MIT Reversible Computing Project
Memo #M5

Physically-Motivated Models of Computation for Complexity Theory

WORKING DRAFT
Revision: 1.1

Michael P. Frank
MIT AI Lab, Rm. 747
545 Technology Sq.
Cambridge, MA 02139
`http://www.ai.mit.edu/~mpf`

Started February 17, 1997.
Revision Date: 1997/02/18 23:32:31 GMT
Formatted March 4, 1997.

A current version is available online at
`http://www.ai.mit.edu/~mpf/rc/memos/M05_physical.html`

Abstract

The theory of computational complexity studies how the time and space required for a computer to solve problems in a given class depends on the size of the problem. Unfortunately, different models of computation, such as Turing machines, RAM machines, and parallel models, are found to differ by polynomial factors in the speeds at which they can solve particular classes of problems. Algorithms that seem fastest in one model may not turn out to be fastest in another. As computing technology improves, inaccuracies of the earlier models become apparent, and the run-time analysis of problems and algorithms must be re-evaluated.

In this paper, we attempt to leap ahead, and get away from the model-relativism of traditional complexity theory, by proposing a new theory based on "ultimate" models of computation, in which we forecast a "best possible" computing technology that takes full advantage of the known laws of physics, while modeling the limitations imposed by these laws, including the three-dimensionality of space, the finiteness of the speed of light, and the laws of thermodynamics.

We conjecture a "Tight Church's Thesis" that claims that any model of computation that predicts a smaller order of growth for the time to solve any class of problem than our model does is not actually physically realistic, in the sense that its growth predictions will break down in any real implementation as the problem size is made larger. It is conjectured that the only fundamental physical effect limiting the scalability of our model will be gravity; i.e., our model's predictions will break down only when the mass of the computer is so great that it inevitably collapses into a black hole. This will be considered optimal modeling, in the sense that no realistic model of computation could ever hope to model computers beyond that point anyway.

1 Introduction

1.1 Existing Models of Computation.

[I need to add some references in this section, and throughout this document really.]

Computer science has historically used many different abstract models of what constitutes a "computer." Probably the earliest computing model explored by matematicians was the concept of a "recursive function." Around the same time, Turing proposed his tape machine model. Shortly thereafter Von Neumann explored the power of cellular automata. Over the decades, many other models of computation have popped up, from register machines to pointer machines to RAM machines to parallel RAMs, hypercube and butterfly networks, meshes, etc. The models mentioned so far seem at first glance to be resonable models of real computers. Typically they assume some unlimited size part of the computer (such as the infinite blank tape in the Turing machine), but generally the unlimited resource is of the sort that it does not seem unreasonable that we could just provide as much of it as is needed for a computation.

All of the above-mentioned models have the property that they have been shown to be equivalent in the sense of obeying the "Strong Church's Thesis," which claims that all reasonable models of computation are equivalent in power to the Turing machine within a polynomial factor. That is, the running time to solve a problem on any of these models is at worst a polynomial function of the running time needed to solve the problem on any of the others.

Additionally, models have been proposed that seem patently unreasonable, such as nondeterministic Turing machines, alternating machines, machines defined relative to "oracles", infinite-precision analog systems that can perform infinite amounts of computation in finite time, and so forth. Some of these models, such as nondeterministic Turing machines, have not been proven to be impossible to realize in polynomial time on a normal Turing machine, but most researchers would still consider them unreasonable models, and would be very surprised if they did turn out to be equivalent to Turing machines within a polynomial factor. We will not deal with such models henceforth in this paper.

The strong Church's thesis is useful because it permits researchers studying the computational complexity of problems and algorithms to derive their results in whichever model they prefer, with the assurance that any real computer will be able to utilize their algorithm with at worst a polynomial factor slowdown.

Polynomial factors are great for studying the theoretical relationships between complexity classes, but in the real world, polynomial factors do matter, and when picking an algorithm to solve a problem, it is important to consider how much time the algorithm will take on a real computer, not just on some abstract model that may be polynomially slower—or faster—than the real computer on the given problem. If we do resort to using an abstract model, it should at least be one that accurately models the asymptotic difficulty of solving problems in the computer architecture we are actually using.

What's more, as our skill in computer technology increases, architectures change, and the appropriate abstract models change as well.

For very early computers with tape drives and extremely little memory, the Turing machine may once have been an appropriate model. But very quickly, the memory of computers increased to the point where the finite state machine of a TM tape head was no longer a reasonable model of how the state of the CPU's memory was updated—a lookup table for all possible state transitions would be far too large. Real computers calculated their next state based only on the contents of a few randomly-accessed memory locations pointed to by a small set of registers. This architecture inspired the register machine, pointer machine, and RAM machine models of computation. The "finite" memory was now large enough to represent most inputs, and so now memory (or disk) was considered the new kind of practically-unlimited storage, and the sequential-access tape was dropped from the model.

The next major architectural development was the creation of parallel machines. The Turing and RAM machines suffered from the drawback that they could only operate on small amounts of data at a time, one piece after the other, rather than being able to simultaneously perform many identical (or perhaps different) operations on the data stored at many different places in storage. New architectures had many CPUs that operated in parallel. Thus, to analyze appropriate algorithms for these machines, parallel models of computation were born. These appeared in great variety: PRAMs, hypercube networks, butterfly networkes, meshes, each appropriate for some particular sort of parallel architecture.

Next I will argue that a number of properties of many of the models representing today's computer architectures are actually physically unrealistic and misleading, in the sense that a computer built according to the given architecture or abstract model would in actuality be physically unable to be scaled up arbitrarily to achieve the algorithmic performance on problems that would have been predicted from the abstract model. At various points, these models will all break down, because they fail to take into account one or several of the fundamental realities of physics.

The goal of this paper is to develop a model of computation that does not ignore any fundamental physical principles, and yet utilizes the full computational power afforded by physics. The advantage of having such a model is that when we analyze an algorithm for it, we will be assured that the analysis will still hold no matter how much we scale up the size of the problem or the computer. Also, since the model takes full advantage of physics, we will know that whenever we prove that a given problem requires a certain minimum amount of time to solve, no future advances in computer technology—barring any newly discovered fundamental laws of physics—can possibly nullify the validity of that result.

2 Problems with Existing Models

This section lists ways in which existing computational models fail to recognize fundamental scaling limits imposed by physics.

Unlimited RAM. Models of computation such as the RAM machine typically assume that there is some unlimited amount of storage space, and moreover that any individual piece is accessible within constant time. This is physically unrealistic, because presumably, in any particular technology, there is a limit on the number of bits that can be encoded per unit of space, and the speed of light sets a lower bound on the access time to bits located far away.

More than cubic connectivity. Some types of graph-based models of computation prevent you from accessing arbitrarily many storage locations in constant time, by only allowing you to step from one node to the next in a graph.

But if the graph is a binary tree, say, then exponentially many nodes may still be reached in a given time. This is also unrealistic, because only $O(n^3)$ storage locations may be physically located within a distance n in three-dimensional space, so that even if the information travels at the speed of light, only that number of different locations can be accessed within a given time.

Similarly, many parallel computing architectures such as hypercubes have more than cubic connectivity between their processing nodes, so that as one scales up the number of processors, the wires between them take up more and more space and get longer and longer, until most of the computer consists of wires between processors. This is clearly suboptimal use of space. Mesh and Cellular Automata architectures do not suffer from this difficulty.

Free irreversible operations. Next, existing models allow processing elements to perform irreversible operations (i.e., transitions into states that have more than one predecessor) without considering the impact of the necessary resulting energy dissipation that is implied by thermodynamics. Microscopic dynamics is reversible; therefore the information that is thrown away when performing an irreversible operation must be some exported from the system, and this exporting of information will necessarily limit the scale of computations, since as the volume of the computer increases by n the number of processors increases by $O(n^3)$ whereas the surface area only increases by $O(n^2)$, so that the rate of production of unwanted information will eventually overwhelm our ability to remove it. A realistic computing model should be reversible, so that the means of disposal of unwanted information must be explicitly accounted for the algorithm.

Free reversible operations. Even reversible models of computation also assume that reversible operations require zero energy expenditure. This is true, but only in the limit of taking infinite time to perform the operation. Generally, it seems that any particular computing technology will require dissipating energy per operation that scales proportionally to speed, to make up for frictional/resistive losses during the operation. However it is not clear that there is any fundamental limit to how low the scaling factor may be. (An alternative allowing zero energy per operation is to run quadratically slower, by depending on a random-walk through the computer's state space to perform the computation.)

The fact that even reversible operations require some energy to progress forwards with finite time per step means that even a reversible computer cannot be scaled arbitrarily without slowing down, because eventually there will not be enough surface area to feed in enough energy (and pump out enough heat) to keep the whole volume running. Halving the speed will only allow us to double the number of processors, so there is no net performance increase beyond some critical size.

Note however, that with reversible operations, the energy per operation even at high frequencies can be much lower than kT, the minimum energy for irreversible operations, so correspondingly the machine can be larger than if it were irreversible, by some constant factor that increases as technological improvements decrease the resistive losses.

Error-free operation. Also, most traditional models of computation assume that computational operation is error-free, or that the error rate can be made as small as desired with no impact on the speed of a system. A better model would be to have a fixed probability of error per primitive operation that depends on the technology being used.

Non-quantum Operation. Next, traditional computational models take a classical viewpoint, that the computer is in a definite state at any given time. However, recent research in quantum computing seems to indicate that computers that use quantum superpositions of states and take advantage of interference effects between them may be asymptotically faster than classical computers on some problems. Thus, non-quantum computational models may fail to take advantage of the full computational power offered by physics.

However, most people find quantum computers more difficult to reason about than classical computers, and furthermore, the quantum computers have not been conclusively proven to be more powerful than classical computers. Therefore in this paper we will consider non-quantum computational models in addition to quantum ones. Either the two models are equivalent, in which case the classical model should be used because of its simplicity, or else the quantum model is superior and should be the "ultimate" model of computation.

Digitality. Most models of computation also do not take advantage of the seemingly continuous nature of physics. We will not either, other than allowing continuous amplitudes in our quantum computers, because it is not clear either that physics really is continuous, or that its continuity, if real, confers any extra power. (Some research has shown that systems of point bodies obeying Newtonian mechanics can perform infinite amounts of computation in finite time if their initial positions and velocities are set with infinite precision. However, our universe is not Newtonian, and anyway infinite precision in initial configurations is not a reasonable assumption.)

Unlimited computer size. Finally, some parallel computing models presume that a parallel machine may have an arbitrarily large number of processing elements. However, this is not realistic, because even if unlimited matter and energy and construction time were available to construct the computer, if you were to group together a sufficiently large number of processing elements of any given technology, they would inevitably collapse into a black hole due to their own gravity. Preventing this collapse would require spreading the processing elements farther apart, which increases their communication latency. Thus gravity changes the scaling laws in the realm of extremely large parallel machines.

However, one has to stop modeling somewhere, and the gravitational limit is so unreasonably far away that we will not attempt to deal with it in this paper. That is, our own models will also be invalid in the realm where the computer's self-gravity becomes a significant factor.

3 Proposed Models

In this section we present our proposed "ultimate" models of computation. We will present both a classical and a quantum model, because the quantum model has not been proven to be more powerful than the classical one, and if it turns out not to be more powerful, then the classical model is preferable, because it is simpler and easier to reason about. However, I believe that most likely the quantum model is indeed strictly more powerful, in which case it represents the true ultimate model of computation.

3.1 Classical Physical Model of Computation

We call our non-quantum model "the Classical Physical Model of Computation" or CPMC. It is as follows.

There is an infinite (or unbounded-size) three-dimensional cubic mesh of computing elements, O(1) distance apart from each other. Each computing element has O(1) memory and has inputs and outputs only from and to its immediate six neighbors. Each O(1)-length time step, O(1) bits may pass along each I/O connection, and each computing element sets its new state and its outputs based on its old state and its inputs. The transition function must be the same for all computing elements, and it must be reversible.

Furthermore, there is some O(1) energy cost per O(1)-time operation, and O(1) max energy flow capacity through each cell per unit time. The net effect of all this would be to limit the size of the mesh to O(1), in which case we might as well use a serial model. However! We assume that because the system is reversible and dissipates very little energy per operation, the maximum mesh size is large compared to the size of problems we will want to solve, so that in our model we will just ignore this fixed upper limit. Besides, as far as we know, physics does not forbid implementations where the limit on mesh size imposed by energy requirements exceeds the mesh size that would cause gravitational collapse. In other words, the black hole limit may be the real limiting factor to size anyway. Therefore we ignore the size limit due to frictional losses, since anyway it will increase as the technology improves. (The is in contrast to the kT energy limit if the computer were irreversible, which cannot improve no matter what the technology was.)

Finally, we assume an O(1) probability of error per processing element per step.

The settings of all the O(1) parameters in the model depend on the particular technology being used to construct the processing elements.

When analyzing the complexity of algorithms in this model, one assumes that an input consists of some finite region of the grid that is pre-configured into some initial state representing the input, where the remainder of the conceptually infinite grid is filled with some standard and uniform initial pattern. We let the mesh compute, possibly utilizing additional computing elements outside the input region if necessary, and at some point, some definite region of the grid contains the answer, in some prespecified format.

Note that the model described above is really just a type of 3-D reversible cellular automaton. Such models have been described and explored before [cite Norm, others], but there has not so far been much emphasis on using such a model as the basis for a theory of computational complexity.

3.2 The Quantum Physical Model of Computation

Our "ultimate" quantum model of computation will be called "the Quantum Physical Model of Computation" or QPMC. It will be almost identical to the classical model, except that the transition function applied to each computing element at each step will be an arbitrary unitary transformation over the state of the O(1) memory and input bits, considered as qubits, and furthermore, after each such update, the input and output bits of each computing element will exchange roles. (So that the modified inputs will become output bits and be available for the neighboring cells to manipulate on the next step.)

Additionally, there will be an O(1) probability of an unwanted interaction with the environment per qubit per time step. These interactions are modeled as being uncorrelated so that quantum error-correction algorithms may apply.

The justification for considering this model to be an "ultimate" model of computation is that it is conjectured to be able to simulate physics itself (ignoring gravity) with only a constant factor slowdown, at least if the probability of unwanted interactions is sufficiently small. Therefore any physical computer (unless it relies on large gravitational fields or takes advantage of some unknown laws of physics) ought to be unable to solve any problem with a smaller order of growth of run-time.

4 The Tight Church's Thesis.

We now present a couple of strong conjectures about the physicality and optimality of our models of computation. We present separate conjectures for the classical and quantum models, because we wish to make a meaningful statement about the power of the classical model even if it turns out that the quantum model is indeed strictly more powerful.

Conjecture. (*Classical Tight Church's Thesis.*) It is conjectured that the classical physical model of computation (CPMC) has the following two properties:

Implementability. For a suitable choice of constant bounds on all the $O(1)$ parameters in the model, it is possible to actually implement the classical physical model of computation (CPMC) in such a way that, given access to sufficient quantities of matter and organized energy (work, not heat), and sufficient construction time, a civilization could build computers that actually realize all the runtime order of growth predictions of the model, at least until input sizes reach the point at which the required computer would have a mass so great that it would inevitably collapse into a black hole.

Optimality. For any model of computation not based on quantum interference effects, if the model meets the above implementability criterion, then it predicts an order of growth for all problems that is asymptotically as large as or larger than the order of growth predicted by the CPMC.

Conjecture. (*Quantum Tight Church's Thesis.*) It is conjectured that the quantum physical model of computation (QPMC) has the following two properties:

Implementability. QPMC is implementable, in the sense above.

Optimality. For any model of computation, if the model is implementable in the sense above, then it predicts an order of growth for all problems that is asymptotically as large as or larger than the order of growth predicted by the QPMC.

The upshot of the above two conjectures is to claim that both CPMC and QPMC are physically realizable, and that CPMC is as powerful as any non-quantum model of computation (within a constant factor), and QPMC is as powerful as any model of computation whatsoever (within a constant factor).

5 Ultimate Computational Complexity

In any case, given the ultimate classical and quantum models of computation proposed above, we can now proceed to derive classical and quantum bounds on the ultimate computational complexity of various classes of problems.

Some of this work can proceed by merely embedding algorithms for earlier models into our framework. Any multi-tape turing machine (with a constant number of tapes) and any reversible cellular automaton in up to three dimensions, and any irreversible cellular automaton in up to 2 dimensions, can be simulated with no increase in order run time by our CPMC model, so bounds on order growth of problems derived for these machines will immediately apply to our model.

However, translating algorithms for many other models into our framework is more difficult. This is because of the unphysical assumptions made by those other models. Simulating these nonphysical models in our models requires an increase in run time; which is exactly why we wish to get away from those other models, because results derived for them are misleading because they are not physically realizable.

Note this is also true of quantum computing models that assume that quantum gate operations can be applied to any selection of qubits out of an arbitrarily large number in the computer, in constant time. Instead, arbitrarily-chosen qubits will generally be distributed in space, and must be moved to be close together before we can operate on them. So simulating those earlier quantum computing models in our QPMC model will introduce some slowdowns.

Note, however, that for all the "reasonable" models of computation that have previously been proposed, the slowdowns incurred by simulating them in CPMC or QPMC are at worst polynomial. So although some of the earlier reasonable models may exaggerate the speed of some algorithms, the exaggerations are not too bad. Still we would like to make sure that orders of growth of problems are expressed with the *right* polynomial.

To find good algorithms for solving problems in our CPMC and QPMC models, it will often be a good strategy to start with an algorithm for some traditional model, simulate it straightforwardly in our model (possibly with increased run time) and then figure out how to perhaps modify the resulting algorithm to improve its performance in our model. However, it may turn out that the best algorithm for a given problem in CPMC or QPMC may operate in a totally different fashion from the best algorithm for that problem in a traditional model. In some cases, the best algorithm in our model may be slower than the best algorithm in a traditional model. But it should be remembered that the best algorithm in our model *is really the best algorithm* in an absolute sense, because the other models are not physically realistic, and the runtime growth functions derived for them are generally *not physically achievable* beyond a certain scale. Our model, on the other hand, is not only physically realistic, but it takes full advantage of what physics offers. (This is our claim.)

6 Conclusion

Most existing models of computation are quantitatively either too weak, in the sense that asymptotically faster computers are physically possible, or too strong, in the sense that any physical implementation would perform asymptotically more slowly on large problems than the model predicts.

In this paper, we proposed two models of computation that attempt to be *just right* in the sense that they take full advantage of the computational power that physics offers, but do not exceed it. We propose a "Tight Church's Thesis" that conjectures that our models achieve this goal.

If the Tight Church's Thesis is correct, then bounds on the computational complexity of problem classes that are derived in our models represent bounds on the true difficulty of solving those problems in our universe. Lower bounds we derive will always still apply, no matter how computer technology is improved. And upper bounds we derive will always still apply no matter how large our inputs become, until either we reach the limits imposed by frictional losses in a particular computing technology (for good technologies this limit may be very high), or we reach the point where the required computer cannot be manufactured because it would be so large that it would collapse under its own gravity.

Our models include both quantum and classical models. The quantum model is thought to be more powerful, in which case it, rather than the classical model, is the actual *just right* model of computation that we are looking for. However, the quantum model is difficult to implement and to reason about, and it has not yet been conclusively proven to be more powerful, so the classical model may still be preferred. In any case, we conjecture that one of these two models correctly represents the computational capabilities of our universe (within a constant factor), and we suggest that both models are appropriate targets for further complexity-theoretic study.