On fundamental limits to glacier flow models: computational theory and implications

David BAHR

Department of Physics and Computational Science, Regis University, Denver, Colorado 80221, USA
E-mail: dbahr@regis.edu

ABSTRACT. No single flow model can simulate all possible glaciers and ice sheets without violating fundamental tenets of computational science. The root cause is not one of numerical sophistication, precision or accurate initial conditions. Instead, using flow and transport as data transmission, glaciers inadvertently function as information processors. This computational capability confers a level of complexity that inherently limits our ability to accurately and efficiently predict glacier flow and therefore, for example, to forecast those aspects of climate systems that depend on glaciers. In particular, even with considerable future advancements in glacier physics, computational theory shows that no dramatic improvements in numerical speed are likely when compared to today's glacier models. Therefore, to increase speed and resolution, the next generation of climate and sea-level models must rely on simulations tailored to specific ice-sheet geometries rather than general-purpose glacier flow models. However, because glaciers process information, entirely new computation-theoretic advances in glaciology are possible, and concepts from information entropy may help to define new glacier scaling relationships and identify which geometries will be most problematic for modeling.

1. INTRODUCTION

Glaciologists already understand that some fundamental questions are intractable. Consider the well-studied problem of inversions: we can create a very accurate picture of velocities on the surface of a glacier, but even with a precise description of the physics, this surface information cannot be leveraged into a fine-scale description of velocities at the base of the glacier. Errors at the surface, no matter how infinitesimally small, will grow exponentially and swamp the basal description. This is not a matter of numerical precision or technique; the limitation is mathematically provable, and the only way to know fine-scale basal velocities with certainty is to measure them directly (Balise and Raymond, 1985; Bahr and others, 1994).

As with other fluidic systems, glaciologists extract meaning from glaciers by defining appropriate physics in the form of continuum mechanics. For simple systems like Poiseuille flow, analytical solutions give complete descriptions. For ever-so-slightly more complicated systems, like a glacier with two branches that flow together at a confluence, analytics fail, and numerics are used as approximations. In the continuum limit, these numerics can accurately reproduce the available physics. So the problem of modeling very complex real systems like the multi-branched Columbia Glacier, Alaska, USA, (Fig. 1) would appear to boil down to a process of refining our understanding of the physics (like a sorely needed improvement to the basal sliding relationship, and an increasingly accurate ice rheology) and refining the accuracy and resolution of our numerical techniques.

Unfortunately, as demonstrated below, the question of intractability goes much deeper. As the glacier system becomes increasingly complex, even the numerics are destined to fail. No amount of improvements in the physics can rescue our modeling, and it is not an 'analytics-to-numerics-to-something better' hierarchy where we need yet another approach. Instead, like the speed of light, there is a provable and inviolate wall beyond which the best model of a glacier is simply the glacier itself.

Why glaciers and not other geophysical systems? In fact, the same arguments may apply to any number of other branching networks like rivers, but the role of glaciers and ice sheets in climate change makes a deeper understanding of glacier flow pressingly relevant. Roughly 10% of the Earth's surface is covered in high-albedo glacier ice that directly affects climate. While all glaciers and ice sheets wax and wane with changes in regional precipitation and temperature, recent satellite data indicate that the Greenland ice sheet's response to warming may be far more sensitive than previously believed (Luckman and others, 2006; Howat and others, 2007). In this case, glacier flow rather than surface balance may play a dominant role in short-term climate and sea-level changes (Howat and others, 2007). To elucidate and predict this behavior, the next generation of climate models will need significantly higher-resolution descriptions of glacier response to atmospheric forcing (Luckman and others, 2006; Howat and others, 2007). While improved physics will certainly enhance these models, at some point the inevitable limitations in our models of glaciers will translate to limitations in climate models. Because atmospheric and ocean circulation components already have their own complexities (that strain the limits of computational hardware) any limitations in glacier modeling become very concrete questions of allocating computing resources between the ocean, atmosphere and solid-Earth components. Therefore, this analysis investigates those situations in which glacier models will fail.

The subsequent development differs from standard continuum mechanical arguments, and in the following sections, fundamental limits in glacier flow models are derived using techniques from basic computability theory. In particular, sections 2 and 3 lay a theoretical foundation by demonstrating a correspondence between glaciers and computer algorithms. Glacier confluences behave as logic gates and can be used as information processors. To relate this otherwise abstract derivation to real glaciers, section 4 briefly explores building and using glacier logic gates. However, for somewhat obvious reasons, these developments are

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

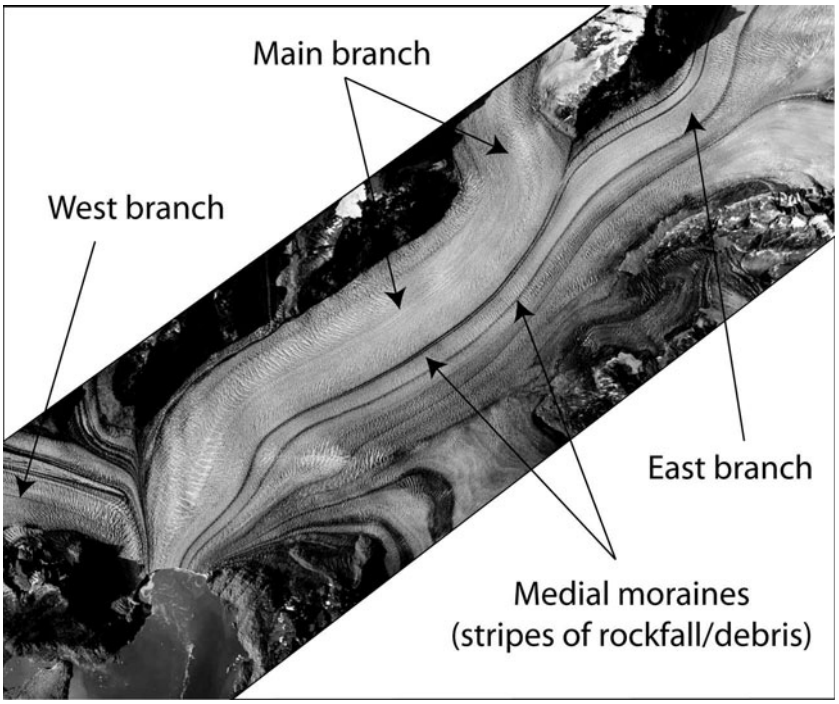

Fig. 1. Aerial view of Columbia Glacier, 2005. Several branches of the glacier network are visible, flowing from the top right to the bottom left and terminating in the waters of Prince William Sound. The black stripes are formed from rock, which falls from cliffs and is then advected along the glacier. (Image courtesy of W.T. Pfeffer, Institute of Arctic and Alpine Research, University of Colorado.)

not intended as a practical prescription for building functioning processors or 'computers' out of glaciers. Instead, section 5 shows that the real value lies in the correspondence between glaciers and algorithms, which allows important inferences from computational theory. In particular, applying a well-known theorem in computer science proves that not all glaciers can be modeled with a single numerical scheme (no matter what scheme is selected, some glaciers cannot be accurately modeled). In section 6, several new directions are proposed, and Shannon's entropy is hypothesized as a relevant parameter for identifying those glaciers that are most likely to cause difficulties for modelers.

2. GLACIERS AS INFORMATION PROCESSING

Consider a relatively simple glacier with no branches that follows a straight line. A cube of ice at the head of the glacier is advected downstream to the terminus, but arrives sheared and deformed. Nevertheless, the original cube of ice has been transported and in effect has transmitted information from the top of the glacier to the terminus. If we are clever, we can measure physical properties (like chemical composition) within the deformed cube to infer information about the ice at the time it started its journey. The glacier has acted like a slow wire that takes a signal in and translates (some distance away) to a signal out. Instead of electrons, the glacier 'wire' transmits packets of ice, rock and any other debris entrained in the ice.

We can continue the analogy and ask about information that is transmitted through a glacier with two branches meeting in a single confluence. In this case, 'information' (e.g. a cube of ice, or rocks that fall on the glacier from the valley walls) is input at the head of each branch and then combined at the confluence and transmitted to the terminus (Fig. 1). Depending on the values x and y that are passed in, the combined flow of the glacier determines what value $f(x, y)$ comes out.

Obviously, f will be non-trivial and non-linear. By restricting x and y to information about geometric position in space, the following section demonstrates that f processes the ice, rocks and other information as a Boolean logic gate (Fig. 2). In fact, by adding delays to the transmission of information (e.g. by adding a sinuous curve in the glacier's path), f becomes 'universal', meaning that multiple copies of f can be combined to construct all other possible Boolean functions (analogous to 'universal' basis vectors that can be combined to reach any point in a vector space).

In computability theory, the implication is immediate: universal logic gates can be combined to solve any algorithm. Therefore, every glacier is a representation of a particular algorithm, and for any given algorithm we could map the topology of a corresponding glacier. A consequence is that we can now use principles of computer science (which deals with algorithms) to understand many aspects of glaciology.

One algorithm is particularly important: the universal or programmable algorithm. This specially constructed algorithm accepts 'programs' as inputs, and then outputs the result of the program (e.g. Hopcroft and others, 2001, p. 377–379). In computability theory, this is the definition of a 'computer'. In other words, a program could be written in Basic, Java or any other language, and then compiled into a series of 1's and 0's. Then rocks could be positioned as information (1's and 0's) at the head of a universal glacier that represents the universal algorithm. This hypothetical universal glacier would process the information and effectively 'run' the program. Mapping the geometry of such a universal glacier would be interesting, but difficult, impractical, and unnecessary. In the subsequent analyses it is enough to know that such a mapping is theoretically possible.

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

While unusual, the notion that a glacier can behave as a universal or 'all-purpose' computer is not entirely unexpected. Computers come in two flavors: specialized and limited, like a digital wristwatch, or universal and general-purpose like a desktop PC. Virtually every physical process in nature manipulates some form of information and thus performs specialized computations (Wolfram, 1985; Landauer, 1991; Adamatzky, 2002, in preface; Lloyd, 2002). However, examples of universal computers have been far less common, though diverse, based for example on silicon chips, DNA, neural nets, collective behavior, molecular arrays, quantum mechanics, collision systems and specialized fluid-flow geometries (e.g. Siegelmann and Sontag, 1995; Solé and Delgado, 1996; Nielsen and Chuang, 1997; Adamatzky, 2002; Benenson and others, 2004; De Silva and others, 2006; Prakash and Gershenfeld, 2007). Because of the search for practical devices, most known universal systems are microscopic, and in most cases the construction of general-purpose computation is by design and is rarely an accidental by-product. Nevertheless, the complex interactions of many natural systems suggest that the universe must be replete with macroscopic examples of universal computers formed as fortuitous side effects of their underlying physics (Zuse, 1969; Wolfram, 1985; Forrest, 1990; Fredkin, 1990; Langton, 1990; Crutchfield and Mitchell, 1995; Lloyd, 2000). Branching glaciers are an example of such fortuitous physics, and they are as algorithmically capable as any desktop computer, albeit ironically slow and outrageously impractical in comparison.

3. GLACIERS AS COMPUTERS

Using a glacier confluence as a logic gate necessitates a shift in philosophy. In particular, experience tells us that logic gates and computers require electricity. The differences in electric potential along a wire generate high and low voltages that are interpreted as 1's and 0's, the traditional building blocks of information. While this has been historically and physically convenient, there is no fundamental requirement that computers use electricity or even wires (e.g. Lloyd, 2000; Adamatzky, 2002). Instead we can build computers from arbitrary particles (rather than electrons) that travel along arbitrary networks (rather than wires) under arbitrary forces (rather than electrical fields).

In the context derived here, three conditions specify a subset of naturally occurring networks that can compute. The networks must have: (a) a sense of direction, as in a directed graph; (b) a metric for distance along the graph; and (c) mobile particles, solitons, or other non-diffusive entities that flow along the graph. Each of these particles must move at identical speeds under identical physical conditions, and intersecting and adjacent particles must maintain their unique identity (Fig. 3). Examples include rock falling from nearby cliffs and being advected down the many branches of a large glacier such as Columbia Glacier (Fig. 1) or flotsam transported down the many branches of a large river system such as the Mississippi. The rockfall and flotsam 'particles' need not be uniform in size or other properties as long as these differences do not affect their movement. Other examples include blood cells in circulatory systems, cars in traffic, sewage systems, and even ski moguls which migrate counter-intuitively uphill along branching ski trails (http://academic.regis.edu/dbahr/moguls.htm).

input 1 (branch 1)	input 2 (branch 2)	**AND** gate output (below confluence)
0	0	0
0	1	0
1	0	0
1	1	1

input 1 (branch 1)	input 2 (branch 2)	**OR** gate output (below confluence)
0	0	0
0	1	1
1	0	1
1	1	1

single input (above glacier's curve)	**NOT** gate output (below curve)
0	1
1	0

input 1	input 2	**NAND** gate output
0	0	1
0	1	1
1	0	1
1	1	0

Fig. 2. Truth tables for logic gates. Each gate has one or two inputs whose values can be 0 or 1 as shown on the left (sometimes referred to as false = 0 and true = 1). The output of each gate is shown on the right. The output of a NAND gate is the same as an AND gate whose output is fed to a NOT gate. A variety of other gates can be specified by switching values at the outputs. However, all possible gates can be constructed by stringing together sequences of NAND gates (Mano and Ciletti, 2007). The text shows how to build AND gates from glacier confluences (with incoming branches 1 and 2) and NOT gates from sinuous curves in a glacier path. Together a glacier confluence and curve form a universal NAND gate.

To perform Boolean logic, all rockfall or other particles are compared with the movement of a reference particle, either real or hypothetical. Given a unit length scale L, each particle will fall either an even or odd number of units from the reference at a given time t. Over a time interval τ, some number of even and odd particles will pass through any particular cross-section of a branch on the glacier network. If the majority of particles are even (or odd) then the Boolean value at that cross-section is 0 (or 1). If the distributions are equal, then the Boolean value is arbitrarily assigned to match the leading particle(s).

More explicitly, let the reference particle move a distance $d_x(t)$ along a hypothetical straight line deemed the x axis. If $p_x(t)$ is the x position of another particle on the glacier, then $\text{ceiling}(|[d_x(t) - p_x(t)]/L|) \bmod 2$ gives its value as even (0) or odd (1) where the ceiling function rounds up all fractional values. Although particles can move laterally and vertically through (and on) the ice, only the difference along the specified x axis is measured.

Note that for a total of N particles at some position on a glacier, there are $N/2$ possible ways to have a majority of even (or odd) particles and therefore $N/2$ possible ways to represent a 0 (or 1). Except near $N/2$, the Boolean is relatively insensitive to noise (variations in the number of particles), but to prevent unexpected changes in value, the particles should not diffuse significantly. In other words, over

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

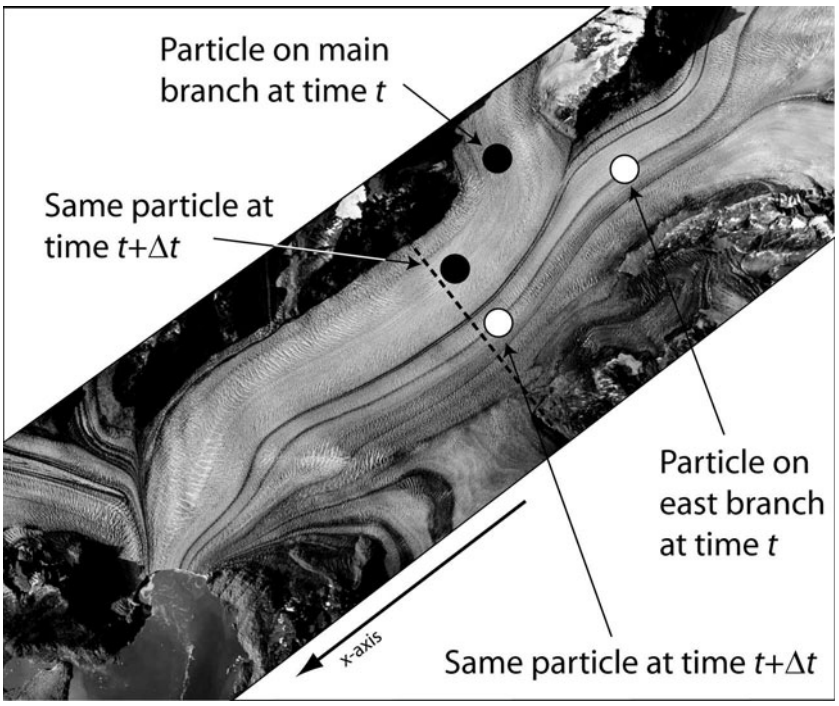

Fig. 3. Logic gates are formed by particles moving along branches of a glacier and merging at a confluence (shown as the dashed line). Analogous to electrons moving along two wires that join into a single wire, the merging particles do not occupy the same physical space at the confluence. The particles may move laterally and even vertically by following plunging and emerging flow vectors, but the Boolean value assigned to each particle is given only by its x position relative to a hypothetical particle traveling at a uniform speed along an x axis (see text and Fig. 4). All particles that reach the confluence (dashed line) within a specified time interval are assigned Boolean values; if the majority of particles over that time interval are 0's (or 1's) then the logic gate's output value is a 0 (or 1).

the timescale of the computation τ, the length scale of diffusion must be small relative to L, a reasonable restriction for many advective systems like rocks transported by glaciers where the rock's flow velocities are large compared to diffusion.

Universal computation requires a logically complete gate such as NAND or NOR from which all other circuits can be assembled (Fig. 2) (cf. Mano and Ciletti, 2007). A NAND gate is an AND gate followed by a NOT gate, and a NOR gate is an OR followed by a NOT. Towards this end, a NOT gate (which inverts its single input by switching a 0 to a 1 and a 1 to a 0) can be constructed by lengthening a path of the glacier by $i \times L$ for any odd integer i (equivalently, the velocity of the particle may be slowed so that the particle appears to have taken a longer path). The arrival of each particle is delayed by i units (Fig. 4), thereby swapping even and odd particles and flipping the Boolean value.

AND and OR gates are any confluence on the glacier. An AND gate outputs a 1 if both inputs are 1, and outputs 0 otherwise (Fig. 2). OR gates output a 1 if either or both inputs are a 1 (Fig. 2). So if the total number of incoming particles is N and M (over the time interval τ) and n and m of these are odd particles (1's), then the intersection behaves as an AND gate when either (a) $n+m<(N+M)/2$ or (b) both $n \geq N/2$ and $m \geq M/2$ (see shaded regions and explanation in Fig. 5). Otherwise the intersection behaves as an OR gate. Combined with a NOT gate, each confluence behaves as a NAND or NOR.

To build arbitrary functions with multiple gates, we can assume combinatorial logic (Mano and Ciletti, 2007) with fan-out and crossover achieved by repeating the inputs and circuit logic as necessary. For the moment, assume that each glacier confluence will always function as a NAND. For all possible particle arrangements, each NAND gate will give the correct answer only three-quarters of the time. The occasional wrong answer comes from the incorrect assumption that the gate will always behave as a NAND. When the inputs are 00 and 11, the NAND and NOR gates behave identically, but when the inputs are 01 and 10, the gate fails half the time and behaves as a NOR gate (marked as the white shaded regions in Fig. 5). Therefore, with a total of G gates, the correct output of the multiple gate circuit decreases exponentially as $(3/4)^G$. However, there always exists an arrangement of input particles that will correctly pass through the gates satisfying conditions (a) and (b) in the previous paragraph (see Appendix). This is analogous to the behavior of universal programmable quantum circuits (Nielsen and Chuang, 1997; Bužek and others, 2004), which require non-deterministic gates that succeed only with probability 1/4.

Although we are unlikely to try an actual computation with a glacier (see next section), using a confluence as a NAND gate requires a mechanism for identifying failed calculations and separating them from successful calculations. Practically, we would set rocks (or other particles) at appropriate positions to represent 0's and 1's on each branch of the glacier above the confluence. At some later time, we would read the output value below the confluence by interpreting the new position of the rocks as 0's and 1's. At that later time, we would know if the gate has failed by reading the original inputs (I_1 and I_2) and the output (O_1). If any triplet (I_1, I_2, O_1) for a single gate is 010 or 100, then the gate did not behave as a NAND. Note that although erroneous, 000 and 111 are not possible because gate inputs of 00 and 11 always give the correct answer (triplets 001 and 110) (Fig. 5).

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

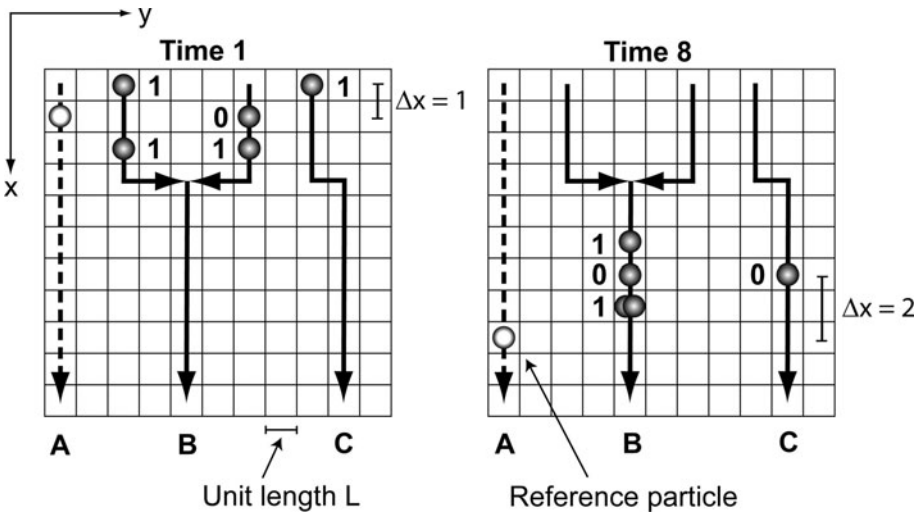

Fig. 4. Assigning Boolean values to particles on a glacier network. The two grids represent different times separated by an interval $\tau = 7$. On each grid, part A shows a hypothetical reference particle moving along the x axis. Part B shows a stylized version of a confluence like that on Columbia Glacier in Figure 3. Part C shows a stylized section of a glacier that has a sinuous curve but no confluence. For convenience, assume each particle moves one unit L per time-step in the directions indicated by the network (in general, the velocities may vary). Each particle falls an even or an odd distance from the reference A (as measured along the x axis only) and represents a 0 or 1 respectively. In total, the left input of B is assigned value 1 because the majority of particles are 1's. The right input of B has an equal distribution and is assigned a 1 because the leading particle is a 1 (see text). Over the time interval $\tau = 7$, the particles in B have merged onto a single path, but each particle retains its unique identity (see Fig. 3). After merging, the majority of particles are odd and the output of the gate is a 1 (consistent with an AND gate). In C, the particle starts as a 1 but ends as a 0 due to the delay of one unit L caused by the sinuous curve. Therefore C represents a NOT gate.

Therefore, to check an entire glacier circuit, take each of the circuit's original outputs and add $3G$ additional outputs, one triplet for each of the circuit's G gates. Each new triplet gives the input and output values for one of the gates (Fig. 6) and acts as error checking. Read all of these new values to see if the original output is acceptable or uncertain. Conveniently, many glaciers have fractal or self-similar topologies (Bahr and Peckham, 1996), which roughly means that any selected subsection of a glacier will be topologically identical to another subsection. In other words, any subset or sequence of confluences (gates) is repeated elsewhere, so copies of the gates are readily available for error checking. Even if copies of the gates are not available, we can set up the simulation and return to the glacier each year to relocate every particle and check each individual gate for errors.

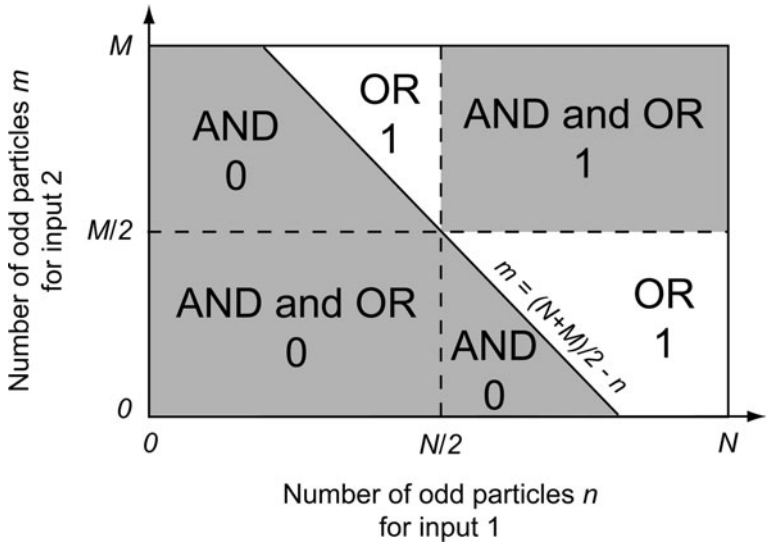

Fig. 5. Regions of AND versus OR behavior. Consider a glacier confluence (as in Fig. 3 and Fig. 4 part B) where each input has n and m odd particles out of a total of N and M particles. The inputs represent 0 or 1 depending on whether or not the odd particles are in the minority or the majority. For the given inputs, the output of the intersection will behave as either an AND or an OR gate (or both). In particular, for any point in the parameter space, the output is 0 or 1 as indicated. In general, if an AND gate is desired (gray regions) then one-quarter of the parameter space inappropriately behaves as an OR gate (white regions). The line separating 0's from 1's is derived from $n + m = (N + M)/2$, the condition that separates a minority from a majority of odd particles at the output (an output of 0 versus 1). The Appendix proves that if we desire only AND gates, then some input arrangement of particles (n, m, N and M) exists so that only the gray shaded region of the parameter space is reached at every glacier confluence.

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

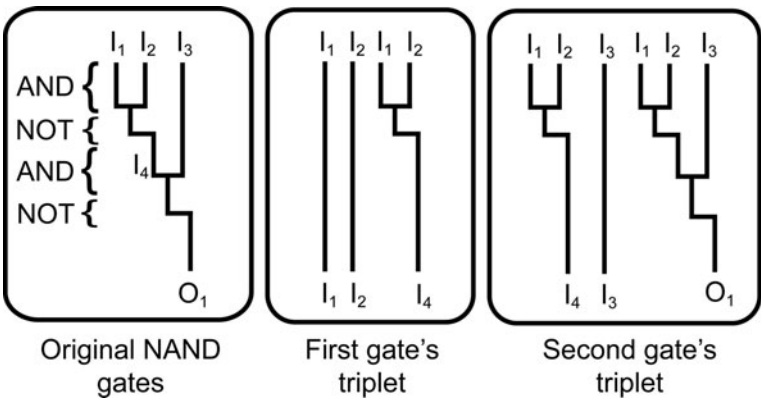

Fig. 6. Example of error-checking triplets. The original circuit in the first box contains two stylized NAND gates constructed from ANDs and NOTs. For all possible inputs I_1, I_2, I_3 (i.e. particle arrangements), this circuit will give the correct answer only $(3/4)^2$, or 9/16, of the time and must be error-checked. As shown, six additional outputs are required to check the original output O_1. The second box shows the circuits used to error-check the upper NAND gate. If the triplet (I_1, I_2, I_4) is (0, 1, 0) or (1, 0, 0), then the gate has failed. The third box shows the circuits and triplet (I_4, I_3, O_1) used to error-check the lower NAND gate.

While successful computations decrease exponentially with the number of gates, an exponential number of simulations can be run in parallel by differentiating glacier-rockfall particles based on 'color', size, weight, chemical tag or other attribute. Different-colored particles can be placed simultaneously at each input (upstream locations) and evaluated separately at the output (downstream locations). Similar to other parallel systems (e.g. Forrest, 1990; Nielsen and Chuang, 1997), with I inputs we can run 2^I simultaneous calculations, one for each possible set of input values. Also, for each input 0 or 1, we can run all possible $N/2$ arrangements of particles that give that input, thereby guaranteeing a successful computation.

4. USING A GLACIER COMPUTER

Is a glacier computer useful? A reasonable reaction is that such a computer must be absurdly slow and therefore of little interest either practical or theoretical. However, surprisingly, if speed is considered the only issue, then there are many scenarios where glaciers can outperform traditional desktop computers.

Suppose we wish to factor a number or to find the shortest path between five points. First we need to find a glacier that can solve this problem. Using the scheme outlined above, each glacier solves a particular algorithm that corresponds to that glacier's particular branching topology. By considering only some of its branches and ignoring others, every subset of a glacier also solves some (probably different) algorithm. If we specify a particular problem, then larger multi-branched glaciers are most likely to have an appropriate subset.

Certainly, a practical consideration is that if a glacier or glacier subset cannot be found, then building a glacier from scratch is hardly a viable option. Furthermore big algorithms (like a spreadsheet or a numerical glacier flow model) would require tremendously large glaciers with hundreds of thousands of branches that exceed the size of anything found in nature. However, if the reasons were compelling, scaled glaciers could be constructed in a laboratory with a centrifuge. While seemingly far-fetched, similar fluid-based logic gates have been miniaturized for microscale chemical detectors and for use in nuclear environments that are unfriendly to traditional electronics (Prakash and Gershenfeld, 2007). These fluidic gates are constructed on a different principle that takes advantage of turbulence rather than advection of particles, but the motivation for miniaturization is similar.

Assuming an appropriate glacier can be found, running the algorithm requires input and output. First we create a reference particle. (It does not need to be real. Just assume that a particle moves at a fixed speed along an x axis of our choice.) Then we assemble a pile of colored rocks, chemically treated rocks, radioactive rocks, or other particles that can in some way be traced and identified over long periods of time. To specify an input, we place these rocks at the head of each branch of a glacier (or glacier subset) so that the majority of the rocks represent 1's or 0's as desired. A rock is a 1 if it is an odd distance from the reference particle, and it is a 0 if it is an even distance from the reference particle (as measured along the x axis). We then read the results by counting the number and position of the rocks at a later time. While unintuitive, this unconventional interface might not seem foreign to someone who is, for example, blind and unable to use standard keyboards and monitors. The rocks are functioning like a version of Braille: run your hand over the terminus and 'feel' the location of the rocks to interpret the result. Veterans of 1970s-era punch-card computers would also feel more at home with the unusual interface.

Ignoring other practical considerations, time would seem to be the biggest constraint. Who will wait hundreds of years for the results of a simple calculation? Oddly enough, because glaciers can simultaneously solve all possible inputs (using 'color'-differentiated rocks), they have more in common with the fast parallel processing than with regular desktop sequential computing. Ironically, this means that glacier computers can solve certain tasks (like factorizations) much faster than desktop computers or even (sequential) supercomputers. Suppose a glacier takes on the order of 100 years to solve a single problem, but with all possible input values simultaneously. The desktop computer might solve the same problem in 100 seconds but only for a single set of input values. If we wish to see the result for every possible set of input values, then for any problem with greater than 31 536 000 possible arrangements of the inputs (the number of seconds per year), the desktop computer will take more than 100 years to complete its task and the glacier

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

will finish more quickly. Such problems are very common: this example corresponds to a binary input with only 24 bits.

However, practically speaking, problems with 24 inputs will require the placement of a minimum of 31 536 000 rocks or other particles on the glacier. Parallel simulations with bigger algorithms (like spreadsheets and flow models) will require many more inputs and, consequently, an exponentially increasing number of rocks. With only 263 inputs, the number of rocks will exceed the number of atoms in the known universe ($\sim 2^{262}$) and there is no practical way to fit even a fraction of this many particles on a finite-sized glacier. Therefore, we have effectively traded a problem in time for a problem in space. To nobody's surprise, glaciers are ineffective as parallel computers, but not because they are slow. Glaciers are too small.

5. IMPLICATIONS FOR MODELERS

While nobody is likely to use a glacier as a computer, the richly developed theory of computation does provide a valuable new leverage arm for understanding glacier behavior. While practically useless, glaciers as algorithms are theoretically invaluable. Because they can compute, we know that any theories about algorithms are also theories that apply to glaciers, and decades of theoretical developments in computer science can apply to glaciology. In essence, take any theorem that says 'algorithm' or 'computer', and substitute the word 'glacier'.

For example, computability theory tells us that our ability to simulate glaciers must be limited and approximate. Why? A well-known theorem in computer science says that we cannot write a single all-purpose computer program that determines whether or not any other arbitrary program will have some particular outcome (e.g. Hopcroft and others, 2001, p. 403–404). Simply reworded, we cannot write a single all-purpose numerical model that determines whether or not any arbitrary glacier will have some particular algorithmic outcome (i.e. move ice to the terminus in a particular manner).

The basis for this claim comes from computability theory which has firmly established that some problems cannot be solved by any computer or algorithm. A corollary of Rice's theorem (Rice, 1953) says that predicting any non-trivial outcome of a computer program is one of these unsolvable or 'undecidable' problems (e.g. Hopcroft and others, 2001, p. 403–404). In particular, every glacier is a computer program (implemented as a network of confluences) with inputs given by rockfall and other debris. Therefore, without violating Rice's theorem, we cannot write code in C, Java or any other language that will model all of our 'glacier computers' and always accurately predict their 'outcomes'. In other words, we cannot construct a reliable model of glacier flow that will work equally well for all arbitrary glaciers. At best, we can perform accurate simulations for a single glacier with a particular geometry; or because it is not excluded by Rice's theorem, we can make reasonable but less accurate approximations that will apply to many glaciers.

It is worth noting that if glaciologists could program numerical models that worked for all glaciers, then we could disprove Rice's theorem and ultimately solve undecidable propositions such as Turing's halting problem which is famously and provably intractable (Turing, 1936; Hopcroft and others, 2001, p. 380). In other words, if there are no limits to glacier modeling then we can upend decades of computer science.

None of this is meant to suggest that we cannot vastly improve our understanding of glacier physics and, concomitantly, glacier flow models. In practice, to increase numerical speed, most glacier simulations have used highly idealized geometries or simplifications of the continuum equations and physics. The good news is that this has been sufficient for most applications, including current global climate models. The bad news is that future climate models need higher-resolution flow models, and with or without advances in the physics, predicting glacier behavior is as hard as predicting an arbitrary circuit (the circuit specified by the glacier's geometry). No known algorithms can speed up circuit predictions, nor do computer scientists believe that one is likely to be found (e.g. Wolfram, 1985; Moore and Nilsson, 1999; Hopcroft and others, 2001, p. 413–434), so general-purpose closed-form solutions and fundamentally faster numerical models are exceedingly unlikely for complex glacier geometries.

In particular, parallelizing circuit-prediction algorithms is not thought possible, so parallelizing glacier circuits is also unlikely. Such inherently sequential problems are labeled 'P-complete' (e.g. Moore and Nilsson, 1999) and are among the slowest algorithms that are still considered tractable. Note that incremental speed improvements are possible and even assured with advances in hardware, but exponential improvements in speed are not possible.

Modelers, therefore, can and should continue as before, but they should be aware of the inherent limitations. General-purpose flow models can simulate and predict circuit behavior, so glacier modelers will never be able to take advantage of the gains anticipated with massively parallel-processing machines or quantum computing. Instead, to improve resolution and speed, flow models should take advantage of an ice sheet's (and/or glacier's) particular geometry rather than using generic boundary conditions applicable to all glaciers. This is theoretically equivalent to building a look-up table for the particular logic function that corresponds to the glacier, and such tables can provide a dramatic improvement in speed. However, if the ice-sheet/glacier geometry will change with time (to the extent that the corresponding glacier 'circuit' is altered in unknown ways), then for all but the simplest glaciers the look-up table becomes arbitrarily large and unknown. In this case, simulations will be forced to choose between accuracy and speed, and the loss in accuracy would have to be sufficient to negate any ability to predict the glacier's circuit output.

6. NEW DIRECTIONS

By treating glaciers as information processors, a vast array of developments are possible, all borrowing from advances in theoretical computer science. As an example of one possible direction, this section hypothesizes and explores the role of Shannon's information entropy. Rather than a complete development with supporting data, this section suggests some of the many possibilities for future study.

The algorithmic content or information content of a glacier is an excellent proxy for its geometric structure and dynamic behavior. Small valley glaciers, for example, are simple in shape, have few branches, and can only represent a few simple algorithms: one algorithm for each possible subset of the branching topology. However, a large glacier

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

like Columbia Glacier (Fig. 1) has thousands of branches (Bahr and Peckham, 1996) and can process many algorithms, both simple and complex. The number of algorithms grows exponentially with the number of branches.

Consider a small glacier with a positive mass balance that slowly accumulates ice and grows in size. The algorithmic content starts small but increases as the glacier acquires more branches. At some point the number of branches will reach a maximum, and beyond this size further increases in the glacier's area will start to inundate the topography. Once separate branches of the glacier will now breach passes, cover ridge lines and blend together. As the size continues to grow, the number of branches drops, and the algorithmic content once again decreases. At some point, most terrain features are buried and the glacier has grown into an ice field or ice cap with comparatively few branches. Once again, the algorithmic content is minimal.

This increase and then decrease in glacier algorithms is a characteristic feature of entropy, or, in this case, Shannon entropy which is a computer scientist's traditional measure of information content. An analogy is a parking lot. An empty lot, like a small glacier, is very ordered (low entropy) and can be described with very little information. As cars randomly fill the lot, the disorder increases and becomes increasingly difficult to describe without using lots of information about the specific location of the cars. When half full, the parking lot (like a large many-branched glacier) is maximally disordered (high entropy) and requires huge amounts of information to describe the location of each car. As the lot continues to fill, the once scattered cars start to blend together and there are fewer and fewer empty spaces. At this point the system has once again become highly ordered (low entropy), and relatively little information is necessary to describe the location of the remaining empty parking spaces. Eventually, like an ice field that has inundated all the terrain features, the parking lot is full, completely ordered, and is described with minimal information.

In physical systems, the change in entropy with a parameter like temperature will lead to a phase change and fundamental transformations in structure and behavior. So-called second-order or continuous phase transitions (like the superfluid and ferromagnetic transitions) reach a critical value where entropy diverges. Behaviors change above and below the critical value (e.g. paramagnetic versus ferromagnetic), and right at the critical value the system fluctuates between both. In computer science, second-order phase transitions are associated with the divergence of Shannon entropy or information content at a critical value. Simple binary models of computation called cellular automata, for example, will change from predictable to random behaviors at the critical value and are capable of universal computation only when near the critical value (Langton, 1990).

In the case of a glacier, we can hypothesize that the controlling parameter is size (or equivalently volume, length, etc.). As size changes, we should expect the entropy or algorithmic content to diverge at a critical value. The phase below the critical value should be characterized by valley glaciers, and the phase above the critical value should be characterized by ice fields and ice caps. Directly at the critical value should be maximally complex dendritic glaciers capable of universal computation. Confirming this hypothesis would involve a straightforward though time-consuming measure of glacier sizes and branching (algorithmic content).

Like other computational and physical systems (Ma, 1976; Stauffer and Arahony, 1992, p. 57–69), the geometrical and topological properties of glaciers near a critical point can be expected to grow as power laws and show associated fractal distributions. Large and presumably maximally complex glaciers like Columbia, Matanuska and Tazlina glaciers, Alaska, are known to have fractal topologies (Bahr and Peckham, 1996). While the critical point is only hypothesized in this analysis, the observed fractal topology of large glaciers supports the existence of such a phase transition.

Also, near the phase transition, physical and computational systems exhibit a variety of complex critical phenomena. In particular, most physical quantities will grow as power laws with the controlling parameter (Ma, 1976). In the case of glaciers, we should expect dynamic quantities like flux, velocity and response time to grow as a power law with size. In what manner this might be related to volume–area, response time and other widely observed glacier scaling relationships is an intriguing question. These previously observed scaling relationships (e.g. Jóhannesson and others, 1989; Bahr and others, 1997) apply to glaciers of all sizes, so these new critical-phenomena power laws might predict entirely different and as yet unobserved relationships that apply strictly to large dendritic glaciers. Alternatively, any connection between volume–area scaling and critical-phenomena scaling would provide a direct link between continuum mechanics (from which volume–area scaling can be derived) and the computation-theoretic approach developed here.

From a modeling perspective, the glaciers most likely to resist simulation are those which are capable of universal computation and can represent any algorithm, including any problematic algorithms. In other words, troublesome glaciers will have the most dendritic geometries nearest the phase transition where quantities are diverging. With size as a controlling parameter, a critical-sized glacier (at the phase transition) might be identified that cannot be modeled with sufficient speed or accuracy. While most glaciers might be simulated efficiently with a single numerical model, glaciers near the critical size could be dealt with separately using different models or with approximations. Presumably the critical size is on the order of 1000 km^2, like large multi-branched Alaskan glaciers (e.g. Columbia, Tazlina, Matanuska and Barnard glaciers); but if possible, identifying this size with data could be invaluable to modeling efforts.

7. DISCUSSION AND CONCLUSIONS

Continuum mechanics rather than computability theory is the traditional language of glaciology. The differences are notable, so the basic arguments can be summarized as follows:

1. Glaciers process information by moving rocks and other particles along their branches (Fig. 1). Confluences function as logic gates (Figs 2–4).

2. All computational algorithms can be constructed from a network of logic gates, or in this case, a network of glacier confluences. Therefore, every glacier represents an algorithm, and every algorithm has a corresponding glacier topology. Due to this equivalence, the richly developed theory of computation applies to glaciology.

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

3. In particular, computational theory tells us that there must be limits to numerical glacier models, just as there are well-known limits in computer science. If there were no model limitations, then a fundamental theorem of computer science would be wrong.

4. Shannon's entropy is hypothesized as a relevant measure of glacier complexity. Maximally complex, very large multi-branched glaciers (e.g. Columbia Glacier) have the greatest entropy and are most likely to cause difficulties for modelers. However, these maximally complex glaciers may also harbor new scaling relationships that could advance our understanding of glaciers.

5. Other theorems and insights from the theory of computation can be applied to glaciology, and some may yield novel results.

Although they are not designed as computers, glaciers advect particles along directed graphs, resulting in computation as an emergent property. As a result, flow models fall into a class of problems known as 'P-complete', the slowest and most complex set of algorithms that are still considered tractable by computer scientists. The complexity of the glacier simulations has nothing to do with particular finite-difference, finite-element or other numerical schemes. The complexity is inherent in the flow of the glacier itself. In order to correctly model the flow of ice from the headwall to the terminus of an arbitrary multi-branched glacier, we have to be able to predict the outcome of arbitrary Boolean logic circuits formed by the glacier's topology. Such circuit predictions can only be done with non-parallelizable and inherently slow serial techniques.

In a limited sense, we can speed the simulation of glacier flow by simplifying the model. If glaciers represent circuits, then we can create a list of all possible Boolean circuits for a given glacier geometry and pre-tabulate the outcomes. Glacier modeling becomes an exercise in formulating appropriate look-up tables: 'given x amount of ice accumulating at the head of this branch, and y amount of ice on this other branch, then the look-up table says we will get $f(x,y)$ ice at time t at the terminus'. However, a particular look-up table will work for some glaciers but cannot work for all glaciers. In fact, a fundamental limit of glacier simulations is that no single flow model can work for all possible glaciers. All models must fail for some glacier.

The glaciers most likely to cause difficulties for flow models are the large multi-branched glaciers capable of universal computation. The preceding analysis does not identify particular geometries, but strongly suggests that there is a critical size. Glaciers at the critical size would resist attempts to model, while smaller and larger glaciers should be more amenable to analysis. The critical size could be identified from data as the point where the Shannon or information entropy of a glacier diverges in a phase transition. A host of new glacier scaling relationships might also be identified near the phase transition.

Based on these results, modelers should not discard their numerics but should understand that the current approaches to flow modeling are simply the best that we can do. When climate models call for faster and more accurate glacier flow simulations, modelers should continue to refine their understanding of the physics and its incorporation into any particular numerical scheme. Model funding should also continue to be allocated for faster processors and memory which can always provide incremental improvements in speed. However, glaciologists should not wait for advances in parallel computing or any other technology to fundamentally improve the speed and accuracy of glacier simulations. Instead, for those situations where continuum models have limitations, these results suggest using a computational-theoretic approach to a deeper understanding of glacier behavior.

REFERENCES

Adamatzky, A., *ed.* 2002. *Collision-based computing.* London, Springer-Verlag.

Bahr, D.B. and S.D. Peckham. 1996. Observations and analysis of self-similar branching topology in glacier networks. *J. Geophys. Res.,* **101**(B11), 22,511–22,521.

Bahr, D.B., W.T. Pfeffer and M.F. Meier. 1994. Theoretical limitations to englacial velocity calculations. *J. Glaciol.,* **40**(136), 509–518.

Bahr, D.B., M.F. Meier and S.D. Peckham. 1997. The physical basis of glacier volume–area scaling. *J. Geophys. Res.,* **102**(B9), 20,355–20,362.

Balise, M.J. and C.F. Raymond. 1985. Transfer of basal sliding variations to the surface of a linearly viscous glacier. *J. Glaciol.,* **31**(109), 308–318.

Benenson, Y., B. Gil, U. Ben-Dor, R. Adar and E. Shapiro. 2004. An autonomous molecular computer for logical control of gene expression. *Nature,* **429**(6990), 423–429.

Bužek, V., M. Ziman and M. Hillery. 2004. Probabilistic programmable quantum processors. *Fortschr. Phys.,* **51**(11–12), 1056–1063.

Crutchfield, J.P. and M. Mitchell. 1995. The evolution of emergent computation. *Proc. Natl. Acad. Sci. USA (PNAS),* **92**(23), 10,742–10,746.

De Silva, P., M.R. James, B.O.F. McKinney, D.A. Pears and S.M. Weir. 2006. Molecular computational elements encode large populations of small objects. *Nature Mater.,* **5**(10), 787–789.

Forrest, S. 1990. Emergent computation: self-organizing, collective, and cooperative phenomena in natural and artificial computing networks: introduction to the proceedings of the ninth annual CNLS Conference. *Physica D,* **42**(1–3), 1–11.

Fredkin, E. 1990. An informational process based on reversible universal cellular automata. *Physica D,* **45**(1–3), 254–270.

Hopcroft, J.E., R. Motwani and J.D. Ullman 2001. *Introduction to automata theory, languages and computation. Second edition.* Boston, MA, Addison Wesley.

Howat, I.M., I.R. Joughin and T.A. Scambos. 2007. Rapid changes in ice discharge from Greenland outlet glaciers. *Science,* **315**(5818), 1559–1561.

Jóhannesson, T., C. Raymond and E. Waddington. 1989. Time-scale for adjustment of glaciers to changes in mass balance. *J. Glaciol.,* **35**(121), 355–369.

Landauer, R. 1991. Information is physical. *Phys. Today,* **44**(5), 23–29.

Langton, C.G. 1990. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D,* **42**(1–3), 12–37.

Lloyd, S. 2000. Ultimate physical limits to computation. *Nature,* **406**(6799), 1047–1054.

Lloyd, S. 2002. Computational capacity of the Universe. *Phys. Rev. Lett.,* **88**(23), 237,901–237,904.

Luckman, A., T. Murray, R. de Lange and E. Hanna. 2006. Rapid and synchronous ice-dynamic changes in East Greenland. *Geophys. Res. Lett.,* **33**(3), L03503. (10.1029/2005GL025428.)

Ma, S. 1976. *Modern theory of critical phenomena.* Redwood City, CA, Addison Wesley.

Mano, M.M. and M.D. Ciletti 2007. *Digital design. Fourth edition.* Upper Saddle River, NJ, Pearson International.

Moore, C. and M. Nilsson. 1999. The computational complexity of sandpiles. *J. Stat. Phys.,* **96**(1–2), 205–224.

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press

Nielsen, M.A. and I.L. Chuang. 1997. Programmable quantum gate arrays. *Phys. Rev. Lett.,* **79**(2), 321–324.
Prakash, M. and N. Gershenfeld. 2007. Microfluidic bubble logic. *Science,* **315**(5813), 832–835.
Rice, H. 1953. Classes of recursively enumerable sets and their decision problems. *Trans. Am. Math. Soc.,* **74**(2), 358–366.
Siegelmann, H.T. and E.G. Sontag. 1995. On the computational power of neural nets. *J. Comput. Sys. Sci.,* **50**(1), 132–150.
Solé, R.V. and J. Delgado. 1996. Universal computation in fluid neural networks. *Complexity,* **2**(2), 49–56.
Stauffer, D. and A. Aharony 1992. *Introduction to percolation theory. Second edition.* London, Taylor and Francis Ltd.
Turing, A.M. 1936. On computable numbers with an application to the Entscheidungsproblem. *Proc. London Math. Soc.,* 2nd ser., **42**, 230–265.
Wolfram, S. 1985. Undecidability and intractability in theoretical physics. *Phys. Rev. Lett.,* **54**(8), 735–738.
Zuse, K. 1969. *Rechnender Raum.* Braunschweig, Friedrich Vieweg.

APPENDIX

Existence theorem: An arrangement of even and odd particles exists at the inputs such that every intersection in a directed-graph circuit is treated as an AND gate (rather than an OR gate). (Adding delays trivially creates universal NAND gates.)

Proof: The truth table for AND is (1) inputs 0 and 0 give an output 0; (2) inputs 0 and 1 give an output 0; (3) inputs 1 and 0 give an output 0; and (4) inputs 1 and 1 give an output 1 (Fig. 2). Therefore, it is sufficient to demonstrate that for any possible arrangement of output particles on any given gate, there must exist corresponding inputs (for each of the four cases listed above) that produce this output in a manner consistent with an AND gate. Each of the four possible cases is addressed separately below.

Consider the output of a gate $O = (n, N)$, where N is the total number of particles at the output, and n is the number of odd particles at the output.

1. If the output is 0, then one set of the possible AND gate inputs is 0 and 0. Note that for an output 0, $n < N/2$. So there exist inputs $I_1 = (n/2,\ N/2)$ and $I_2 = (n/2,\ N/2)$ which will both be 0 as required. (Fractional values may be rounded as appropriate. See below.)

2. If the output is 0, then one set of the possible AND gate inputs is 0 and 1. Consider inputs $I_1 = (n-m,\ N-M)$ and $I_2 = (m,\ M)$, where (a) $m > M/2$ and (b) $M < N$. Condition (a) ensures that input I_2 will be 1. Condition (b) ensures that I_1 is well defined. Also note that $n-m < n-(M/2) < (N/2)-(M/2)$ by condition (a) and because $n < N/2$ (as required for the output to be 0). Therefore $n-m < (N-M)/2$ and I_1 is 0 as required.

3. If the output is 0, then one set of the possible AND gate inputs is 1 and 0. Then inputs $I_1 = (m,\ M)$ and $I_2 = (n-m,\ N-M)$ will be 1 and 0 as required (by the same argument as above).

4. If the output is 1, then the AND gate requires inputs 1 and 1. Note that for an output 1, $n > N/2$. So there exist inputs $I_1 = (n/2,\ N/2)$ and $I_2 = (n/2,\ N/2)$ which will both be 1 as required.

In the degenerate case where $n = N/2$, rounding may not work. However, the same proof then applies by doubling all particles at all of the inputs. This guarantees that $n/2$ and $N/2$ are whole numbers, and we choose the leading particles to be even (0) or odd (1) as necessary.

MS received 27 June 2008 and accepted in revised form 26 October 2008

https://doi.org/10.3189/002214309788608831 Published online by Cambridge University Press