# Are computer scientists the sutlers of modern biology?[*]

Bioinformatics is indispensible for progress in molecular life sciences but does not get credit for its contributions.

Peter Schuster[†]

At present bioinformatics is indispensible for the life sciences. The data flood is enormous and cannot be inspected and analyzed anymore by human eye and brain. This fact alone initiates a new area of science and mathematics: The experimentalist records amounts of data, which escape all attempts of direct visualization and cannot be handled without extensive assistance by computers. Elaborate mathematical proofs are often so complex that they require at least partial help by computation. Burning questions arise in this context: "Can we trust computers?" or "Is large scale software running on our gigantic machines really free of bugs?" Different disciplines react differently to such issues: Mathematical purists are very reluctant to accept proofs by computer, theoretical physicists are most open minded and commonly rely on their gigantic computer programs, chemists in essence got acquainted with computational chemistry and are accepting theory with decreasing resistance, and finally biologists cannot do modern molecular life sciences without an impressive collection of computational tools.

Proofing – already existing – theorems by computers goes back to the nineteen fifties and has been generally accepted as a useful tool in pure and applied mathematics [1]. Since proofs for unsolved problems were done by computer assistance – the first popular example has been the four color problem [2,3] – the society of mathematicians is divided on the issue of proof by computer [4]. The arguments of the proponents of computer proofs are straightforward: The conventional proofs for many conjectures are so complex that they fail to be discovered by the unaided brain, and automated proofing provides an alternative that is cheap and has shown to be successful in numerous cases of problems already solved by conventional methods. The arguments of the opponents are really not less convincing: A proof to be intellectuable to the human mind has to be casted into a number of – logical – statements that are comprehensible. If the number of statements is so large that the proof cannot be understood in reasonable time, then the proof has to be rejected. A problem arises with most computer proofs, because the number of statements to be executed in performing formal proofs is so large that human step-by step tracking might last thousands of years and more. The conflict has been ignited by the computer solution of Johannes Kepler's conjecture on the solution of the ball stacking problem [4,5]: Thomas Hales succeeded to provide a computed proof for the conjecture, which required about 3 GByte memory space for code, input and output. The reviewers of the 250-page manuscript submitted by Thomas Hales and Samuel Ferguson to the Annals of Mathematics saw themselves unable to decide whether or not this monstrous proof is correct. They ended up exhausted in the year 2003 by concluding that the proof is certainly 99% correct. Evidently this is not sufficient for a mathematical proof. Finally an

[*] The essay has been published in *Complexity* **19**/4: x-xx (2014).

[†] Peter Schuster, Editor-in-chief of *Complexity* is at the Institut für Theoretische Chemie der Universität Wien, Währingerstraße 17, 1090 Wien, Austria. E-mail: pks@tbi.univie.ac.at

abridged version of the proof appeared in 2005 in this journal [6] and full details were given in an entire issue of Discrete and Computational Geometry [7]. Apart from the Sisyphus work like task to complete such a complex proof there is good reason for being skeptic about error-free hard- and software. To indicate the problem I give two quotations that I found in Thomas Hales article on formal proof [8]: “I have come to the conclusion that no microprocessor is ever perfect; they just come closer to perfection” [9] and “Don’t insist that every bug be fixed … When the programmer fixes a minor bug, he might create a more serious one” [10]. It is very hard to trim large computer codes to perfection, and a recent development, for example, makes even use of evolutionary methods to remove bugs from programs [11,12]: Improvements in the sense of removal of bugs are created by random variation and selection. In short, automated proofs will become more and more important in the future but computer scientists have to invest a lot more brain work in making computer codes more reliable but we should not forget that nothing that is sufficiently complex can be entirely error-free in a finite world.

Physicists and chemists use huge computer codes for large scale computations ranging from data harvesting and interpretation in high-energy physics to the application of quantum mechanics to calculations of molecular structures and to large scale simulations of stochastic models. The requirement of *absolutely error-free programs* has never been such a central issue as in computational mathematics. The reason is easy to understand: The ultimate goal of computation in physics and chemistry is to produce data that are more accurate and more reliable than those collected by empirical observation. It is necessary therefore only to check software for consistency and to clean the major bugs off the computer code. Numerical mathematicians provided and provide better and better algorithms and computer scientists implement more and more efficient codes for computational science. An example should illustrate this fact quantitatively: It is now commonplace that speed of computation and digital storage capacities are growing exponentially and they do this uninterruptedly since the nineteen sixties with an approximate doubling time of eighteen month, a fact that is commonly addressed as Moore’s law [13]. It is not so well known, however, that the spectacular exponential growth in computer power has been overshadowed by the progress in numerical mathematics that led to an enormous increase in the efficiency of algorithms. Martin Grötschel from the Konrad Zuse-Zentrum in Berlin [14, p.71] reports one spectacular case: Using the computers and the linear programming algorithms of 1988, the solution of a benchmark production planning model by linear programming would have taken 82 years central processing unit (CPU) time. In 2003 – fifteen years later – the same model could be solved in one minute and this means an improvement by a factor of about 43 million. Out of this, a factor of roughly 1 000 resulted from the increase in processor speed whereas a factor of 43 000 was due to improvement in the algorithms. Many other examples of similar progress in the design of algorithms can be given and remarkably, progress continues at the same speed.

Biology is in a special situation since neither traditional biology nor early molecular biology required support by computer science. Classical biology had hardly a single root in mathematics – Darwin’s centennial ‘Origin of Species’ [15] does not contain a single equation – and the only exception of a formal theoretical approach that has been acknowledged by twentieth century biologists is population genetics founded by the mathematician Ronald Fisher and his contemporary colleagues J.B.S. Haldane and Sewall Wright. In the second half of the twentieth century novel experimental techniques changed the situation completely: New methods for DNA sequencing made possible the determination of whole genomes in relatively short time. Was the human inspection of the short genomes of viroids and viruses sometimes tedious but still possible so had even the smartest unaided brain to give up in view of millions and billions of nucleotides in a sequence. Fortunately, algorithms for sequence comparisons were already available when they were needed [16,17]. The presumably oldest method for sequence comparison was conceived by Saul Needleman and Christian Wunsch [16] and it was thought to be applied to comparisons of protein sequences for phylogenetic reconstructions of evolutionary trees because this was the problem of the day as DNA or RNA sequence determinations were very rare in nineteen seventy. The Smith-Waterman algorithm is a variant of Needleman-Wunsch in the sense that it aims at fitting a small sequence optimally to a much longer sequence – a problem commonly addressed as *local alignment* whereas Needleman-Wunsch aims at *global alignment*. Both algorithms are based on linear programming and this means that the result is optimal within the frame of the approach. Although these algorithms were perfect for the sequence comparison problem at the time of their implementations as computer programs, fast advancing efficiency of DNA sequencing demanded more elaborate and in particular faster tools. We mention two frequently used packages used in sequence comparisons: FASTA (fast-all) [18] being the DNA adapted version of an earlier tool for protein sequence comparison [19] and BLAST (basic local alignment research tool) [20]. FASTA provided a standard format for sequence input that is still in use now and BLAST became the most widely used software package for sequence alignment and comparison. What makes BLAST so attractive is the high speed at which it operates. An old American saying is: “There is no free lunch”, and indeed the high speed of alignment and comparison is achieved because quality of performance is sacrificed. The original algorithms based on linear programming – Needleman-Wunsch and Smith-Waterman – are targeted towards finding optimal alignments whereas BLAST is a highly efficient heuristic, which eventually finds only a suboptimal solution in case the optimal one is too hard to find. Nevertheless, without algorithms like BLAST it would be impossible to handle the genomes of higher organisms. The simultaneous alignment of several sequences became an important issue in phylogeny and again software corresponding to the new demand has been developed in form of the CLUSTAL (cluster analysis) algorithm [21]. All these programs were further developed in the following years and several other more efficient algorithms for sequence alignments, comparisons, and other analytic tasks were added (For more information on sequence comparisons see, for example, the monograph ref. 22). Present day genomics and phylogenetic analysis were unthinkable without the toolboxes developed in bioinformatics.

Software development for the prediction and analysis of the structures of biomolecules is often characterized as structural bioinformatics. The progress in our understanding of protein and nucleic acid structures was spectacular and interpretation of function without referring to structure has become unthinkable in present day biology. The currently most promising approaches combine algorithms, which are based on theory, with empirical data. We dispense here from all details which can be retrieved, for example, from a webpage [23] but mention the currently highly important task combining sequence analysis and structure prediction: the detection of conserved structural elements through sequence analysis. In case of simplified RNA structures – the so-called secondary structures – simultaneous sequence analysis and structure prediction can be combined allowing for the determination of consensus structures of two or more aligned sequences. The first solution of the problem has been provided by David Sankoff [24] but his algorithm was too slow for most applications. Later developments led to much faster algorithms, which are heading for slightly suboptimal practical solutions, for example RNAalifold [25,26] (for a toolbox of RNA secondary structure related programs see ref. 27). The detection of a conserved structure is the best indication for the biological function of a stretch of DNA or RNA, which became a hot topic after the discovery that most of the human genomic DNA is not mere junk but transcribed into RNA whose function remains to be revealed [28,29,30]. Deciphering the "*functional code*" of genomes is presumably the most important topic of cutting-edge research in present day biology.

Many other disciplines were also revolutionized by application of novel technologies – we can mention here only two examples: (i) high-throughput experiments and (ii) large scale modeling. Biological, medical, and pharmaceutical research was put on a new basis through the development of high-throughput methods. Simultaneous experiments on up to thousands of samples evidently require efficient software tools for planning, execution, and analysis. Systems biology aims at modeling entire cells with thousands of molecular players and in order to be successful, reaction parameters in similarly large numbers are required that can be obtained only from biochemical research work (see, for example, ref. 31 and 32). Modeling entire cells and organisms on the molecular level – the ultimate goal of systems biology – is among the greatest challenges of the near future.

Highly renowned molecular biologists, Sydney Brenner for example, see currently a room for a new theoretical biology [33]. In order to fulfill its task such a discipline has to find new approaches to handle the huge piles of data and to extract the relevant scientific knowledge sleeping among less informative stuff. Certainly, this will be impossible without a firm basis in mathematics and computer science. Bioinformatics provides the essential tools for modern biology in particular for storing, handling, retrieval, and analysis of the hitherto unseen amounts of results. As stated initially, software has to be highly reliable, since results can only be checked for consistency and details cannot be verified by human eye or brain as it was obligatory done in conventional science up to now. Still open at least for me is the question, what consistency means in unknown scientific territory. Physics can rely on the solid construction of theory as long as no paradigm shift is on the horizon – as it happened at the turn of the nineteenth to the twentieth century in the form of relativity theory and quantum mechanics. Present day biology is in weaker and from another point of view in a more favorable position, because it is lacking a strong and well established theoretical frame: It is less clear what is consistent with the dominant current view and what is contradictory but the resistance against the necessary shift in biological thought is much weaker, and we are currently in the middle of such a shift: The conventional Mendelian view of genetics that has been put on a firm basis by population geneticists: Now it has to attribute room to alternative views that were discovered by molecular biologists: Regulation of gene activities in higher organisms, in particular imprinting and regulation by RNA, allow for phenomena that were commonly interpreted as epigenetic mechanisms. These mechanisms allow for the inheritance of acquired features in the sense of what is called Lamarckian evolution in the popular literature.[‡] The occurrence of several epigenetic mechanisms of inheritance is well documented and understood in principle, and therefore the question is not whether Lamarckian evolution occurs, it is how much inheritance follows the Darwinian pathway of mutation and selection and how much is direct transmittance from parents to progeny. Genetics after all is working: children resemble their parents and grandparents, and breeders in nursery gardens and animal farms produce excellent results by applying conventional genetic rules.

Finally we come back to the theme indicated in the title: Sutlers provided a great variety of goods for soldiers and entire armies during wartime, they were and are indispensible for military logistics not to the least because they can react instantaneously to the needs of the day, and are not dependent on the bureaucracy of an army. Sutlers are not estimated highly despite their importance for the function of military forces. The analogy to computer scientist is straightforward, in particular in biology: Bioinformatics is indispensible for progress but the credit for the achievements is given to other people. Craig Venter for good reasons became famous for his works in genomics but nobody knows the names of the computer scientists who made possible the evaluation and the synthesis of fragments to an entire genome, and other examples could be added. The only exceptions in chemistry – I am aware of – are in the exploration of molecular structures where the Nobel Prices 1998 and 2013 went to people doing computer work: Walter Kohn and John Pople, and Martin Karplus, Michael Lewitt and Arieh Warshel, respectively. In times where every experimentalist could interpret his results this was well justified but now when more and more responsibility goes to computer software it might be worth to rethink the relative importance of experimental and computational work.

References:

[1] Bledsoe, W. W., K., Loveland, D. W., Eds. Automated theorem proofing: After 25 years. Contemporary Mathematics, Vol.29, AMS: Providence, RI, 1984.
[2] Appel, K., Haken, W. Every planar map is four colorable. Part I. Discharging. Illinois Journal of Mathematics 1977, 21, 429-490 and Part II. Reducibility. Illinois Journal of Mathematics 1977, 21, 491-567.
[3] Gonthier, G. Formal proof – The four color problem. Notices of the AMS 2008, 55, 1382-1393.
[4] Szpiro, G. Does the proof stack up? Nature 2003, 424, 12- 13.
[5] Szpiro, G. Kepler's conjecture: How some of the greatest minds in history helped solve one of the oldest math problems in the world. John Wiley & Sons: Hoboken, NJ, 2003

[‡] The distinction between germline inheritance and transmission of acquired properties is commonly addressed as Darwinian and Lamarckian evolution, although Darwin's views on the mechanism of inheritance had definitely Lamarckian features. In the orthodox view of genetics there is no room for Lamarckian mechanisms, and this is apparently wrong in the light of modern molecular biology.

[6] Hales, T. C. A proof of the Kepler conjecture. Annals of Mathematics 2005, 162, 1065-1185.
[7] Fejes Toth, G., Lagarias, J. C. Guest editors' foreword. Discrete & Computational Geometry 2006, 36, 1-3.
[8] Hales, T. C. Formal proof. Notices of the AMS 2008, 55, 1370-1380.
[9] MacKenzie, D. Mechanizing proof. MIT Press: Cambridge, MA, 2001.
[10] Kaner, C., Bach, J., Pettichord, B. Lassons learned in software testing. John Wiley & Sons: Hoboken, NJ 2001
[11] Weimer,W., Forrest, S., Le Goues, C., Nguyen, T. V. Automatic program repair with evolutionary computation. Discrete & Computational Geometry 2010, 36, 1-3.
[12] Le Goues, C., Nguyen, T. V., Forrest, S., Weimer,W. A generic method for automated software repair. Communications of the ACM Research Highlights 2012, 53, 109-116.
[13] Moore, G. E. Cramming more components onto integrated circuits. Electronics 1965, 38(8), 4-7.
[14] Holdren, J. P., Lander, E., Varmus, H. Designing a digital future: Federally funded research and development in networking and information technology. President's Council of Advisors on Science and Technology, Washington, DC, 2010.
[15] Darwin, C. On the Origin of species by means of natural selection or the preservationof favoured races in the struggle for life. John Murray, London, 1859.
[16] Needleman, S. B., Wunsch, E. A general method applicable to the search for similarities in the amnino acid sequence of two proteins. J. Molecular Biology 1970, 48, 443-453.
[17] Smith, T. F., Waterman, M. S. Identification of common molecular subsequences. J. Molecular Biology 1981, 147, 195-197.
[18] Pearson, W. R., Lipman, D. J. Improved tools for biological sequence comparison. Proc. Natl. Acad. Sci. USA 1988, 85, 2444-2448.
[19] Lipman, D. J., Pearson, W. R. Rapid and sensitive protein similarity searches. Science 1985, 227, 1435-1441.
[20] Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D. Basic local alignment research tool. J. Molecular Biology 1990, 215, 403-410.
[21] Chenna, R., Sugawara, H., Koike, T., Gibson, T. J., Higgins, D. G., Thompson, J. D. Multiple sequence alignment with the CLUSTAL series of programs. Nucleic Acids Research 2003, 31, 3497-3500.
[22] Mount, D. M. Bioinformatics: Sequence and Genome Analysis, Second Ed. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 2004.
[23] The Virtual Library of Biochemistry, Molecular Biology and Cell Biology. URL: **www.biochemweb.org/structural.shtml**, last retrieval Dec.27, 2013.
[24] Sankoff, D. Simultaneous solution of the RNA folding, alignment and protosequence problems. SIAM J. Appl. Math. 1985, 45, 810-825.
[25] Hofacker, I.L., Fekete, M., Stadler, P.F. Secondary structure predicition for aligned RNA sequences. J. Molecular Biology 2002, 319, 1059-1066.
[26] Bernhart S. H., Hofacker, I.L., Will, S., Gruber, A.R., Stadler, P.F. RNAalifold: Improved consensus structure prediction for RNA alignments. BMC Bioinformatics 2008, 9, e474.
[27] Vienna RNA package. URL: **www.tbi.univie.ac.at/~ivo/RNA/**, last retrieval Dec.27, 2013.
[28] The ENCODE project consortium. The ENCODE (**ENC**yclopedia **O**f **D**NA **E**lements) project. Science 2004, 306, 636-640.
[29] The ENCODE project consortium. Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. Nature 2007, 447, 799-816.
[30] The ENCODE project consortium. ENCODE explained. Nature 2012, 489, 52-55.
[31] Palsson, B. O. Systems biology: Properties of reconstructed networks. Cambridge Unversity Press, Cambridge, UK, 2006.

[31] Wilkinson, D. J. Stochastic modeling for systems biology. Second Ed. Chapman & Hall/CRC Press, Boca Raton, FL, 2012.

[33] Brenner, S. Theoretical biology in the third millenium. Phil.Trans.Roy.Soc.London B 1999, 354, 1963-1965