Chapter 4

Ultimate physical models of computation

In this chapter we elaborate in some depth our motivating long-term goal (first mentioned in §1.2) of describing "ultimate" physically-based models of computation. We contrast this concept with the models that have traditionally been studied in computer science. We outline several plausible candidates for what the ultimate model ought to look like, based on the fundamental physical observations from chapters 2.

Traditional complexity theory. As we saw in chapter 3, the subfield of computer science known as "the theory of computational complexity" traditionally deals with 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 various parallel models, are found to differ by polynomial factors in the speeds at which they can solve particular classes of problems. The algorithms that seem fastest in one model may not turn out to be fastest in another. One reason for this is that issues of the physical movement and routing of information are often considered part of the architecture, rather than part of the algorithm. Thus, as computing technology continues to develop, we occasionally find ourselves in a situation where the old models have become inapplicable, and previous work on finding the run-times of problems or on developing efficient algorithms must often be redone.

Ultimate models of computation. In our work, 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 computational power of the known laws of physics, while also recognizing the limitations imposed by these laws, including the three-dimensionality of space, the finiteness of the speed of light, and the second law of thermodynamics. In this sort of model, the way information and entropy are routed become an explicit part of the algorithm.

A tight Church's thesis. For our ultimate models, we conjecture what we call a "Tight Church's Thesis," which claims that any model of computation that predicts a smaller order of growth for the time to solve any class of problem than is predicted for our model 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 accuracy of our model will be gravity; *i.e.*, our model's predictions will break down only when the size of the computer described by our model becomes so large that even if it were built in space, it would collapse under its own self-gravity. Thus, even if not truly "ultimate," our class of model is, at least, expected to remain accurate for all computers that are built for a very long time. As we already stated in §2.3, in this thesis we will not attempt to explore the high-gravity regime.

4.1 What is a model of computation?

A model of computation is simply an abstract formal framework within which we can describe a computation process.

Role of infinities. Many models invoke infinities, such as the infinitely-long tape in a Turing machine, or an infinite cellular automaton array. However, in the more realistic models, any such infinities will be mere mathematical conveniences, and do not cause the model to be infinitely more computationally powerful than anything we might achieve in the actual universe, which may be finite.

Finiteness of the universe. Speaking more precisely, the best guess of modern cosmology is that the total volume and quantity of mass-energy in the part of the universe that is causally connected to us is finite. Therefore, with reference to the fundamental limits discussed in section 2.2, the maximum entropy or informational complexity of the universe (or in computer science terms, its maximum storage capacity) is finite as well. However, the universe is expanding, and according to some of the latest studies may very well continue to expand forever, in which case its entropy may indeed, over time, increase without bound. But at any particular time, everything is finite: propagation speeds, processing rates, information capacities, *etc.*, as we reviewed throughout chapter 2. In realistic models of computation, such quantities should be finite as well.

Turing machines are realistic. The Turing machine is a good example of a realistic model, because the only infinity in the model, namely the infinite tape, is a mathematical convenience that is inessential to the model. It can be replaced by a tape that is finite at any moment, but that can grow unboundedly large, as needed, given unbounded amounts of time. Similarly, the finite controller for the tape-head is allowed to be unboundedly large, but is fixed for any particular machine.

The above property of Turing machines, which we might call *unbounded space*, might even conceivably be realistic, in the very long term, if the universe is ever-growing and its maximum entropy increases without bound. In the near term, there will certainly be many much tighter real-world constraints on storage capacities. But whether unbounded space is realistic or not, it is a useful property because it frees our model from having to worry about any specific space bound, which would be highly situation-dependent anyway, and focus instead on other issues.

Note also that unbounded space does not imply infinite complexity in the initial configuration; indeed such complexity would make the model nonphysical.

Families of fixed-space machines. As we pointed out, Turing machine models usually provide unbounded space through either an infinite blank tape, or a tape that may be indefinitely extended as needed during the course of a computation. Another way to permit unbounded space within finite models is to specify a particular fixed, finite amount of space as part of the initial condition for any instance of the model, but define "the model" to encompass an infinite family of instances having arbitrarily large fixed amounts of space. This conception corresponds nicely to what we do when we build a particular computer; its information capacity is finite, and the machine cannot of its own accord increase its capacity; nevertheless for any given finite computation, a machine can be built that has sufficient capacity to perform it.

In this work we will often describe models of computation in this way, in terms of a family of machines, each instance of which has fixed capacity, but where for any desired capacity, some machine in the family has that capacity. Moreover, the instances must all be Turing-computable; we are not allowed to hide the values of an uncomputable function in the family of instances, or at least, such models would not be considered realistic.

It is important to realize that models of computation that consist of computable families of fixed-capacity machines are equivalent in power to Turing machines with extensible tapes, on any problem class that has computable space bounds, even though all particular machine instances in our model family are finite-state machines. This is because we are allowed to pick larger machine instances as needed, as problem sizes become larger.

If no bound can be placed on the space requirements for a given problem class, we can handle this by extending our model with a protocol whereby a particular fixed-capacity machine is simply rebuilt with a larger capacity if it runs out of space during a particular problem instance.

Of course, if we include the human builders in the concept of the system, or if we create machines that construct more of themselves automatically out of raw materials, expanding along with the universe, then even a particular instance of a computing system could, in reality, have unbounded capacity just like a theoretical Turing machine with extensible tape.

We have broadly outlined what we consider to be a legal candidate for a realistic model of computation, in terms of what quantities may or may not be infinite. Now we survey more closely the range of existing models.

4.2 Existing models of computation

Computer science has historically used many different abstract models of what constitutes a computer. One of the earliest computing models explored by mathematicians was the concept of a *recursive function* (*cf.* the text [116]). Around the same time, Turing proposed his tape machine model. Shortly thereafter, von Neumann explored the power of cellular automata (*cf.* [154]). Over the decades, many other models of computation have popped up, from register machines to pointer machines and RAM machines to parallel PRAMs, hypercubes, butterfly networks, meshes, *etc.* The models mentioned so far seem at first glance to be reasonable models of real computers. Typically the models assume the availability of some unbounded resource (such as the infinite blank tape in the Turing machine, or the infinite grid in a cellular automaton), but as we discussed above, generally the models' complexity predictions do not actually require constructing impossible machines having an infinity of resources, but rather just a reasonable ability to construct more of the resource when needed.

All of the models mentioned above have the property that they have been shown to be equivalent, in the sense of obeying the "Quantitative Church's Thesis" (or Strong Church's Thesis) [148, 147], 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 class on any of these models is at worst a polynomial function of the running time needed to solve the problem class on any of the others.

Additionally, models have been proposed that seem patently unreasonable, such as nondeterministic Turing machines ([108] §2.7, p. 45), alternating machines (which quickly solve problems in the cumulative polynomial hierarchy, [108] §17.2, p. 425), machines that can query infinitely complex or uncomputable "oracles" ([108] §14.3, p. 339), infinite-precision analog models that can perform infinite amounts of computation in finite time [125], and so forth. Some of these models, such as nondeterministic Turing machines, have not actually 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 were to turn out to be equivalent to Turing machines within a polynomial factor. Therefore we will not deal with such models henceforth in this work.

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 favored algorithm with at worst a polynomial factor slowdown.

Such polynomial factor slowdowns are perfectly ignorable when one is just studying the theoretical relationships between broad complexity classes (such as **P** and **EXPTIME**), but in the real world, differences by polynomial factors do matter, and when picking an algorithm to solve a real-world 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 for the given problem. If we do resort to using an abstract model, then one might argue that it should at least be one that accurately models the true asymptotic difficulty of solving problems in a computer architecture we can actually build.

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

For example, for some very early computers having magnetic tape storage and extremely little memory, a simple 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 in 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 state memory previously modeled as "finite" 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, log-depth boolean circuits, hypercube networks, butterfly networks, meshes; each appropriate for some particular sort of parallel architecture.

Next I will argue that a number of properties of many of the models supposedly 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 while still achieving the same performance that would have been predicted from the abstract model. At various points, the models will break down, because they fail to take into account one or several of the fundamental realities of physics.

As we have said before, the ultimate goal of this work 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 (within a constant factor). 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 true no matter how much we scale up the size of the problem or the computer. Also, since the model is designed to take full advantage of known physics, we will know that whenever we prove that a given problem requires a certain *minimum* order of growth in the time to solve it, no future advances in computer technology—barring some newly discovered fundamental laws of physics—can ever nullify the validity of that result.

There is of course always the possibility that our civilization's technological advancement will plateau for economic or other reasons, before we reach the level where all of the physical effects that we address come into play, in which case the models developed here might never actually be appropriate for application to real computers. However, it is our optimistic assumption that this will not be the case.

4.3 Problems with the existing models

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

Unlimited amounts of unit-access-time 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 of it is accessible within constant time. This is physically unrealistic, because presumably, in any particular technology, there will always be a limit on the number of bits that can be encoded per unit of spatial volume, and the speed of light sets a lower bound on the access time to bits located in the more distant volume elements.

Super-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 storage 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 $\mathcal{O}(d^3)$ storage locations may be physically located within a distance d in three-dimensional space, so that even if the information travels at the speed of light, only $\mathcal{O}(t^3)$ different locations can be accessible within a given time t.

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, and delays scale up in ways not accounted for by the model. If the model accounts for the delays it won't be unrealistic, but still, it cannot be presumed to be making best possible use of the space available. Three-dimensional mesh architectures and cellular automata do not suffer from this difficulty.

Free irreversible operations. Next, nearly all of the 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. As we saw in ch. 2, microscopic dynamics is reversible; therefore the information that is thrown away when performing an irreversible operation must be somehow exported from the system.

As we will demonstrate in more detail in ch. 5, this exporting of information will necessarily constrain the scaling of irreversible computations, since as the radius of the computer increases by a factor of n, the number of processors increases by $\mathcal{O}(n^3)$ whereas the surface area only increases by $\mathcal{O}(n^2)$, so that the rate of production of unwanted information will eventually overwhelm our ability to remove it.

A realistic computing model should instead be reversible, so that the means of disposal of unwanted information must be explicitly accounted for in the algorithm.

We should note that some irreversible models such as Turing machines or 2-D cellular automata have sub-cubic connectivity, and are therefore physically realistic because the unused dimension(s) can be used to deliver free energy and dispose of waste heat. However, models that don't use 3 dimensions can't be assumed to be taking full advantage of physics.

Free reversible operations. Even reversible models of computation sometimes assume that reversible operations require zero energy expenditure. This is true, but only in the limit of taking infinite time to perform the operation. As we will see later, 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 friction-related scaling factor may be. (An alternative approach that allows exactly zero energy per operation is to run quadratically more slowly, by depending on a random-walk through the computer's state space to perform the computation. [17])

The fact that even reversible operations require some energy to progress forwards with a fixed lower-bound on the time per step means that even a *reversible* computer cannot be scaled arbitrarily without slowing its clock rate, 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 at the desired speed despite the frictional or resistive losses. Chapter 5 will address this issue in detail.

Note however, that with reversible operations, the energy per operation even at high frequencies might eventually be made much lower than $k_B T \ln 2$, the minimum energy for irreversible bit-erase operations, so correspondingly, the machine can be made much larger than if it were irreversible before slowing its clock rate, by some constant factor that increases as technological improvements decrease the magnitude of frictional losses compared to the fundamental irreducible energy cost of destroying information.

Error-free operation. Additionally, 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, with the magnitude of the probability depending on the particular technology being used. Error-correction techniques in robust architectures can be used to convert a small but fixed probability of error per operation into an arbitrarily small probability of error for the whole computation. However, we will not deal with error models and error correction techniques extensively in our research. We presume that given good engineering, the combination of a low base rate of errors with efficient error-correction techniques would be effective enough that there would not be a large impact on our overall asymptotic scaling results.

Non-quantum operation. Traditional computational models take the classical-physics viewpoint that the computer is in a definite classical state at any given time. However, as we mentioned in §2.6, 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 and algorithms much more difficult to reason about than classical ones, and furthermore, quantum computers have not yet been conclusively proven to be more powerful than classical computers. Therefore in this work we will focus on non-quantum computational models, rather than quantum ones. Either the two models are really equivalent in power, in which case the classical model should be used because of its simplicity, or else the quantum model is more powerful and should instead be considered the "ultimate" model of computation. But, quantum features can be added to our models without changing most of our overall conclusions.

Digitality. Most models of computation also are digital, and do not take advantage of the seemingly continuous range of values that some physical quantities may take. We will not either, other than allowing continuous amplitudes in our quantum computers, because it is not clear either that physics really is continuous (rather than discrete at some very fine scale), or that its continuousness, even if real, confers any additional computational 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 [125]. However, our universe is not Newtonian, and anyway infinite precision in initial configurations is not a reasonable assumption. For some discussions of the power of analog computation models, see [149, 125].

Unlimited computer size. Finally, many parallel computing models presume that a parallel machine may have an arbitrarily large number of processing elements. However, this may not be realistic, since the total mass and maximum entropy of the accessible universe may be bounded. Also, given a fixed processor density, gravitational effects will come into play with a large enough number of processors. There are certainly many other technological and economic limits to computer size that apply at even smaller scales. Nevertheless, it is convenient for modeling purposes to ignore all these facts, since the precise maximum number of processors that can be attained does not qualitatively affect the form that the best algorithms will take at scales that are well below the maximum.

4.4 Some candidates for an ultimate model

We now outline what we believe are some plausible candidates for ultimate physically-based models of computation. Our basic model is a reversible three-dimensional mesh of processing elements. We present three variations of the basic model, which vary in their power. The reason we must do this is that the latter two variations, which are more powerful, depend on the future development of hypothetical technologies that, although perhaps plausible in principle, may not be physically achievable in practice at a useful scale. However, if these technologies do turn out to be feasible, then one of them is actually the ultimate model, and our basic model is not. However, all three models at least share the basic property of reversibility which is the focus of this thesis.

Since we are unable to say with certainty which of these models is the right one, we have not yet attained the ultimate model of computation. However, we feel that with these models we have narrowed the scope of possible models, and have taken a significant step in the right direction.

4.4.1 The reversible 3-D mesh (R3M) model

Our basic, most feasible proposed model is something we will call the reversible 3-D mesh, or R3M. As presently conceived, the R3M model describes a family of fixed-capacity machines. Each machine consists of a uniform three-dimensional array of processing elements, each of which has the capability of fully logically reversible and arbitrarily thermodynamically reversible operation. (Irreversible operations can also be permitted, but these should be entirely optional, and under program control.) Each processor is connected locally to (say) its 6 nearest neighbors. The following parameters of the model are considered to be fixed by a given implementation technology, and are not permitted to vary among the instances in a given family of machines:

- Hardware functionality of each processing element (PE).
- Finite storage capacity of each PE.
- Finite information rate (bandwidth) of each connection between neighboring PEs.
- Finite maximum frequency (minimum cycle time) of each PE.
- Non-zero minimum spacing between neighboring PEs.
- Finite speed at which signals travel between neighboring PEs.

The following parameters are permitted to vary among different instances of a given family, but are fixed for any given instance.

- Number of processing elements in each dimension across the PE grid.
- Actual spacing between neighboring PEs.
- Actual clock frequency at which the PEs are operated.

In addition to the information-processing architecture itself, the R3M model also explicitly accounts for the generation and transport of entropy within the computer, using the following parameters which are fixed for all machines using a given technology:

- Non-zero “entropy coefficient” of each PE. (See ch. 5.)
- Finite maximum entropy flux density within PE grid.
- Finite velocity of entropy transport.
- Entropy generated in case of (optional) information erasure.

Entropy is generated by each PE at a rate that is proportional to the square of its operating frequency, with the constant of proportionality given by the entropy coefficient. This entropy travels at a constant velocity from where it is generated, along some preferred dimension through the grid. (This represents a flow of coolant.) The entropy flux is not permitted to exceed the given maximum. At the edge of the grid, the entropy is considered to be emitted into space, and we don't worry about it further.

Processing elements at the edge of the mesh are permitted to irreversibly convert information to entropy, as frequently as desired, and emit it outwards as well. For processors internal to the mesh, any entropy generation due to information erasure must be explicitly accounted for as contributing to the entropy flow through the machine. Internal processors must refrain from performing more information erasure than can be accommodated given the technology's entropy flux capacity.

If an internal processor wants to discard information to the outside world, it can either send the information to the edge in digital form, and have it dissipated there, or dissipate it internally, and send it out using the coolant flow. The two approaches are asymptotically completely equivalent. The determination of which one is better depends entirely on the constant factors involved in the particular technology.

In chapter 5 we will discuss why the R3M is a realistic model, and show that it scales better physically than *any* irreversible model of computation. Thus it is a a good candidate for the ultimate model, unless there is some way to do better. Let us briefly mention two possible ways we might do better.

4.4.2 The ballistic 3-D mesh (B3M) model

This is just like the R3M except that the entropy coefficient is allowed to be exactly zero. In this model, reversible computation is completely dissipationless and the clock frequency of the PEs can be the same for all instances of a given family of machines, independent of the mesh size. As we will see in 5, a B3M is asymptotically strictly more powerful than any R3M model under various cost measures. Unfortunately it is probably not physically possible to achieve exactly zero dissipation at a fixed speed in any real system. However, it might be possible to approach zero dissipation so closely that dissipation doesn't become a concern at any realistic scale.

4.4.3 The quantum 3-D mesh (Q3M) model

This is just like the R3M, except that the machine can be in a global quantum superposition of states, and the operation of each PE consists of a local unitary transformation on this superposition. A Q3M can simulate other quantum computation models in polynomial time, and thus can factor in polynomial time. It thus may be exponentially faster than the R3M or B3M on some problems.

Feasibility. The use of quantum error correction codes [25, 29, 129] may conceivably permit a Q3M to operate dissipatively while still maintaining a coherent superposition, so a Q3M may be possible even if a B3M is not. However, we are still very far from any practical realization, so in any case the Q3M is a very tentative proposal.

4.5 A "tight" Church's thesis

We are now in a position to state an asymptotically *tight* version of Church's thesis, and conjecture that it holds for one of our proposed models.

Conjecture 4.1. (*Tight Church's Thesis.*) It is conjectured that for at least one of the three models of computation described above (R3M, B3M, Q3M), the following two properties hold:

(a) Implementability. For some suitable choice of constant bounds on all the $\mathcal{O}(1)$ parameters in the model, it is possible to actually implement the models 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 order of growth predictions (for run-time and space) of the model (at least, below the high-gravity regime).

(b) Optimality. For any other model of computation, if the model meets the above implementability criterion, then it implies an order of growth for all problems that is asymptotically at least as large or larger than the order of growth predicted by the model.

The upshot of the above conjecture is to claim that at least one of the types of reversible models that we are considering is physically realizable (at least the R3M is, and maybe the B3M and Q3M are also), and also that at least one of the realizable models is at least as powerful as any other realizable model of computation, on all problems. That is, *any* physically realizable computing device could be simulated in our ultimate computer with *at most* a constant factor slowdown and a constant factor increase in space usage. (Moreover, we conjecture reasonable, non-astronomical constants.)

We cannot rigorously prove our conjecture, in part because the laws of physics are not yet completely understood, but in chapter 5 we at least present a number of rigorous analyses that provide strong support in favor of it. Therefore we go ahead and introduce our conjecture to the research community, partly to see if anyone can find an argument to shoot it down.

4.6 Ultimate computational complexity

In any case, given the apparent correctness of our conjecture, theorists 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 asymptotic space or run time by our R3M 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, often in their interprocessor communication networks: for example, the unit access time to an unbounded shared memory in a PRAM. Simulating these nonphysical models in our models requires an increase in asymptotic run time, but so would *any* physical implementation of those models. This is exactly why we wish to get away from those other models; performance results derived with 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 of qubits 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 they can interact. So, simulating those earlier quantum computing models in our Q3M 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 R3M or Q3M are at worst polynomial. So although some of the earlier reasonable models may exaggerate the achievable speed of some algorithms, the exaggerations are not too bad. Still, we would like to make sure that the orders of growth of problems are expressed with the *right* polynomial.

To find good algorithms for solving problems in our R3M and Q3M 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 if the original model was unrealistic) and then figure out how to perhaps modify the resulting algorithm to optimize its performance in our model. However, it may turn out that the best algorithm for a given problem in R3M or Q3M 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. (At least, this is our conjecture.)

4.7 Summary of discussion of ultimate models

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 some problems than the model predicts.

In this chapter, we proposed three 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 at least one of our three models achieves this goal, at least for all sizes of machines that might be achieved in the foreseeable future.

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.

Our models include both quantum and classical models. The quantum model is thought to be more powerful, in which case it, rather than one of the classical models, 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 one of the classical models may still be preferred. In any case, we conjecture that one of these three models correctly represents the computational capabilities of our universe (within a constant factor), and we suggest that all three models are appropriate targets for further complexity-theoretic study.

In the next chapter, we show why we believe the ultimate model *must at least* be capable of strict *reversible* (if not also ballistic and quantum-coherent) operation, by demonstrating that reversible 3-D meshes are more powerful asymptotically than *any* physical implementation of *any* irreversible model of computation.