Database Theory, Yuri, and Me

Jan Van den Bussche

Hasselt University and transnational University of Limburg, Diepenbeek, Belgium.

Abstract. Yuri Gurevich made many varied and deep contributions to logic for computer science. Logic provides also the theoretical foundation of database systems. Hence, it is almost unavoidable that Gurevich made some great contributions to database theory. We discuss some of these contributions, and, along the way, present some personal anecdotes connected to Yuri and the author. We also describe the honorary doctorate awarded to Gurevich by Hasselt University (then called Limburgs Universitair Centrum) in 1998.

Dedicated to Yuri Gurevich, the "man with a plan", on his 70th birthday.

1 Database Theory

The theory of database systems is a very broad field of theoretical computer science, concerned with the theoretical design and analysis of all data management aspects of computer science. One can get a good idea of the current research in this field by looking at the proceedings of the two main conferences in the area: the International Conference on Database Theory, and the ACM Symposium on Principles of Database Systems. As data management research in general follows the rapid changes in computing and software technology, database theory can appear quite trendy to the outsider. Nevertheless there are also timeless topics such as the theory of database queries, to which Yuri Gurevich has made a number of fundamental contributions.

An in-depth treatment of database theory until the early 1990s can be found in the book of Abiteboul, Hull and Vianu [1]; Yuri Gurevich appears nine times in the bibliography.

A *relational database schema* is a finite relational vocabulary, i.e., a finite set of relation names with associated arities. Instead of numbering the columns of a relation with numbers, as usual in mathematical logic, in database theory it is also customary to name the columns with *attributes*. In that case the arity is replaced by a finite set of attributes (called a *relation scheme*). A *database instance* over some schema is a finite relational structure over that schema, i.e., an assignment of a concrete, finite, relation content to each of the relation names. So, if R is a relation name of arity k and D is a database, then $D(R)$ is a finite subset of U^k, where U is some universe of data elements. The idea is that the contents of a database can be updated frequently, hence the term "instance". We will often drop this term, however, and simply talk about a "database".

For example, consider a relation name Hobby of arity 3; we associate to it the relation scheme {name, hobby, location}. The intention is that an instance will store tuples (n, h, l) where n is the name of a person who has a hobby h and he performs this hobby in location l. Then a concrete instance will have a relation Hobby containing tuples like (john, birdwatching, lake) and (mary, violin, town hall).

2 Typed Template Dependencies

Usually we do not want to allow completely arbitrary relation contents in our database instances; we will typically expect certain integrity constraints to be satisfied, depending on the application. The early years of database theory focused on integrity constraints expressible in first-order logic, called "dependencies", and studied the implication problem for classes of dependencies, i.e., fragments of first-order logic. Not surprisingly, this new field of theoretical computer science attracted experts on logical decision problems, such as Gurevich. Gurevich and Lewis [27, 28] settled the undecidability of the implication problem for the class of "typed template dependencies".

For an example of a template dependency, consider, for example, the following hypothetical (and admittedly rather weird) integrity constraint on the Hobby relation: if two people each perform their own hobby in some common location, they also perform there each other's hobby; in short, common location implies common hobby. We can write such a dependency in the following syntax:

$$\mathrm{Hobby}(n_2, h_1, l) \leftarrow \mathrm{Hobby}(n_1, h_1, l), \mathrm{Hobby}(n_2, h_2, l).$$

The semantics is that of implication, where all variables are assumed universally quantified.

The implication problem for template dependencies then is: given a finite set Σ of template dependencies and another template dependency σ, decide whether each database instance satisfying all dependencies of Σ also satisfies σ. The typed version, which is easier but still shown undecidable by Gurevich and Lewis, restricts attention to relations with pairwise disjoint columns.

Interestingly, Gurevich and Lewis were working on this result concurrently with Moshe Vardi [36, 37]. To obtain his sharpest results, Vardi could apply work by Gurevich and by Lewis on the word problem for semigroups [22, 29].

3 Database Queries

One of the main purposes of a database is to query it. For many good reasons into which we cannot go here, the answer of a query to a relational database takes again the form of a relation. For example, on our Hobby relation, we could pose the query "list all pairs (n, h) such that n performs hobby h in some location where nobody else performs any hobby in that location".

So, in the most general terms, one could define a query simply as a mapping from database instances to relations. But to get a good theory, we need to impose some criteria on this mapping. First, it is convenient that all answer relations of a same query have the same arity. Second, the basic theory restricts attention to answer relations containing only values that already appear in the input database. We call such queries "domain-preserving". Third, a domain-preserving query should be "logical" in the sense of Tarski [34], i.e., it should commute with permutations of data elements.[1] This captures the intuition that the query need not distinguish between isomorphic databases; all the information required to answer the query should already be present in the database [3]. Thus, formally, a query of arity k over some database schema $\mathcal{S}$ is a domain-preserving mapping from database instances over $\mathcal{S}$ to k-ary relations on U, such that for every permutation ρ of U, and for every database instance D over $\mathcal{S}$, we have $Q(\rho(D)) = \rho(Q(D))$.

For example, if the database schema consists of a single unary relation name, so that an instance is just a naked set, a function that picks an arbitrary element out of each instance is not a query, because it is not logical. The intuition is that the database does not provide any information that would substantiate favoring one of the elements above the others.

4 The QPTIME Problem

The definition of query as recalled above was formulated by Chandra and Harel [11, 12]. This notion of query comes very naturally to a logician; indeed, Gurevich independently introduced the very same notion under the name "global relation" or "global predicate" in his two seminal papers on finite model theory [23, 24]. These papers also widely publicized one of the most fundamental open problems in database theory, the QPTIME problem [13, 14, 30, 31, 35]: is there a reasonable programming language in which only, and all, queries can be expressed that are computable in polynomial time? Gurevich's conjecture is that the answer is negative. The QPTIME problem has been actively investigated since its inception, as can be learned from two surveys, one by Kolaitis from 1995 [32] and one by Grohe from 2008 [19]. We will get back to it in Section 8. The problem also nicely illustrate how database theory lies at the basis of the areas of finite model theory and descriptive complexity which grew afterwards.

5 Datalog vs First-Order Logic

In the 1980s, much attention was devoted to the query language Datalog. One of the toughest nuts in this research was cracked by Ajtai and Gurevich [4, 5].

A Datalog program is a set of implications, called rules, that are much like the template dependencies we have seen above. But an essential difference is that

[1] We mean here a permutation not just of the data elements that appear in some input database, but of the entire global universe of possible data elements.

the relation name in the head of a Datalog rule is not from the database schema; it is a so-called "intensional" relation name. Thus a Datalog program defines a number of new relations from the existing ones in the database instance; the semantics is that we take the smallest expanded database instance that satisfies the rules. For example, on our Hobby relation, consider the following program:

$T(x, y) \leftarrow \mathrm{Hobby}(x, h, l_1), \mathrm{Hobby}(y, h, l_2)$
$T(x, y) \leftarrow T(x, z), T(z, y)$

This program computes, in relation T, the transitive closure of the binary relation that relates a person x to a person y if they have some common hobby.

Since the transitive closure is not first-order definable [3, 17], the above Datalog program is not equivalent to a first-order formula. Neither is it "bounded": there is no fixed constant so that, on any database instance, the rules have to be fired only so many times until we reach a fixpoint. As a matter of fact, a non-first-order Datalog program cannot be bounded, as bounded Datalog programs are obviously first-order; an equivalent first-order formula can by obtained by unfolding the recursive rules a constant number of times.

The converse is much less obvious, however: every first-order Datalog program must in fact be bounded, and this is the above-mentioned celebrated result by Ajtai and Gurevich.

6 Metafinite Structures

In database theory, a relational database is considered to be a finite relational structure, and also in practice, relational database instances are finite. But still there is a gap between theory and practice in this manner, as in practice, relational databases do contain interpreted elements from an infinite structure, such as numbers, and queries expressed in the database language SQL can perform computations on these numbers. In a very elegant paper, Grädel and Gurevich [18] proposed the theory of metafinite structures as a way to close the gap. That theory later also inspired the development of the theory of constraint databases [33].

7 Honorary Doctorate

All the work described up to know happened before I had ever personally met Yuri. That would happen in May 1996, on the occasion of an AMS Benelux meeting at the University of Antwerp, Belgium, where I was working as a postdoc at the time. I had noticed that the famous Yuri Gurevich was scheduled to give an invited talk at the logic session, and since I had some ideas related to the QPTIME problem, I approached him and asked if he would be interested in having dinner in Antwerp together and talk mathematics. To my most pleasant surprise he readily accepted. It was my first personal encounter with Yuri and we spent an agreeable evening. He patiently listened to my ideas and made suggestions. Being a native from Antwerp I could give him a flash tour of the city and also knew a typical restaurant, things he could certainly appreciate.

Shortly afterwards, I would join the faculty of what was then known as the Limburgs Universitair Centrum in Diepenbeek, Belgium; nowadays it is called Hasselt University. The university was just preparing for its 25th anniversary in the year 1998, and there was an internal call for nominations for honorary doctorates to be awarded during the Dies Natalis ceremony. Given his fame in finite model theory and database theory, and given the pleasant experience I had had with Yuri, I nominated him before the Faculty of Sciences. Obviously he was such a strong candidate that my nomination was enthusiastically accepted. Thus in May 1998, Yuri received the honorary doctorate and became a friend of Hasselt University, and of me personally as well.

Appendix A contains a transcript, translated into English, of the nomination speech I gave (in Dutch) for the honorary degree. In the course of preparing that speech, I collected information from many people around Yuri. These people gave me so much information that much of it could not be used for the short and formal speech. On the occasion of Yuri's 60th birthday, however, Egon Börger organised a special session at the CSL 2000 conference in Fischbachau, Germany, and in a speech given there I could use that material, consisting mainly of anecdotes. Appendix B contains a transcript of that speech.

8 Choiceless Polynomial Time

Upon getting to know him better, I started to collaborate with Yuri on a scientific level. I remember a meeting on finite model theory in Oberwolfach in 1998, just a few months before the honorary doctorate ceremony, where he gave a talk on Choiceless Polynomial Time [8], a very expressive database query language in which only polynomial-time queries can be expressed. The language is nice because it borrows its high expressivity from set theory in a natural way, and also because it is based on Gurevich's Abstract State Machines (ASM [25]) . It is nice to see how Yuri's work on ASMs, originally disjoint from the QPTIME problem, is applied to that problem.

Yuri wondered about the precise relationship between choiceless polynomial time and the work that was going on in database theory, e.g., by Abiteboul and Vianu on "generic machines" [2]. We collaborated on that question and that led to our joint paper (with Andreas Blass) on ASMs and computationally complete query languages [6, 7]. In short, it turns out that choiceless polynomial time is the same as the polynomial-time fragment of a natural complete query language based on first-order logic, object creation [10], and iteration. We also showed that the "non-flat" character of choiceless polynomial time, be it through arbitrarily deeply nested sets, or through object creation, is essential to its high expressive power.

We note that extensions of choiceless polynomial time are still being actively researched in connection with the QPTIME problem [9, 16, 15].

A small personal recollection I have on this joint research is that, when working on the proof, I visited Yuri in Paris, where he liked to spend his summers. We worked at the stuffy apartment where Yuri rented a room, and in the evening, Yuri felt like going to the movies and asked me to suggest a good movie. I remembered my father telling me the week before that he had been impressed by the then-running movie "Seven years in Tibet" (with Brad Pitt). So I suggested we go to that movie, and indeed, the movie impressed Yuri and me greatly.

9 Finite Cursor Machines, Stream Queries

The most recent chapter in my interactions with Yuri was a great visit I made to him at Microsoft Research, Redmond, for a week in October 2006, together with my student Dirk Leinders. It was impressive to visit Microsoft Research and to see Yuri thrive in these surroundings. It was also interesting to visit Zoe and Yuri's beautiful house. At the time I was working on querying streaming data and had an idea for an ASM-based model for computing such queries. Yuri was interested and we had fruitful brainstorm sessions on the model, which came to be called "Finite Cursor Machines". This research led to two nice papers on stream queries [20, 21, 26]. I know that Yuri is still interested in modeling computation on data streams.

10 Conclusion

My life has been enriched in many ways through my encounters with such a great person as Yuri Gurevich. Yuri, I wish you much happiness on the occasion of your 70th birthday!

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Abiteboul, S., Vianu, V.: Generic computation and its complexity. In: Proceedings 23rd ACM Symposium on the Theory of Computing. pp. 209–219 (1991)
3. Aho, A., Ullman, J.: Universality of data retrieval languages. In: Conference Record, 6th ACM Symposium on Principles of Programming Languages. pp. 110–120 (1979)
4. Ajtai, M., Gurevich, Y.: Datalog vs first-order logic. In: Proceedings 30th IEEE Symposium on Foundations of Computer Science. pp. 142–147 (1989)
5. Ajtai, M., Gurevich, Y.: Datalog vs first-order logic. J. Comput. Syst. Sci. 49(3), 562–588 (1994)
6. Blass, A., Gurevich, Y., Van den Bussche, J.: Abstract state machines and computationally complete query languages (extended abstract). In: Proceedings International Workshop on Abstract State Machines. Lecture Notes in Computer Science, vol. 1912, pp. 22–33. Springer (2000)

7. Blass, A., Gurevich, Y., Van den Bussche, J.: Abstract state machines and computationally complete query languages. Information and Computation 174(1), 20–36 (2002)
8. Blass, A., Gurevich, Y., Shelah, S.: Choiceless polynomial time. Annals of Pure and Applied Logic 100, 141–187 (1999)
9. Blass, A., Gurevich, Y., Shelah, S.: On polynomial time computation over unordered structures. Journal of Symbolic Logic 67(3), 1093–1125 (2002)
10. Van den Bussche, J., Van Gucht, D., Andries, M., Gyssens, M.: On the completeness of object-creating database transformation languages. J. ACM 44(2), 272–319 (1997)
11. Chandra, A., Harel, D.: Computable queries for relational data bases. In: Proceedings 11th ACM Symposium in Theory of Computing. pp. 309–318 (1979)
12. Chandra, A., Harel, D.: Computable queries for relational data bases. J. Comput. Syst. Sci. 21(2), 156–178 (1980)
13. Chandra, A., Harel, D.: Structure and complexity of relational queries. In: Proceedings 21st IEEE Symposium on Foundations of Computer Science. pp. 333–347 (1980)
14. Chandra, A., Harel, D.: Structure and complexity of relational queries. J. Comput. Syst. Sci. 25, 99–128 (1982)
15. Dawar, A.: On the descriptive complexity of linear algebra. In: Hodges, W., de Queiroz, R. (eds.) Logic, Language, Information and Computation, Proceedings 15th WoLLIC. Lecture Notes in Computer Science, vol. 5110, pp. 17–25. Springer (2008)
16. Dawar, A., Richerby, D., Rossman, B.: Choiceless polynomial time, counting and the Cai-Fürer-Immerman graphs. Annals of Pure and Applied Logic 152(1–3), 31–50 (2008)
17. Gaifman, H., Vardi, M.: A simple proof that connectivity is not first-order definable. Bulletin of the EATCS 26, 43–45 (1985)
18. Grädel, E., Gurevich, Y.: Metafinite model theory. Information and Computation 140(1), 26–81 (1998)
19. Grohe, M.: The quest for a logic capturing PTIME. In: Proceedings 23rd Annual IEEE Symposium on Logic in Computer Science. pp. 267–271 (2008)
20. Grohe, M., Gurevich, Y., Leinders, D., Schweikardt, N., Tyszkiewicz, J., Van den Bussche, J.: Database query processing using finite cursor machines. In: Schwentick, T., Suciu, D. (eds.) Database Theory—ICDT 2007. Lecture Notes in Computer Science, vol. 4353, pp. 284–298. Springer (2007)
21. Grohe, M., Gurevich, Y., Leinders, D., Schweikardt, N., Tyszkiewicz, J., Van den Bussche, J.: Database query processing using finite cursor machines. Theory of Computing Systems 44(4), 533–560 (2009)
22. Gurevich, Y.: The word problem for some classes of semigroups (russian). Algebra and Logic 5(2), 25–35 (1966)
23. Gurevich, Y.: Toward logic tailored for computational complexity. In: Richter, M., et al. (eds.) Computation and Proof Theory, Lecture Notes in Mathematics, vol. 1104, pp. 175–216. Springer-Verlag (1984)
24. Gurevich, Y.: Logic and the challenge of computer science. In: Börger, E. (ed.) Current Trends in Theoretical Computer Science, pp. 1–57. Computer Science Press (1988)
25. Gurevich, Y.: Evolving algebra 1993: Lipari guide. In: Börger, E. (ed.) Specification and Validation Methods, pp. 9–36. Oxford University Press (1995)

26. Gurevich, Y., Leinders, D., Van den Bussche, J.: A theory of stream queries. In: Database Programming Languages: 11th International Symposium, DBPL 2007. Lecture Notes in Computer Science, vol. 4797, pp. 153–168. Springer (2007)
27. Gurevich, Y., Lewis, H.: The inference problem for template dependencies. In: Proceedings 1st ACM Symposium on Principles of Database Systems. pp. 221–229 (1982)
28. Gurevich, Y., Lewis, H.: The inference problem for template dependencies. Information and Control 55(1–3), 69–79 (1982)
29. Gurevich, Y., Lewis, H.: The word problem for cancellation semigroups with zero. Journal of Symbolic Logic 49(1), 184–191 (1984)
30. Immerman, N.: Relational queries computable in polynomial time. In: Proceedings 14th ACM Symposium on Theory of Computing. pp. 147–152 (1982)
31. Immerman, N.: Relational queries computable in polynomial time. Information and Control 68, 86–104 (1986)
32. Kolaitis, P.: Languages for polynomial-time queries: An ongoing quest. In: Gottlob, G., Vardi, M. (eds.) Database Theory—ICDT'95. Lecture Notes in Computer Science, vol. 893, pp. 38–39. Springer-Verlag (1995)
33. Kuper, G., Libkin, L., Paredaens, J. (eds.): Constraint Databases. Springer (2000)
34. Tarski, A.: What are logical notions? History and Philosophy of Logic 7, 143–154 (1986), edited by J. Corcoran
35. Vardi, M.: The complexity of relational query languages. In: Proceedings 14th ACM Symposium on the Theory of Computing. pp. 137–146 (1982)
36. Vardi, M.: The implication and finite implication problems for typed template dependencies. In: Proceedings 1st ACM Symposium on Principles of Database Systems. pp. 230–238 (1982)
37. Vardi, M.: The implication and finite implication problems for typed template dependencies. J. Comput. Syst. Sci. 28, 3–28 (1984)

A Honorary Doctorate Nomination Speech

This speech was given by me (originally in Dutch) at the Dies Natalis ceremony of Hasselt University (then called Limburgs Universitair Centrum), on 28 May 1998, to nominate Yuri Gurevich for a honorary degree. Note the speech was directed to a general audience.

Dear Guests:

Dear Professor Gurevich:

It is my honor and my pleasure to give a brief exposition of your life, your work, and your achievements.

In these days, the field of information technology (IT) receives plenty of attention. It is hard to imagine our present-day information society without IT: in our daily lives we live and work thanks to products and services that would have been either unaffordable, or simply impossible, were it not for IT. Computer science, as an academic discipline, profits from this success, but at the same time runs some risk because of it. Indeed, the danger is that those less familiar with computer science, view IT as an obvious technology that we just use when and were we need it. This purely technological view of computer science is too limited. Computer science is just as well an exact science, a relatively young one at that, and still in full growth, that investigates the possibilities and limitations of one of the hardest tasks for us humans: the design and programming of computer systems, in a correct and efficient manner. Logical and abstract reasoning are essential skills in this endeavor.

Now if there is one who is a champion in logical reasoning, it is our honored guest, professor Yuri Gurevich. Yuri was born in 1940 in Russia, and studied mathematics at Ural university. Already at the age of 24 he earned his doctorate, and four years later the Soviet state doctorate, which allowed him access to a professor position at the highest level. Thus he found himself at the age of 29 as head of the mathematics division of the national institute for economics in Sverdlovsk. Russian colleagues have told me that such a steep career was almost unthinkable in the communist Russia of the 1960s, also because Gurevich had refused to become a member of the party.

But it was indeed impossible to ignore the scientific results he had obtained during his doctorate in mathematical logic. As a young graduate Yuri Gurevich had been directed towards a subdiscipline of mathematics, called the theory of ordered abelian groups. Fortunately I do not have to explain this theory in order to show the depth of the results that Gurevich obtained. Normally one expects of a mathematician that he or she finds some answers to specific mathematical questions posed within the discipline in which he or she is active. Gurevich, however, developed an automated procedure—think of a computer program—by which *every* question about the theory of ordered abelian groups could be answered! One might say that he replaced an entire community of mathematicians by a single computer program. At that time, as well as in the present time, it was highly unusual that an entire subdiscipline of mathematics is solved in such manner.

In the early 1970s it becomes increasingly harder for Yuri Gurevich to struggle against the discrimination against Jews in the anti-Semitic climate in Russia in those years. When he hears that the KGB has a file against him, he plans to emigrate. Unfortunately this happens under difficult circumstances, so that his scientific activities are suspended for two years. Traveling via the Republic of Georgia, from where it was easier to emigrate, he finally settles in 1974 in Israel, where he becomes a professor at the Ben-Gurion University. Yuri amazed everyone by expressing himself in Hebrew at departmental meetings already after a few months of arriving.

During his Israeli period, Yuri Gurevich develops into an absolute eminence, a world leader in research in logic. We cannot go further into the deep investigations he made, nor into the long and productive collaboration he developed with that other phenomenal logician, Saharon Shelah. The clear computer science aspect of his earlier work is less present during this period, although some of the fundamental results that he obtains here will find unexpected applications later in the area of automated verification of computer systems. The latter is not so accidental: having unexpected applications is one of the hallmarks of pure fundamental research.

Along the way, however, the computer scientist in Yuri Gurevich resurfaces. Computer science in the late 1970s was in full growth as a new, young academic discipline, and Gurevich saw the importance of a solid logical foundation for this new science. In a new orientation of his career, he accepts in 1982 an offer from the University of Michigan as professor of computer science. Since then professor Gurevich, as a leadership figure, serves as an important bridge between logic and computer science. Partly through his influence, these two disciplines have become strongly interweaved. Of the many common topics where the two disciplines interact, and where Gurevich played a decisive role, we mention finite model theory, where a logical foundation is being developed for computerised databases, an important topic in our own theoretical computer science group here at LUC; the complexity of computation, which he gave logical foundations; and software engineering, for which he designed a new logical formalism in which computer systems can be specified in a natural and correct manner.

To conclude I want to say a few words about the man Yuri Gurevich. With all his obvious talents he remains a modest—I would even say, somewhat shy—person. A good friend of his said it as follows: "Yuri loves science more than himself in it." That is all the more a reason to put him in the spotlights today.

Dear professor Gurevich: for your great achievements in mathematical logic, and for your continued contributions to the promotion and development of logical methods for the benefit of computer science, the Faculty of Sciences proposes you as a honorary doctor at our university.

B Fischbachau Speech

This speech was given by me after the dinner at the end of the symposium in honour of Yuri Gurevich's 60th birthday, held in the charming Bavarian village of Fischbachau, on 24 August 2000, co-located with the CSL conference. I thank the many people who contributed the material for this speech. Their names are mentioned explicitly.

Dear computer science logicians, and, of course, dear Yuri:

My name is Jan Van den Bussche, and during the academic year 97–98, I had the pleasure of promoting an honorary doctorate for Yuri at my university, the University of Limburg in Belgium. Because at that time I did not know Yuri personally very well yet—happily for me this has changed by now—I solicited comments from various friends and colleagues of Yuri. Within two days I received many warm responses. Unfortunately, because my laudation speech had to be rather formal, I could not directly use many of the nice things said about Yuri, and I have always found it a great pity that they remained buried in my files. Therefore I am so glad with this occasion to bring a few anecdotes about Yuri out into the open.

But first I would like to tell how Yuri and I met for the first time. In April 95, then at the university of Antwerp in Belgium, I was thinking about some issues related to one of the many nice papers Yuri wrote with Saharon Shelah. I sent him an email with a technical question, and received a very friendly reply within half an hour, giving me his intuition behind my question, telling me that for the moment he was busy with other things, but that he would be happy to discuss the matter more deeply in Antwerp, where he happened to be going soon for an AMS meeting. So we planned to meet in his hotel lobby some late afternoon, which was a bit exciting because, although I knew his face, he had never even heard about me before that email. But immediately we were both at ease, making a long walk in the old city center, me showing him various sights. I remember that I was a bit afraid of overdoing it: maybe we were walking too long and he was tired of it, but being too nice to say it. Little did I know that Yuri himself is infamous for taking people on walks they don't know when and where they will end! Taking a walk is also one of Yuri's famous ways of dealing with difficult mathematical problems. At dinner we were talking about mathematics, and he was really giving my questions genuine thought. All in all, I was really struck by his friendliness, his generosity, his accessibility. These very qualities of Yuri have been mentioned to me independently by many people.

One of these people is Joe Weisburd, who was a student of Yuri in the old Russian days back in the sixties. Joe also told me the following:

> Yuri always looked very undaunted, very powerful and very confident, which was unusual at those times, unless you were a ranked Communist. And Yuri Shlomovich—wasn't. You also didn't dare to look powerful and confident if you were Jewish. And Yuri Shlomovich—was, even blatantly so in a city where public Jewish life was completely suppressed. His patronymic, Shlomovich, was a challenge, indicating that he wasn't adjusting to sounding more Russian, as was pretty popular then. The Biblical names of his newly born daughters was another challenge. And Yuri was absolutely unprecedented in the Jewish folk song, that he suggested to sing together at the banquet after an annual mathematical Winter School of our Math department.

Yuri became a full professor at the age of 29, which again was unheard of, let alone him being Jewish. Nevertheless, it became increasingly clear that he had to emigrate. Cunningly, he discovered that for some peculiar reason, it was easier to get permission to emigrate out of the Republic of Georgia, so he requested and finally received permission to transfer there, and eventually was able to emigrate. These were severe times; they even had to go on a hunger strike. Vladik Kreinovich, who met Yuri at a seminar in St. Petersburg during this period, told me the following:

> Yuri seemed to be undisturbed by the complexity of the outside life. He radiated strength and optimism, and described very interesting results which he clearly managed to produce lately, during the extremely severe period of his life. His demeanor looked absolutely fantastic.

Once free, Yuri devoted considerable energy to help other Soviet Jews with their emigration. This happened in Israel, and also later in the United States. Vladik, himself an emigrant, continues:

> Together with a local rabbi, Yuri formed a committee which became one of the grassroots activities that finally managed to convince the American public opinion that the Soviet Jews needed help, in a period when the political climate was on the left side. History books have been written about that period, and Yuri is not described there as one of the prominent and visible leaders of the American support campaign. However, this is only because he preferred not to be in the spotlights.

Arriving in Israel, Yuri's talent for languages was very useful. Baruch Cahlon, who shared an office with Yuri in Beer-Sheva, recounts:

> Yuri hardly spoke any Hebrew, but he had to communicate with all of us in this orient-modern language, which he seemed to love, but was yet unable to utter. My wife, who is a Hebrew teacher, used to tell me how Yuri would try to answer the phone when she would call. Believe me, it was funny. It didn't take long, however, and one

> day, during a department meeting, Yuri stood up and spoke fluent, almost flawless Hebrew. We were totally astonished. Just couldn't believe it. At the time, of course, we had not known about his love for languages. His interest in these disciplines far surpasses that of an average mathematician. But then, of course, Yuri isn't that either!

In a similar vein, Jim Huggins, one of Yuri's later students in Michigan, told me that Yuri would ask him questions about English such as "what is the difference in pronunciation of the words 'morning' and 'mourning'?", and they would try to find English equivalents of Russian proverbs. During a number of summers spent in Paris—Yuri likes the French way of life very much—he learnt to speak more than a mouthful of French as well. And last April, when we were together at a nice restaurant in Ascona, Yuri had fun translating the Italian menu to us!

In Israel, Yuri also gave new meanings to religious symbols. Saharon Shelah told me that Yuri once turned up at a logic seminar wearing a kipah, this is the Jewish religious cap. When asked about it, he replied "why, it is very convenient: it covers exactly my bald part!" Yuri, I am sorry, but I guess that by now even an extra large won't do anymore!

I finally must mention the constant source of support in Yuri's life provided by his wife, Zoe. Not for nothing, Yuri himself describes the period before he met Zoe as his "protozoan" period! By the way, Zoe is also a mathematician by education, and she is a hell of a computer programmer.

Dear Yuri, although you are now a professor emeritus, you are definitely not yet retired, and thanks to you we will continue to see a lot of exciting things coming out of Microsoft. I congratulate you with your sixtiest birthday, and am looking forward to the next sixty years!