On Kolmogorov Machines And Related Issues*

Yuri Gurevich
Electrical Engineering and Computer Science Department
The University of Michigan, Ann Arbor
MI 48109-2122, USA

I felt honored and uncertain when Grzegorsz Rozenberg, the president of EATCS, proposed that I write a continuing column on logic in computer science in this Bulletin. Writing essays wasn't my favorite subject in high school. After some hesitation, I decided to give it a try. I'll need all the help I can get from you: criticism, comments, queries, suggestions, etc.

Andrei Nikolayevich Kolmogorov died a few months ago. In recent years he chaired the Department of Mathematical Logic at the Moscow State University. In a later article or articles, I hope to discuss Kolmogorov's ideas on randomness and information complexity; here let me take up the issue of Kolmogorov machines and their close relatives, Schönhage machines. I believe, we are a bit too faithful to the Turing model.

It is often easier to explain oneself in a dialog. To this end, allow me to introduce my imaginary student Quizani.

- *Quizani*: I think you should introduce yourself too. Don't assume everyone knows you.
- *Author*: All right. I grew up in the Soviet Union and started my career in the Ural University as an algebraist and self-taught logician. In 1973, I emigrated to Israel where I did logic and taught at Ben-Gurion University. In 1982, I moved to Michigan and to computer science.

* "The Logic In Computer Science Column", Bulletin of European Assoc. for Theor. Comp. Science, Number 35, June 1988, 71–82; reprinted in "Current Trends in Theoretical Computer Science", Eds. G. Rozenberg and A. Salomaa, World Scientific, 1993, 225–234.

- **Q**: Now, why should I care about Kolmogorov machines? I am sure that neither IBM nor Apple produces them. What is wrong with good old Turing machines?

- **A**: I believe that the Kolmogorov model gives a better measure for time complexity. In particular, it is more appropriate for lower time complexities like real time or linear time. Of course, the two models give the same classes of computable functions and polynomial-time computable functions, but they are expected to define different versions of many other time complexities, like n^2 or $2n$.

- **Q**: How do you know that the Kolmogorov model gives a better measure of time complexity?

- **A**: Well, that is what this article is all about. But I should admit at the outset the circumstantial character of the evidence.

- **Q**: So what are Kolmogorov machines?

- **A**: Kolmogorov, or Kolmogorov-Uspensky, machines [Ko1, KU, US] are similar to Turing machines except that the tape can change its topology. I will use an unorthodox presentation of KU machines that seems more convenient [L2]; it is somewhat influenced by Schönhage's presentations of his machines [Sh]. The tape is a finite connected graph with a distinguished (active) node. The graph is directed but symmetric: If there is an edge from u to v then there is an edge from v to u. The edges are colored in such a way that the edges coming out from any one node have different colors. Thus, every path from the active node is characterized by a string of colors. The number of colors is bounded (for each machine).

  The neighborhood of the active node of some fixed (for every machine) radius is called the active zone. (One may suppose that the radius always equals 2.) For each isomorphism type of the active zone, the program specifies a sequence of instructions of the following forms:

  1. add a new node together with a pair of edges of some colors between the active node and the new one,
  2. remove a node and the edges incident to it,
  3. add a pair of edges of some colors between two existing nodes,

4. remove the two edges between two existing nodes,
5. halt.

Executing the whole instruction sequence creates the next configuration of the tape.

- **Q**: I don't understand why do you need the halt instruction. Some types of the active zone may be assigned the empty instruction sequence. If and when the active zone is of such type, the machine will stop.

- **A**: This is a legitimate approach, but apparently Kolmogorov considered the initial tape as the input and the final tape as the output [L2]. In your approach, the output cannot serve as an input for the same machine. Let us fix a set C of colors and consider the data type D comprising connected, directed, symmetric C-colored graphs with a distinguished node such that the edges coming out from any one node have different colors. If you want that every computable function f from D to D is computable by an appropriate KU machine M, then your approach cannot be used and the halt instruction is handy. The desired M may be allowed to use additional colors; it suffices to use the additional colors only for edges from the active node.

- **Q**: What did Kolmogorov and Uspensky prove about their machines?

- **A**: Kolmogorov and Uspensky write that they just wanted to comprehend the notions of computable functions and algorithms, and to convince themselves that there is no way to extend the notion of computable function. They consider first a more general tape that is a finite first-order structure of a fixed finite signature. The only restriction is that the number of relation instances involving any particular element is bounded. The more general machines can be easily simulated (real-time simulated in the sense of [Sh]) by the canonical machine described above.

  It seems that the thesis of Kolmogorov and Uspensky is that every computation, performing only one restricted local action at a time, can be viewed as (not only being simulated by, but actually being) the computation of an appropriate KU machine (in the more general form). In a sense, this is stronger than Turing's thesis. The only

theorem proved in [KU] is that every partial recursive function is KU computable.

- **Q**: Is there any evidence that the KU model is indeed more powerful than the Turing model?

- **A**: Grigoriev [Gr] exhibited a function real-time computable by some KU machine but not real-time computable by any Turing machine.

- **Q**: What does real-time computable mean?

- **A**: The following description is appropriate in the case of Turing or KU machines: A real-time algorithm inputs a symbol, then outputs a symbol within a fixed number c of steps, then inputs a symbol, then outputs a symbol within c steps, and so on. In the case of more powerful machines, one may want to speak about more complicated data items (than just symbols) as one-step inputs or outputs [PS]. In applications, the situation is much more difficult. There is no consensus among experts on real-time systems what real-time computability is.

- **Q**: How does Grigoriev exploit changing topology?

- **A**: The idea is simple. Prompted by input, the desired real-time KU machine builds a regular binary tree of some depth d, whose edges are labeled with 0's and 1's, and returns to the root of the tree. Then, reading a binary string of length d, the machine traverses the corresponding branch of the tree and outputs the labels. No Turing machine can carry out that task because the number of cells its heads can visit in at most n steps, from a given configuration, is bounded by a polynomial of n.

  (Grigoriev mentions that general devices with polynomial-access memory were introduced in [CA], and that his impossibility proof is similar to the proof of the main result in [PFM].)

- **Q**: Is KU linear time more powerful than Turing linear time?

- **A**: This is an open problem. In general, the notion of linear-time computability is very dependent on the computational model. A closely related class NL of functions KU computable in nearly-linear time $n \cdot \log n^{O(1)}$ is very robust [GS]. Instead of the KU model, one can use the Schönhage model, numerous RAM models, and so on and so forth. It

is conjectured in [GS] that not all NL functions are Turing computable in nearly linear time (the latter class was studied by Schnorr [Sr]).

- **Q**: Is Grigoriev's result the only hard evidence that the KU model is more powerful than the Turing model?

- **A**: I don't know any additional hard evidence and would appreciate any information from the readers on the issue.

- **Q**: What are additional advantages of KU machines vs. Turing machines?

- **A**: There is a number of theorems that have a nice, and sort of complete, form in the KU model. It is not known and probably not true that these theorems, in their exact formulations, survive in the Turing model. The following theorem of Leonid Levin, a former student of Kolmogorov who teaches now at Boston University, is a good illustration.

  Definition 1. Let $F(w) = x$ be a computable function from binary strings to binary strings. Say that an algorithm A conclusively inverts F if, given any x in the range of F, A computes an F-witness w for x and then runs F to check that $F(w) = x$. A diverges on inputs outside the range of F.

  Theorem 1 ([L1, L2])**.** *For every computable function $F(w) = x$ from binary strings to binary strings, there exists a KU algorithm A such that A conclusively inverts F and (Time of A on x) = O(Time of B on x) for every KU algorithm B that conclusively inverts F.*

  For example, F may extract the theorem from a proof in a fixed formal system, or F may be a function which, given a propositional formula x and a satisfying assignment for x, produces x.

- **Q**: This is very interesting. It follows that if SAT is polynomial-time decidable then Levin's algorithm for SAT is polynomial-time as well. The corollary has nothing to do with the KU model and should remain true for the Turing model as well.

- **A**: That is correct. One can prove that in the Turing model there exist inverting algorithms that are optimal in a weaker sense.

- **Q**: I would like to see the proof of Levin's theorem.

- **A**: Levin never published the proof, but he kindly explained it to me (in his usual telegraphic style): Identify machines with binary strings representing their programs. If p is a machine, let $T(p, x)$ be the time that the universal machine U needs for simulating p on input x. By analogy with what is known as Kolmogorov or information complexity (introduced explicitly or implicitly by Kolmogorov [Ko2], Solomonoff [So] and somewhat later Chaitin [Ch]), define the Levin complexity (this is my term) of a string w relative to x (and F):

  $L(w/x) = \min\{|p| + log(T(p, x)) : \text{Given } x, p \text{ computes } w \text{ and then runs } F \text{ on } w \text{ and finds out whether } F(w) = x\}$

  Here the logarithm is of base 2. Notice that every $L(w/x)$ is finite. Given x, the desired A tries all strings w in the order of $L(w/x)$ until it conclusively inverts F.

  Lemma 1. *All strings of Levin complexity $\leq k$ can be generated and tested in 2^k steps.*

  (It is not clear and probably not true that the lemma remains true in the Turing model, though the KU model is not the only suitable model; another suitable model is that of Schönhage [Sh].)

  Hence F can be conclusively inverted in time $2^{m(x)}$ where $m(x) = \min\{L(w/x) : w\ is\ an\ F - witness\ for x\}$.

  Suppose that some machine B conclusively inverts F in time $t(x)$. There exists a constant b, depending on B, such that $T(B, x) \leq b \cdot t(x)$. Then $m(x) \leq |B| + \ log(b \cdot t(n))$ and $2^{m(x)} \leq (2^{|B|}) \cdot b \cdot t(x)$

- **Q**: And how difficult is the lemma?

- **A**: The lemma is not very difficult. It is supposed that the universal machine U reads the given program p bit by bit. Consider the run of U on p and x until U halts or requires an extra bit of program or makes $2^{(k-|p|)}$ steps, whichever comes first. If U reads all of p and does not require an extra bit, say that p is a good program for x and k.

  The algorithm A generates and tries all good programs p in the lexicographical order. It starts by running U on program 000... . When

U halts or runs out of time, A knows the first good program. Let p be the good program generated and tested most recently. If p contains only 1's, the process is finished. Otherwise p can be presented in the form $q0r$ where r contains no 0's. In this case, A resets U and runs it on program $q1000...$ until U halts or runs out of time. This way A generates and tests the good program that is the successor of p in the lexicographical order.

For each good p, A runs U on p and x for at most $2^{(k-|p|)}$ steps. Notice that $2^{(k-|p|)}$ is the number of k-bit continuations of p. Since no two good programs have common k-bit continuations, the total simulation time is bounded by 2^k.

- **Q**: I wonder why do you think that Levin's theorem, in its exact form, is not true in the Turing model. We know that KU machines work better with trees. Are there additional advantages of KU machines used in the proof?

- **A**: Yes. A universal Turing machine has a certain number j of tapes, and the question arises of simulating machines with greater number of tapes. Even if it simulates only j-tape machines, there is a problem of counting steps without loosing time.

- **Q**: Allow me a naive question. Do you see any practical use for KU machines? Reconfiguring hardware has got to be ridiculously inefficient. Also, in order to build a tree (as required in Grigoriev's proof) in our 3-dimensional space, a KU machine must use very long edges and to arrange the cells in some manner available also to a TM with a 3-dimensional tape.

- **A**: Of course, KU machines are theoretical machines. But maybe it is possible to build them after all. Doesn't the human brain resemble somewhat a parallel KU machine?

- **Q**: I don't know. Let me raise another subject. You have mentioned Schönhage machines. Tell me about them.

- **A**: The Schönhage model [Sh] can be quickly described as the generalization of the KU model where the tape graph is not required to be symmetric (though Schönhage's description is very different). Thus, the

fan-in is not necessarily bounded, though the fan-out is still bounded by the number of colors.

- **Q**: But too many edges cannot enter the same physical location.
- **A**: Think about edges as pointers. Similar machines were considered by Knuth [Kn, p. 462–463].
- **Q**: But then the amount of information at one node may be unbounded. The addresses of pointers from a node v should be somehow stored at v. As the number of nodes grows, the addresses cannot be bounded.
- **A**: That is correct.
- **Q**: Did Schönhage introduce his machines as an improvement of KU machines?
- **A**: No. I believe that the three models - those of Kolmogorov, Knuth and Schönhage - were conceived independently. In contrast with the "competitors", Schönhage provided convenient syntax and proved some theorems about his machines. He proved, for example, that his machines can simulate Turing machines with infinite multi-dimensional tapes in real time.
- **Q**: I don't understand. Schönhage's machines generalize KU machines which generalize Turing machines. What is there to prove?
- **A**: Suppose that the simulated Turing machine has a two-dimensional tape that is infinite from the very beginning.
- **Q**: I thought that a Turing machine is supposed to be only potentially infinite, not actually infinite.
- **A**: All right. Then imagine a Turing machine with a two-dimensional tape which is always a square. If the machine needs to create another cell, it extends the current square to a larger square in one step. The simulating Schönhage machine should ensure that the eastern neighbor of node $(i, j+1)$ is also the northern neighbor of node $(i+1, j)$. It is not clear at all how to achieve that in the course of a real-time sim-ulation.
- **Q**: Please, define real-time simulations.

- **A**: Sorry, allow me to sidestep that issue; this article is getting too long. Let me only say that if the simulated machine computes in real time, then the simulating machine computes in real time too.

- **Q**: Why did Schönhage introduce his machines?

- **A**: Pointing out that "so far no unified and generally accepted measure for the time complexity of algorithmic problems has been established", Schönhage writes that his model "possesses extreme flexibility and should therefore serve as a basis for an adequate notion of time complexity".

- **Q**: Do you agree with his suggestion?

- **A**: Pragmatically speaking, the Schönhage model provides a good measure of time complexity at the current state of art (though I would prefer something along the lines of the random access computers of Angluin and Valiant [AV]). On the theoretical side, all good things said above about the KU model apply also to the Schönhage model. Still, I don't think that Schönhage's suggestion has sufficient theoretical justifications. What exactly does extreme flexibility mean? One possible formalization of his thesis is that every sequential computing device can be simulated in real time by an appropriate Schönhage machine [Gu]. But then the problem what is a computing device arises. You said, for example, that Turing machines with actually infinite tapes are not legitimate computing devices. I would be very interested to hear arguments for and against this or another formalization of Schönhage's thesis.

- **Q**: Are Schönhage machines more powerful than KU machines with respect to real time?

- **A**: That is a hard question; some progress is reported in [St]. Schönhage machines seem more powerful with respect to real time and linear time. They can multiply integers in linear time [Sh]; KU machines probably cannot do that. It is known that Turing machines cannot multiply integers in real time [CA, PFM].

- **Q**: Which of the two models, I mean the KU model and the Schönhage model, is philosophically, if you will, preferable.

- **A**: That is a hard question too.
- **Q**: Finally, do you call all this logic?
- **A**: Well, the theory of algorithms was traditionally a part of logic. Also, these are foundational questions, and I find it difficult to distinguish clearly between foundational and logical issues.
- **Q**: Some logicians sound imperialistic.
- **A**: I know, and I wonder what the readers have to say.

Thankful Acknowledgements

Andreas Blass, Kevin Compton, Naomi Gurevich, Leonid Levin, Arch Naylor, Georg Schnitger and Joel Seiferas commented on a draft of this article.

References

[AV] Angluin D. and Valiant L. G., "Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings", Journal of Computer and System Sciences 18 (1979) 155–193.

[CA] Cook S. A. and Aanderaa S. O., "On the Minimum Computation Time of Functions", Trans. Amer. Math. Soc. 142 (1969), 291–314.

[Ch] Chaitin G. J., "On the Length of Programs for Computing Finite Binary Sequences", Journal of ACM 13 (1966), 547-569, and "On the Length of Programs for Computing Finite Binary Sequences: Statistical Considerations", Journal of ACM 16 (1969), 145–159.

[Gr] Grigoriev D. Y., "Kolmogorov Algorithms are Stronger than Turing Machines", Investigations on Constructive Mathematics and Mathematical Logic. VIII, Notes of the Leningrad Branch of Steklov Math. Institute 60 (1976), 29–37.

[GS] Gurevich Y. and Shelah S, "Functions Computable in Nearly Linear Time", Amer. Math. Soc. Abstracts 7:4 (1986), p. 236.

[Gu] Gurevich Y., "Algorithms in the World of Bounded Resources", In "The Universal Turing Machine - a Half-Century Story", (ed. R. Herken), Oxford University Press, 1988.

[Kn] Knuth D. E., "The Art of Computer Programming", vol. 1, Addison -Wesley, Reading, MA, 1968.

[Ko1] Kolmogorov A. N., "To the Definition of an Algorithm", Uspekhi Mat. Nauk 8:4 (1953), 175–176.

[Ko2] Kolmogorov A. N., "Three Approaches to the Definition of 'the Quantity of Information' Notion ", Problems of Information Transmission 1:1 (1965), 3–11.

[KU] Kolmogorov A. N. and Uspensky V. A., "To the Definition of an Algorithm", Uspekhi Mat. Nauk 13:4 (1958), 3–28 (Russian); English translation in AMS Translations, ser. 2, vol. 21 (1963), 217–245.

[L1] Levin Leonid, "Universal Search Problems", Problems of Information Transmission 9:3 (1973), pages 265–266.

[L2] Levin Leonid, Private Communication.

[PFM] Paterson M. S, Fischer M. J. and Meyer A. R., "An Improved Overlap Argument for On-Line Multiplication", SIAM-AMS Proc. 7 (1974).

[PS] Preparata Franco P. and Shamos Michael Ian, "Computational Geometry: An Introduction", Springer-Verlag, New York, 1985.

[Sh] Schönhage A., "Storage Modification Machines", SIAM J. on Computing 9:3 (1980), 490–508.

[Sr] Schnorr Claus P., "Satisfiability is Quasilinear Complete in NQL", JACM 25:1 (1978), 136–145.

[St] Schnitger Georg, "Storage Modification Machines vs. Kolmogorov-Uspensky Machines: An Information Flow Analysis (Extended Abstract)", Manuscript, Penn State University, Dec. 1987.

[So] Solomonoff R. J., "A formal theory of inductive inference. Part I", Information and Control 7:1 (1964), 1–22.

[US] Uspensky V. A. and Semenov A. L., "Theory of Algorithms: Main Discoveries and Applications", Nauka, Moscow, 1987.