, you get new teachers every year, you get into another grade and so on. Once you are stable in a job, you do always the same thing or similar things, and in retrospect, it seems like much less has happened. So, if you make a change to your life, then you will also get a stronger feeling of time passing, and you will feel like you live longer. So it is for a longer life that I went to Oxford.

Now you have done research in four countries with completely different university systems: Italy, Austria, the US, and now the UK. How do these university systems differ?

In my opinion, the main difference is between the Anglo-Saxon countries, like England and the US on one hand; and the continental European system like Italy and Austria, which are still different from one another, but are comparable in some sense. The main difference is the entrance to a scientific career, namely, at the level of assistant professor in the US, or at the level of a lecturer in the UK. Once you are an assistant professor or a lecturer, whether you can get promoted or not depends only on your work. It doesn't depend on anything else. You are in a tenure track position. The tenure track doesn't exist in most countries of continental Europe. In Austria, for instance, if you are hired as an assistant professor, you must switch universities in order to be promoted. I think that is not very good. I prefer the Anglo-Saxon system, where it is maybe a little more difficult to get an assistant professor job, but once you have it, then you can plan your life in some sense and you know you can succeed if you work hard.

You are such a proponent of change that I would think that you would enjoy a system that encourages people to work at multiple universities.

Yes, I am personally a proponent of change, but many people are not so. They want to have a certain stability. Some young people in continental Europe don't want to work at a university because they think the jobs there aren't secure enough. I personally am not so much interested in security, but I can understand that others are. We lost several potential assistant professors at the Technical University of Vienna for that reason. Many people don't apply to Vienna who would apply for a junior faculty position in the UK or in the United States, because in the UK or US they can get a secure job.

You are interested in data exchange. Tell us about that.

Data exchange is actually a very old problem, and there is a renewed interest in it. Data exchange is related to data integration. In data integration, the main goal is that you ask a query to heterogeneous databases, and the query is processed in a distributed way, then the results are collected together and you get the answer. In data exchange, the main goal is to take data present in one database and materialize it in another database with a different schema. Data exchange involves slightly different problems from data integration; since you want to materialize the data, but you have columns that don't exist in one of the databases, and you have different schemas, you need to deal very much with null values and how to satisfy integrity constraints when you transfer data. This is a very interesting problem. It's a very theoretically demanding problem. It's a problem that the industry is very interested in now. For instance, the IBM Almaden labs already have a system called CLIO that is used for data exchange. So I think our research on data exchange can influence the industry.

Why does computational complexity matter in the database field?

Complexity is one of the major problems in in querying databases, and in computer science in general. If you have an information system, you want to be able to guarantee that your users will get their answers in reasonable time, without having to wait two years for an answer. If the information system is very good, you can guarantee this. It is not sufficient to just say that some queries may be slow, but in general the system is very fast. That is not a good guarantee, because who knows which query is which kind? You want to guarantee performance, and complexity theory is the right tool to do that. You can say, okay, these algorithms are linear, that is very good; or you can say this problem is, say, NP-hard.

In practice people say that complexity theory doesn't interest them. But I have found a way to convince them that complexity theory is important, just by renaming it! If you go to a database practitioner, a systems person, and tell him or her, "Your system, your query language, doesn't scale," then they will get nervous very quickly. And it is actually the same as saying that their approach has a high computational complexity.

Some of our readers might think, "Well, you just submit the query and you get the answer back in seconds, so what's hard about it?"

Quick query answers are possible in simple cases with well designed databases and very good query optimizers, but generally they are not possible. If you want to solve problems that have many constraints and sub-queries, it is quite hard to get an answer. Just ask the people who deal with constraint processing or constraint databases. There they are exasperated, they don't know how to answer these queries, how to get to a good solution to those kinds of constraint problems.

As long as we are in very classical databases and consider short and simple queries, then things may work well. But if you go slightly to other domains, and want to handle that with database technology, it doesn't succeed.

There are so many hard problems, even just with conjunctive queries. Queries are getting longer and longer because of views. We query a view, and the view itself is a query, so if we resolve that, we get a very long query. And these long queries are very hard to answer, computationally hard.

You also have been working on hypertree decompositions and their applications in database theory and AI. What are hypertree decompositions?

Any query can be described by a hypergraph, and the hypertree width of a query is the measure of the degree of cyclicity in the hypergraph of the query. If the query is acyclic, then its hypertree width is 1. Now, query processing is, in general, NP-complete, but acyclic queries are easy to handle. Unfortunately, in practice, acyclic queries don't occur too often. But we observed that in practice most of the non-acyclic queries are nearly acyclic, with a hypertree width of 2 or 3. With Nicola Leone and Francesco Scarcello, we have developed algorithms for efficiently processing queries which are nearly acyclic. In some sense, we can fight complexity through this notion of hypertree decomposition.

Most of the web is still formatted in HTML. Why don't we see more XML data on the web---is XML a good data model?

That is a very interesting question. I remember that in 1995 some people predicted that in the early 2000s most of the web would be in XML, and everything would be annotated in XML. But what we see is the contrary; people are still producing huge amounts of HTML pages, and you almost never find a page in XML. XML is a very good data interchange format, but it has failed so far as a format that people generally use for putting information on the web.

There are, in my opinion, several reasons for that. One reason is that it is so much easier to write an HTML page and right away see what you get. When my son and my daughter were 13, they perfectly understood HTML. They could put it on the web, they could change HTML code, but they would not have been able to write DTD, or an XML DTD, or an XML schema. HTML is much more user friendly. The other reason is that in XML you have to use tags, but what do these tags mean? For one person, one tag means one thing, for another person it means another thing. There is no universal meaning attached to the tags. There may be some conventions, and some XML styles that two companies can use to exchange data, but there is no general meaning for the tags, and this will not happen until we have the semantic web. Basically, there is not much reason to put information in XML on the web (with the exception of the very restricted RSS format). But XML is a very nice language for exchanging data.

Do you believe in the semantic web?

Yes, I believe very much in the semantic web. I think the semantic web will be the Next Big Thing, the next revolution, and it will change our lives, just like the web did. The web did change our life, you have to admit that, and the semantic web will change our life in a similar way. That is my prediction.

Why hasn't it happened yet?

It hasn't happened yet because it must happen through companies. I think we are almost at the point where it will happen, I believe it will happen very soon. The lead will be taken by the big web players, like Google, Yahoo!, Microsoft. Some day soon, one of these big companies will announce that their search engines will now use ontological search based on semantic annotations attached to web pages, and if you want your web page to be found very efficiently, then please annotate it. Then people will do the annotations and we will have the semantic web. Of course, we will probably have another semantic web after the one which is now standardized, but it will be very similar. People will annotate their pages in a semantic way using nice tools offered by Microsoft, Google, Yahoo!, and other companies, and then this information will be put into the search engine databases and indexed on those annotations.

What do you think is currently hot in database theory?

There are many interesting problems in database theory. Data extraction still poses a lot of important problems. For example, the web is in HTML, but corporate data are needed in a more structured format. So how do you convert large amounts of HTML into structured data? That is what the Lixto tool from our startup company does, but while we were building it we discovered so many interesting problems. For example, can complex extraction patterns be learned? How can you best extract data from PDF? Database theory can address those questions, so they are not just practical questions. How can you recognize hierarchically organized data? How can you recognize table structures in a PDF document? These are not just problems for systems people. There can be a good theory, and that theory can help us to build better systems.

Of course there are other theory problems relevant to databases. Let me give you just one that has to do with privacy. Given a view, can we in some sense describe all the queries over the original database that we can answer using just this view? That is an old problem in some sense, but it is also a new problem because nobody has asked the question in these particular terms and little work has been done on it. Victor Vianu is working on it now, and Alan Nash, and a few other people. I would like to work on that problem.

And there are of course plenty of other interesting problems in database theory. The interest in the PODS conference is growing, so it is a good sign. We have been getting more submissions, and high attendance, so it is a very lively field.

Would you advise a gifted youngster to become a computer scientist today?

Definitely. I have been a computer scientist now for 25 years and I never regretted it for a single second in my life. Computer science is a very beautiful field. One of the big advantages of computer science is that once you study computer science, it does not determine what you have to do in your life, because it is such a rich and diverse field. If you study computer science you may end up as a mathematician; most of the time I do mathematics. But you might also end up as a graphics designer, or an electronics engineer. Or you might end up as a sociologist, because if you do computer supported cooperative work, you have to study how people interact. So if you study computer science, you don't choose too early in life a particular branch of a field that offers so many possibilities. This is helpful especially for young people who don't always know their talents; in computer science, they still have time to develop them.

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

My advice is to change a little bit. If you are a mid career computer scientist, then you have done something for a long time. I enjoy changes between different areas, and then coming back to one's area with different perspectives, and I think that is very good. Change places; do not be attached too much to one place.

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

Apart from hypertree decompositions, which I already described, it is certainly the work on semi-structured data. For instance, finding polynomial algorithms for the full XPath language was a challenge. Developing hypertree decompositions was cool. I also enjoyed building up a theory and algorithms for data extraction, and realizing them in the Lixto system.

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

I would like to study quantum computing. I have never had the time to study it, I don't know how it works very well. I would have to study first a little calculus, all these things that I have forgotten, and then study quantum computing. I think quantum computing is a challenging area, and there are nice algorithms.

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

It would be to be less distracted by daily business and short term needs. I am very susceptible to them. I have some long-term plans that I haven't realized so far, simply because I get continuously distracted by short-term needs. I should say to myself, "Why should I always be on top of everything for short term things?" It is much better sometimes to work on long-term goals, and I would like to have more concentration on them.