John H. Reif, Mechanical Computation: it's Computational Complexity and Technologies, invited chapter, Encyclopedia of Complexity and System Science (edited by Robert A. Meyers), Springer-Verlag (2009), pp. 1821-1836, ISBN: 978-0-387-75888-6. DOI: 10.1007/978-0-387-30440-3_325.

Mechanical Computation: its Computational Complexity and Technologies Chapter, Encyclopedia of Complexity and Systems Science

John H. Reif[1]

Article Outline

Glossary

Mechanism: a machine or part of a machine that performs a particular task computation: the use of a computer for calculation.

Computable: Capable of being worked out by calculation, especially using a computer.

The term *simulation* will be both used to denote the modeling of a physical system by a computer, as well as the modeling of the operation of a computer by a mechanical system; the difference will be clear from the context.

[1] Department of Computer Science, Duke University, Box 90129, Durham, NC 27708-90129 USA and Adjunct, Faculty of Computing, King Abdulaziz University (KAU), Jeddah, Saudi Arabia. E-mail: reif@cs.duke.edu. This work has been supported by NSF grants NSF Grants CCF- 1217457 and CCF-1141847.

I. Definition of the Subject and Its Importance

Mechanical devices for computation appear to be largely displaced by the widespread use of microprocessor-based computers that are pervading almost all aspects of our lives. Nevertheless, mechanical devices for computation are of interest for at least three reasons:

(a) ***Historical:*** The use of mechanical devices for computation is of central importance in the historical study of technologies, with a history dating to thousands of years and with surprising applications even in relatively recent times.

(b) ***Technical & Practical:*** The use of mechanical devices for computation persists and has not yet been completely displaced by widespread use of microprocessor-based computers. Mechanical computers have found applications in various emerging technologies at the micro-scale that combine mechanical functions with computational and control functions not feasible by purely electronic processing. Mechanical computers also have been demonstrated at the molecular scale, and may also provide unique capabilities at that scale. The physical designs for these modern micro and molecular-scale mechanical computers may be based on the prior designs of the large-scale mechanical computers constructed in the past.

(c) ***Impact of Physical Assumptions on Complexity of Motion Planning, Design, and Simulation***
The study of computation done by mechanical devices is also of central importance in providing lower bounds on the computational resources such as time and/or space required to simulate a mechanical system observing given physical laws. In particular, the problem of simulating the mechanical system can be shown to be *computationally hard* if a hard computational problem can be simulated by the mechanical system. A similar approach can be used to provide lower bounds on the computational resources required to solve various motion planning tasks that arise in the field of robotics. Typically, a robotic motion planning task is specified by a geometric description of the robot (or collection of robots) to be moved, its initial and final positions, the obstacles it is to avoid, as well as a model for the type of feasible motion and physical laws for the movement. The problem of planning such as robotic motion planning task can be shown to be computationally hard if a hard computational problem can be simulated by the robotic motion-planning task.

II. Introduction to Computational Complexity

Abstract Computing Machine Models.

To gauge the computational power of a family of mechanical computers, we will use a widely known abstract computational model known as the Turing Machine, defined in this section.

The Turing Machine. The *Turing machine* model formulated by Alan Turing [1] was the first complete mathematical model of an abstract computing machine that possessed

universal computing power. The machine model has (i) a finite state transition control for logical control of the machine processing, (ii) a tape with a sequence of storage cells containing symbolic values, and (iii) a tape scanner for reading and writing values to and from the tape cells, which could be made to move (left and right) along the tape cells.

A machine model is *abstract* if the description of the machine transition mechanism or memory mechanism does not provide specification of the mechanical apparatus used to implement them in practice. Since Turing's description did not include any specification of the mechanical mechanism for executing the finite state transitions, it can't be viewed as a concrete mechanical computing machine, but instead is an abstract machine. Still it is valuable computational model, due to it simplicity and very widespread use in computational theory.

A *universal* Turing machine simulates any other Turing machine; it takes its input a pair consisting of a string providing a symbolic description of a Turing machine M and the input string x, and simulates M on input x. Because of its simplicity and elegance, the Turing Machine has come to be the standard computing model used for most theoretical works in computer science. Informally, the Church-Turing hypothesis states that a Turing machine model can simulate a computation by any "reasonable" computational model (we will discuss some other reasonable computational models below).

Computational Problems. A *computational problem* is: given an input string specified by a string over a finite alphabet, determine the Boolean answer: 1 is the answer is YES, and otherwise 0. For simplicity, we generally will restrict the input alphabet to be the binary alphabet {0,1}. The *input size* of a computational problem is the number of input symbols; which is the number of bits of the binary specification of the input. (Note: It is more common to make these definitions in terms of language acceptance. A *language* is a set of strings over a given finite alphabet of symbols. A computational problem can be identified with the language consisting of all strings over the input alphabet where the answer is 1. For simplicity, we defined each complexity class as the corresponding class of problems.)

Recursively Computable Problems and Undecidable Problems. There is a large class of problems, known as *recursively computable* problems, that Turing machines compute in finite computations, that is, always halting in finite time with the answer. There are certain problems that are not recursively computable; these are called *undecidable* problems. The *Halting Problem is*: given a Turing Machine description and an input, output 1 if the Turing machine ever halts, and else output 0. Turing proved the halting problem is undecidable. His proof used a method known as a diagonalization method; it considered an enumeration of all Turing machines and inputs, and showed a contradiction occurs when a universal Turing machine attempts to solve the Halting problem for each Turing machine and each possible input.

Computational Complexity Classes. *Computational complexity* (see [2]) is the amount of computational resources required to solve a given computational problem. A *complexity class* is a family of problems, generally defined in terms of limitations on the

resources of the computational model. The complexity classes of interest here will be associated with restrictions on the time (number of steps until the machine halts) and/or space (the number of tape cells used in the computation) of Turing machines. There are a number of notable complexity classes:

P is the complexity class associated with efficient computations, and is formally defined to be the set of problems solved by Turing machine computations running in time polynomial in the input size (typically, this is the number of bits of the binary specification of the input).

NP is the complexity class associated with combinatorial optimization problems which if solved can be easily determined to have correct solutions, and is formally defined to be the set of problems solved by Turing machine computations using nondeterministic choice running in polynomial time.

PSPACE is the complexity class is defined to be set of problems solved by Turing machines running in space polynomial in the input size.

EXPTIME is the complexity class is defined to be set of problems solved by Turing machine computations running in time exponential in the input size.

NP and **PSPACE** are widely considered to have instances that are not solvable in P, and it has been proved that **EXPTIME** has problems that are not in **P.**

Polynomial Time Reductions. A *polynomial time reduction* from a problem Q' to a problem Q is a polynomial time Turing machine computation that transforms any instance of the problem Q' into an instance of the problem Q which has an answer YES if and only if the problem Q' has an answer YES. Informally, this implies that problem Q can be used to efficiently solve the problem Q'. A problem Q is *hard* for a family F of problems if for every problem Q' in F, there is a polynomial time reduction from Q' to Q. Informally, this implies that problem Q can be used to efficiently solve *any* problem in F. A problem Q is *complete* for a family F of problems if Q is in C and also hard for F.

Hardness Proofs for Mechanical Problems. He will later consider various mechanical problems and characterize their computation power:

- *Undecidable mechanical problems;* typically this was proved by a computable reduction from the halting problem for a universal Turing machine problems to an instance of the mechanical problem; this is equivalent to showing the mechanical problem can be viewed as a computational machine that can simulate a universal Turing machine computation.
- *Mechanical problems that are hard for **NP, PSPACE, or EXPTIME;*** typically this was proved by a polynomial time reduction from the problems in the appropriate complexity class to an instance of the mechanical problem; again, this is equivalent to showing the mechanical problem can be viewed as a computational machine that can simulate a Turing machine computation in the appropriate complexity class.

The simulation proofs in either case often provide insight into the intrinsic computational power of the mechanical problem or mechanical machine.

Other Abstract Computing Machine Models

There are a number of abstract computing models discussed in this Chapter, that are equivalent, or nearly equivalent to conventional deterministic Turing Machines.

- **Reversible Turing Machines.** A computing device is *(logically) reversible* if each transition of its computation can be executed both in forward direction as well in reverse direction, without loss of information. Landauer [3] showed that irreversible computations must generate heat in the computing process, and that reversible computations have the property that if executed slowly enough, can (in the limit) consume no energy in an adiabatic computation. A *reversible Turing machine* model allows the scan head to observe 3 consecutive tape symbols and to execute transitions both in forward as well as in reverse direction. Bennett [4] showed that any computing machine (e.g., an abstract machine such as a Turing Machine) can be transformed to do only reversible computations, which implied that reversible computing devices are capable of universal computation. Bennett's reversibility construction required extra space to store information to insure reversibility, but this extra space can be reduced by increasing the time. Vitanyi [5] give trade-offs between time and space in the resulting reversible machine. Lewis and Papadimitriou [1] showed that reversible Turing machines are equivalent in computational power to conventional Turing machines when the computations are bounded by polynomial time, and Crescenzi and Papadimitriou [6] proved a similar result when the computations are bounded by polynomial space. This implies that the definitions of the complexity classes **P** and **PSPACE** do not depend on the Turing machines being reversible or not. Reversible Turing machines are used in many of the computational complexity proofs to be mentioned involving simulations by mechanical computing machines.

- **Cellular Automata.** These are sets of finite state machines that are typically connected together by a grid network. There are known efficient simulations of Turing machine by cellular automata (e.g., see Wolfram [7] for some known universal simulations). A number of the particle-based mechanical machines to be described are known to simulate cellular automata**.**

- **Randomized Turing machines.** The machine can make random choices in its computation. While the use of randomized choice can be very usefull in many efficient algorithms, there is evidence that randomization only provides limited additional computational power above conventional deterministic Turing machines (In particular, there are a variety of pseudo-random number generation methods proposed for producing long pseudo-random sequences from short truly random seeds, that are which are widely conjectured to be indistinguishable from truly random sequences by polynomial time Turning machines.) A number of the mechanical machines to be described using Brownian-motion have natural sources of random numbers.

There are also a number of abstract computing machine models that appear to be more powerful than conventional deterministic Turing Machines.

- **Real-valued Turing machines.** In these machines due to Blum et al [8], each storage cell or register can store any real value (that may be transcendental). Operations are extended to allow infinite precision arithmetic operations on real numbers. To our knowledge, none of the analog computers that we will describe in this chapter have this power.

- **Quantum Computers.** A *quantum superposition* is a linear superposition of basis states; it is defined by a vector of complex amplitudes whose absolute magnitudes sum to 1. In a quantum computer, the quantum superposition of basis states is transformed in each step by a unitary transformation (this is a linear mapping that is reversible and always preserves the value of the sum of the absolute magnitudes of its inputs). The outputs of a quantum computation are read by observations that that project the quantum superposition to classical values; a given state is chosen with probability defined by the magnitude of the amplitude of that state in the quantum superposition. Feynman [9] and Benioff [10] were the first to suggest the use of quantum mechanical principles for doing computation, and Deutsch [11] was the first to formulate an abstract model for quantum computing and show it was universal. Since then, there is a large body of work in quantum computing (see Gruska [12] and Nielsen [13]) and quantum information theory (see Jaeger [14] and Reif [15]). Some of the particle-based methods for mechanical computing described below make use of quantum phenomena, but generally are not considered to have the full power of quantum computers.

II. The Computational Complexity of Motion Planning and Simulation of Mechanical Devices

Complexity of Motion Planning for Mechanical Devices with Articulated Joints

The first known computational complexity result involving mechanical motion or robotic motion planning was in 1979 by Reif [16]. He consider a class of mechanical systems consisting of a finite set of connected polygons with articulated joints, which are required to be moved between two configurations in three dimensional space avoiding a finite set of fixed polygonal obstacles. To specify the movement problem (as well as the other movement problems described below unless otherwise stated), the object to be moved, as well as its initial and final positions, and the obstacles are all defined by linear inequalities with rational coefficients with a finite number of bits. He showed that this class of motion planning problems is hard for PSPACE. Since it is widely conjectured that PSPACE contains problems are not solvable in polynomial time, this result provided the first evidence that these robotic motion planning problems not solvable in time polynomial in n if number of degrees of freedom grow with n. His proof involved simulating a reversible Turing machine with n tape cells by a mechanical device with n articulated polygonal arms that had to be maneuvered through a set of fixed polygonal obstacles similar to the channels in Swiss-cheese. These obstacles where devised to force the mechanical device to simulate transitions of the reversible Turing machine to be simulated, where the positions of the arms encoded the tape cell contents, and tape read/write operations were simulated by channels of the obstacles which forced the arms to be reconfigured appropriately. This class of movement problems can be solved by

reduction to the problem of finding a path in a O(n) dimensional space avoiding a fixed set of polynomial obstacle surfaces, which can be solved by a PSPACE algorithm due to Canny [17]. Hence this class of movement problems are PSPACE complete. (In the case the object to be moved consists of only one rigid polygon, the problem is known as the piano mover's problem and has a polynomial time solution by Schwartz and Sharir [18].)

Other PSPACE completeness results for mechanical devices.
There were many subsequent PSPACE completeness results for mechanical devices (two of which we mention below), which generally involved *multiple degrees of freedom*:

- **The Warehouseman's Problem.** Schwartz and Sharir [19] showed in 1984 that moving a set of n disconnected polygons in two dimensions from an initial position to a final position among finite set of fixed polygonal obstacles PSPACE hard.

There are two classes of mechanical dynamic systems, the Ballistic machines and the Browning Machines described below, that can be shown to provide simulations of polynomial space Turing machine computations.

Ballistic Collision-based Computing Machines and PSPACE
A *ballistic computer* (see Bennett [20,21]) is a conservative dynamical system that follows a mechanical trajectory isomorphic to the desired computation. It has the following properties:

- Trajectories of distinct ballistic computers can't be merged,
- All operations of a computational must be reversible,
- Computations, when executed at constant velocity, require no consumption of energy,
- Computations must be executed without error, and needs to be isolated from external noise and heat sources.

Collision-based computing [22] is computation by a set of particles, where each particle holds a finite state value, and state transformations are executed at the time of collisions between particles. Since collisions between distinct pairs of particles can be simultaneous, the model allows for parallel computation. In some cases the particles can be configured to execute cellular automata computations [23]. Most proposed methods for *Collision-based computing* are ballistic computers as defined above. Examples of concrete physical systems for collision-based computing are:

- **The Billiard Ball Computers.** Fredkin and Toffoli [24] considered a mechanical computing model, the *billiard ball computer*, consisting spherical billiard balls with polygonal obstacles, where the billiard balls were assume to have perfect elastic collisions with no friction. They showed in 1982 that a Billiard Ball Computer, with an unbounded number of billiard balls, could simulate a reversible computing machine model that used reversible Boolean logical gates known as Toffoli gates. When restricted to finite set of n spherical billiard balls, their construction provides a simulation of a polynomial space reversible Turing machine.

- **Particle-like waves in excitable medium.** Certain classes of excitable medium have discrete models that can exhibit particle-like waves that propagate through the media [25], and using this phenomena, Adamatzky [26] gave a simulation of a universal Turing Machine, and if restricted to n particle-waves, provided a simulation of a polynomial space Turing Machine.
- **Soliton Computers.** A soliton is a wave packet that maintains a self-reinforcing shape as it travels at constant speed through a nonlinear dispersive media. A soliton computer [27,28] makes use of optical solitons to hold state, and state transformations are made by colliding solitons.

Brownian machines and PSPACE

In a mechanical system exhibiting *fully Brownian motion*, the parts move freely and independently, up to the constraints that either link the parts together or forces the parts exert on each other. In a fully Brownian motion, the movement is entirely due to heat and there is no other source of energy driving the movement of the system. An example of a mechanical systems with fully Brownian motion is a set of particles exhibiting Browning motion, as say with electrostatic interaction. The rate of movement of mechanical system with fully Brownian motion is determined entirely by the drift rate in the random walk of their configurations.

Other mechanical systems, known as *driven Brownian motion systems,* exhibit movement is only partly due to heat; in addition there is a driving there is a source of energy driving the movement of the system. Example a of driven Brownian motion systems are:

- Feynman's Ratchet and Pawl [29], which is a mechanical ratchet system that has a driving force but that can operate reversibly.
- Polymerase enzyme, which uses ATP as fuel to drive their average movement forward, but also can operate reversibly.

There is no energy consumed by fully Brownian motion devices, whereas driven Brownian motion devices require power that grows as a quadratic function of the drive rate in which operations are executed (see Bennett [21]).

Bennett [20] provides two examples of Brownian computing machines:

- An *enzymatic machine*. This is a hypothetical biochemical device that simulates a Turing machine, using polymers to store symbolic values in a manner to similar to Turing machine tapes, and uses hypothetical enzymatic reactions to execute state transitions and read/write operations into the polymer memory. Shapiro [30] also describes a mechanical Turing machine whose transitions are executed by hypothetical enzymatic reactions.
- A *clockwork computer*. This is a mechanism with linked articulated joints, with a Swiss-cheese like set of obstacles, which force the device to simulate a Turing machine. In the case where the mechanism of Bennett's clockwork computer is restricted to have a linear number of parts, it can be used to provide a simulation of PSPACE similar that of [16].

Hardness results for mechanical devices with a constant number of degrees of freedom

There were also additional computation complexity hardness results for mechanical devices, which only involved a *constant number of degrees of freedom*. These results exploited special properties of the mechanical systems to do the simulation.

- **Motion planning with moving obstacles.** Reif and Sharir [31] considered the problem of planning the motion of a rigid object (the robot) between two locations, avoiding a set of obstacles, some of which are rotating. They showed this problem is PSPACE hard. This result was perhaps surprising, since the number of degrees of freedom of movement of the object to be moved was constant. However, the simulation used the rotational movement of obstacles to force the robot to be moved only to position that encoded all the tape cells of M. The simulation of a Turing machine M was made by forcing the object between such locations (that encoded the entire n tape cell contents of M) at particular times, and further forced that object to move between these locations over time in a way that simulated state transitions of M.

NP hardness results for path problems in two and three dimensions

Shortest path problems in fixed dimensions involve only a constant number of degrees of freedom. Nevertheless, there are a number of NP hardness results for such problems. These results also led to proofs that certain physical simulations (in particular, simulation of multi-body molecular and celestial simulations) are NP hard, and therefore not likely efficiently computable with high precision.

- **Finding shortest paths in three dimensions.** Consider the problem of finding a shortest path of a point in three dimensions (where distance is measured in the Euclidean metric) avoiding fixed polyhedral obstacles whose coordinates are described by rational numbers with a finite number of bits. This shortest path problem can be solved in PSPACE [17], but the precise complexity of the problem is an open problem. Canny and Reif [32] were the first to provide a hardness complexity result for this problem; they showed the problem is NP hard. Their proof used novel techniques called *free path encoding* that used 2^n homotopy equivalence classes of shortest paths. Using these techniques, they constructed exponentially many shortest path classes (with distinct homotopy) in single-source multiple-destination problems involving O(n) polygonal obstacles. They used each of these path to encode a possible configuration of the nondeterministic Turing machine with n binary storage cells. They also provided a technique for simulating each step of the Turing machine by the use of polygonal obstacles whose edges forced a permutation of these paths that encoded the modified configuration of the Turing machine. These encoding allowed them to prove that the single-source single-destination problem in three dimensions is NP-hard. Similar free path encoding techniques were used for a number of other complexity hardness results for mechanical simulations described below.

- **Kinodynamic planning.** *Kinodynamic planning* is the task of motion planning while subject to simultaneous kinematic and dynamics constraints. The algorithms for various classed of kinodynamic planning problems were first developed in

[33]. Canny and Reif [32] also used Free path encoding techniques to show two dimensional kinodynamic motion planning with bounded velocity is NP-hard.

- **Shortest Curvature-Constrained Path planning in Two Dimensions.** We now consider *curvature-constrained shortest path problems*: which involve finding a shortest path by a point among polygonal obstacles, where the there is an upper bound on the path curvature. A class of curvature-constrained shortest path problems in two dimensions were shown to be NP hard by Reif and Wang [34], by devising a set of obstacles that forced the shortest curvature-constrained path to simulate a given nondeterministic Turing machine

PSPACE Hard Physical Simulation Problems

- **Ray Tracing with a Rational Placement and Geometry.** Ray tracing is given an optical system and the position and direction of an initial light ray, determine if the light ray reaches some given final position. This problem of determining the path of light ray through an optical system was first formulated by Newton in his book on Optics. Ray tracing has been used for designing and analyzing optical systems. It is also used extensively in computer graphics to render scenes with complex curved objects under global illumination. Reif, Tygar, and Yoshida [35] showed the problem of ray tracing in various three dimensional optical, where the optical devices either consist of reflective objects defined by quadratic equations, or refractive objects defined by linear equations, but in either case the coefficients are restricted to be rational. They showed this ray tracing problems are PSPACE hard. Their proof used free path encoding techniques for simulating a nondeterministic linear space Turing machine, where the position of the ray as it enters a reflective or refractive optical object (such as a mirror or prism face) encodes the entire memory of the Turing machine to be simulated, and further steps of the Turing machine are simulated by optically inducing appropriate modifications in the position of the ray as it enters other reflective or refractive optical objects. This result implies that the apparently simple task of highly precise ray tracing through complex optical systems is not likely to be efficiently executed by a polynomial time computer. It is another example of the use of a physical system to do powerful computations.

- **Molecular and gravitational mechanical systems.** A quite surprising example of the use of physical systems to do computation is the work of Tate and Reif [36] on the complexity of n-body simulation, where they showed that the problem is PSPACE hard, and therefore not likely efficiently computable with high precision. In particular, they considered multi-body systems in three dimensions with n particles and inverse polynomial force laws between each pair of particles (e.g., molecular systems with Columbic force laws or celestial simulations with gravitational force laws). It is quite surprising that such systems can be configured to do computation. Their hardness proof made use of free path encoding techniques similar to the proof of PSPACE-hardness of ray tracing. A single particle, which we will call the *memory-encoding particle*, is distinguished. The

position of a memory-encoding particle as it crosses a plane encodes the entire memory of the Turing machine to be simulated, and further steps of the Turing machine are simulated by inducing modifications in the trajectory of the memory-encoding particle. The modifications in the trajectory of the memory-encoding particle are made by use of other particles that have trajectories that induce force fields that essentially act like force-mirrors, causing reflection-like changes in the trajectory of the memory-encoding particle. Hence highly precise n-body molecular simulation is not likely to be efficiently executed by a polynomial time computer.

A Provably Intractable Mechanical Simulation Problem: Compliant motion planning with uncertainty in control.

Next, we consider compliant motion planning with uncertainty in control. Specifically, we consider a point in 3 dimensions which is commanded to move in a straight line, but whose actual motion may differ from the commanded motion, possibly involving sliding against obstacles. Given that the point initially lies in some start region, the problem is to find a sequence of commanded velocities that is guaranteed to move the point to the goal. This problem was shown by Canny and Reif [32] to be non-deterministic EXPTIME hard, making it the first provably intractable problem in robotics. Their proof used free path encoding techniques that exploited the uncertainty of position to encode exponential number of memory bits in a Turing machine simulation.

Undecidable Mechanical Simulation Problems:

- **Motion Planning with Friction.** Consider a class of mechanical systems whose parts consist of a finite number of rigid objects defined by linear or quadratic surface patches connected by frictional contact linkages between the surfaces. (Note: this class of mechanisms is similar to the analytical engine developed by Babbage at described in the next sections, except that there are smooth frictional surfaces rather than toothed gears). Reif and Sun [37] proved that an arbitrary Turing machine could be simulated by a (universal) frictional mechanical system in this class consisting of a finite number of parts. The entire memory of a universal Turing machine was encoded in the rotational position of a rod. In each step, the mechanism used a construct similar to Babbage's machine to execute a state transition. The key idea in their construction is to utilize frictional clamping to allow for setting arbitrary high gear transmission. This allowed the mechanism to execute state transitions for arbitrary number of steps. Simulation of a universal Turing machine implied that the movement problem is undecidable when there are frictional linkages. (A problem is *undecidable* if there is no Turing machine that solves the problem for all inputs in finite time.) It also implied that a mechanical computer could be constructed with only a constant number of parts that has the power of an unconstrained Turing machine.

- **Ray Tracing with Non-Rational Postitioning.** Consider again the problem of ray tracing in a three dimensional optical systems, where the optical devices again may be either consist of reflective objects defined by quadratic equations, or refractive objects defined by linear equations. Reif, et al [35] also proved that in

the case where the coefficients of the defining equations are not restricted to be rational, and include at least one irrational coefficient, then the resulting ray tracing problem could simulate a universal Turing machine, and so is undecidable. This ray tracing problem for reflective objects is equivalent to the problem of tracing the trajectory of a single particle bouncing between quadratic surfaces, which is also undecidable by this same result of [35]. In independent result of Moore [38] also showed that the undecidability of the problem of tracing the trajectory of a single particle bouncing between quadratic surfaces.

- **Dynamics and Nonlinear Mappings.** Moore [39], Ditto [40]and Munakata et al [41] have also given universal Turing machine simulations of various dynamical systems with nonlinear mappings.

IV. Concrete Mechanical Computing Devices

Mechanical computers have a very extensive history; some surveys given in Knott [42], Hartree [43], Engineering Research Associates [44], Chase [45], Martin [46], Davis [47]. Norman [48] recently provided a unique overview of mechanical calculators and other historical computers, summarizing the contributions of notable manuscripts and publications on this topic.

Mechanical Devices for Storage and Sums of Numbers

Mechanical methods, such as notches on stones and bones, knots and piles of pebbles, have been used since the Neolithic period for storing and summing integer values. One example of such a device, the *abacus*, which may have been developed invented in Babylonia approximately 5000 years ago, makes use of beads sliding on cylindrical rods to facilitate addition and subtraction calculations.

Analog Mechanical Computing Devices

Computing devices will considered here to be *analog* (as opposed to digital) if they don't provide a method for restoring calculated values to discrete values, whereas digital devices provide restoration of calculated values to discrete values. (Note that both analog and digital computers uses some kind of physical quantity to represent values that are stored and computed, so the use of physical encoding of computational values is not necessarily the distinguishing characteristic of analog computing.) Descriptions of early analog computers are given by Horsburgh [49], Turck[50], Svoboda [51], Hartree [43], Engineering Research Associates [44] and Soroka [52]. There are a wide variety of mechanical devices used for analog computing:

- **Mechanical Devices for Astronomical and Celestial Calculation.** While we have not sufficient space in this article to fully discuss this rich history, we note that various mechanisms for predicting lunar and solar eclipses using optical illumination of configurations of stones and monoliths (for example, Stonehenge) appear to date to the Neolithic period. Mechanical mechanisms for more precisely predicting lunar and solar eclipses may have been developed in the classical period of ancient history. The most impressive and sophisticated known example of an ancient gear-based mechanical device is the Antikythera Mechanism, which is thought to have been constructed by Greeks in approximately 2200 years ago.

Recent research [53] provides evidence it may have been used to predict celestial events such as lunar and solar eclipses by the analog calculation of arithmetic-progression cycles. Like many other intellectual heritages, some elements of the design of such sophisticated gear-based mechanical devices may have been preserved by the Arabs after that period, and then transmitted to the Europeans in the middle ages.

- **Planimeters.** There is a considerable history of mechanical devices that integrate curves. A *planimeter* is a mechanical device that integrates the area of the region enclosed by a two dimensional closed curve, where the curve is presented as a function of the angle from some fixed interior point within the region. One of the first known planimeters was developed by J.A. Hermann in 1814 and improved (as the polar planimeter) by J.A. Hermann in 1856. This led to a wide variety of mechanical integrators known as wheel-and-disk integrators, whose input is the angular rotation of a rotating disk and whose output, provided by a tracking wheel, is the integral of a given function of that angle of rotation. More general mechanical integrators known as ball-and-disk integrators, who's input provided 2 degrees of freedom (the phase and amplitude of a complex function), were developed by James Thomson in 1886. There are also devices, such as the Integraph of Abdank Abakanoviez(1878) and C.V. Boys(1882), which integrate a one-variable real function of x presented as a curve y=f(x) on the Cartesian plane. Mechanical integrators were later widely used in WWI and WWII military analog computers for solution of ballistics equations, artillery calculations and target tracking. Various other integrators are described in Morin [54].

- **Harmonic Analyzers.** A *Harmonic Analyzer* is a mechanical device that calculates the coefficients of the Fourier Transform of a complex function of time such as a sound wave. Early harmonic analyzers were developed by Thomson [55] and Henrici [56] using multiple pulleys and spheres, known as ball-and-disk integrators.

- **Harmonic Synthesizers.** A *Harmonic Synthesizer* is a mechanical device that interpolates a function given the Fourier coefficients. Thomson (then known as Lord Kelvin) in 1886 developed [57] the first known Harmonic Analyzer that used an array of James Thomson's (his brother) ball-and-disk integrators. Kelvin's Harmonic Synthesizer made use of these Fourier coefficients to reverse this process and interpolate function values, by using a wire wrapped over the wheels of the array to form a weighted sum of their angular rotations. Kelvin demonstrated the use of these analog devices predict the tide heights of a port: first his Harmonic Analyzer calculated the amplitude and phase of the Fourier harmonics of solar and lunar tidal movements, and then his Harmonic Synthesizer formed their weighted sum, to predict the tide heights over time. Many other Harmonic Analyzers were later developed, including one by Michelson and Stratton (1898) that performed Fourier analysis, using an array of springs. Miller [58] gives a survey of these early Harmonic Analyzers. Fisher [59] made improvements to the tide predictor and later Doodson and Légé increase the scale of this design to a 42-wheel version that was used up to the early 1960s.

- **Analog Equation Solvers.** There are various mechanical devices for calculating the solution of sets of equations. Kelvin also developed one of the first known mechanical mechanisms for equation solving, involving the motion of pulleys and tilting plate that solved sets of simultaneous linear equations specified by the physical parameters of the ropes and plates. John Wilbur in the 1930s increased the scale of Kelvin's design to solve nine simultaneous linear algebraic equations. Leonardo Torres Quevedo constructed various rotational mechanical devices, for determining real and complex roots of a polynomial. Svoboda [51] describes the state of art in the 1940s of mechanical analog computing devices using linkages.

- **Differential Analyzers.** A *Differential Analyzer* is a mechanical analog device using linkages for solving ordinary differential equations. Vannevar Bush [60] developed in 1931 the first Differential Analyzer at MIT that used a torque amplifier to link multiple mechanical integrators. Although it was considered a general-purpose mechanical analog computer, this device required a physical reconfiguration of the mechanical connections to specify a given mechanical problem to be solved. In subsequent Differential Analyzers, the reconfiguration of the mechanical connections was made automatic by resetting electronic relay connections. In addition to the military applications already mentioned above, analog mechanical computers incorporating differential analyzers have been widely used for flight simulations and for industrial control systems.

- **Mechanical Simulations of Physical Processes: Crystallization and Packing.** There are a variety of macroscopic devices used for simulations of physical processes, which can be viewed as analog devices. For example, a number of approaches have been used for mechanical simulations of crystallization and packing:

    - ***Simulation using solid macroscopic ellipsoids bodies.*** Simulations of kinetic crystallization processes have been made by collections of macroscopic solid ellipsoidal objects – typically of diameter of a few millimeters - which model the molecules comprising the crystal. In these physical simulations, thermal energy is modeled by introducing vibrations; low level of vibration is used to model freezing and increasing the level of vibrations models melting. In simple cases, the molecule of interest is a sphere, and ball bearings or similar objects are used for the molecular simulation. For example, to simulate the dense random packing of hard spheres within a crystalline solid, Bernal [61] and Finney [62] used up to 4000 ball bearings on a vibrating table. In addition, to model more general ellipsoidal molecules, orzo pasta grains as well as M&M candies (Jerry Gollub at Princeton University) have been used. Also, Cheerios have been used to simulate the liquid state packing of benzene molecules. To model more complex systems mixtures of balls of different sizes and/or composition have been used; for example a model ionic crystal formation has been made by use a mixture of balls composed of different materials

that acquired opposing electrostatic charges.

- ***Simulations using bubble rafts*** [63,64]**.** These are the structures that assemble among equal sized bubbles floating on water. They typically they form two dimensional hexagonal arrays, and can be used for modeling the formation of close packed crystals. Defects and dislocations can also be modeled [65]; for example by deliberately introducing defects in the bubble rats, they have been used to simulate crystal dislocations, vacancies, and grain boundaries. Also, impurities in crystals (both interstitial and substitutional) have been simulated by introducing bubbles of other sizes.

- **Reaction-Diffusion Chemical Computers**. Adamatzky [66,67] described a class of analog computers that where there is a chemical medium which has multiple chemical species, where the concentrations of these chemical species vary spatially and which diffuse and react in parallel. The memory values (as well as inputs and outputs) of the computer are encoded by the concentrations of these chemical species at a number of distinct locations (also known as micro-volumes). The computational operations are executed by chemical reactions whose reagents are these chemical species. Example computations [66,67] include: (i) Voronoi diagram; this is to determine the boundaries of the regions closest to a set of points on the plane, (ii) Skeleton of planar shape, and (iii) a wide variety of two dimensional patterns periodic and aperiodic in time and space.

Digital Mechanical Devices for Arithmetic Operations

Recall that we have distinguished digital mechanical devices from the analog mechanical devices described above by their use of mechanical mechanisms for insuring the values stored and computed are discrete. Such discretization mechanisms include geometry and structure (e.g., the notches of Napier's bones described below), or cogs and spokes of wheeled calculators. Surveys of the history of some these digital mechanical calculators are given by Knott [42], Turck [50], Hartree [43], Engineering Research Associates [44], Chase [45], Martin [46], Davis [47], and Norman [48].

- **Leonardo da Vinci's Mechanical Device and Mechanical Counting Devices.** This intriguing device, which involved a sequence of interacting wheels positioned on a rod, which appear to provide a mechanism for digital carry operations, was illustrated in 1493 in Leonardo da Vinci's Codex Madrid [68]. A working model of its possible mechanics was constructed in 1968 by Joseph Mirabella. Its function and purpose is not decisively known, but it may have been intended for counting rotations (e.g., for measuring the distance traversed by a cart). There are a variety of apparently similar mechanical devices used to measuring distances traversed by vehicles.
- **Napier's Bones.** John Napier [69] developed in 1614 a mechanical device known as Napier's Bones allowed multiplication and division (as well as square and cube roots) to be done by addition and multiplication operations. It consisting of rectilinear rods, which provided a mechanical transformation to and from logarithmic values. Wilhelm Shickard developed in 1623 a six digit mechanical

calculator that combined the use of Napier's Bones using columns of sliding rods, with the use of wheels used to sum up the partial products for multiplication.

- **Slide Rules.** Edmund Gunter devised in 1620 a method for calculation that used a single log scale with dividers along a linear scale; this anticipated key elements of the first slide rule described by William Oughtred [70] in 1632. A very large variety of slide machines were later constructed.

- **Pascaline: Pascal's Wheeled Calculator.** Blaise Pascal [71] developed in 1642 a calculator known as the Pascaline that could calculate all four arithmetic operations (addition, subtraction, multiplication, and division) on up to eight digits. A wide variety of mechanical devices were then developed that used revolving drums or wheels (cogwheels or pinwheels) to do various arithmetical calculations.

- **Stepped Drum Calculators.** Gottfried Wilhelm von Leibniz developed in 1671 an improved calculator known as the Stepped Reckoner, which used a cylinder known as a *stepped drum* with nine teeth of different lengths that increase in equal amounts around the drum. The stepped drum mechanism allowed use of moving slide for specifying a number to be input to the machine, and made use of the revolving drums to do the arithmetic calculations. Charles Xavier Thomas de Colbrar developed in 1820 a widely used arithmetic mechanical calculator based on the stepped drum known as the Arithmometer. Other stepped drum calculating devices included Otto Shweiger's Millionaire calculator (1893) and Curt Herzstark's Curta (early 1940s).

- **Pinwheel Calculators.** Another class of calculators, independently invented by Frank S. Baldwin and W. T. Odhner in the 1870s, is known as pinwheel calculators; they used a pinwheel for specifying a number input to the machine and use revolving wheels to do the arithmetic calculations. Pinwheel calculators were widely used up to the 1950s, for example in William S. Burroughs's calculator/printer and the German Brunsviga.

Digital Mechanical Devices for Mathematical Tables and Functions

- **Babbage's Difference Engine.** Charles Babbage [72,73] in 1820 invented a mechanical device known as the *Difference Engine* for calculation of tables of an analytical function (such as the logarithm) that summed the change in values of the function when a small difference is made in the argument. That difference calculation required for each table entry involved a small number of simple arithmetic computations. The device made use of columns of cogwheels to store digits of numerical values. Babbage planned to store 1000 variables, each with 50 digits, where each digit was stored by a unique cogwheel. It used cogwheels in registers for the required arithmetical calculations, and also made use of a rod-based control mechanism specialized for control of these arithmetic calculations. The design and operation of the mechanisms of the device were described by a symbolic scheme developed by Babbage [74]. He also conceived of a printing

mechanism for the device. In 1801, Joseph-Marie Jacquard invented an automatic loom that made use of punched cards for the specification of fabric patterns woven by his loom, and Charles Babbage proposed the use of similar punched cards for providing inputs to his machines. He demonstrated over a number of years certain key portions of the mechanics of the device but never completed a complete function device.

- **Other Difference Engines.** In 1832 Ludgate [75] independently designed, but did not construct, a mechanical computing machine similar but smaller in scale to Babbage's Analytical Engine. In 1853 Pehr and Edvard Scheutz [76] constructed in Sweden a cog wheel mechanical calculating device (similar to the Difference Engine originally conceived by Babbage) known as the Tabulating Machine, for computing and printing out tables of mathematical functions. This (and a later construction of Babbage's Difference Engine by Doron Swade [77] of the London Science Museum) demonstrated the feasibility of Babbage's Difference Engine.

- **Babbage's Analytical Engine.** Babbage further conceived (but did not attempt to construct) a mechanical computer known as the Analytical Engine to solve more general mathematical problems. Lovelace's extended description of Babbage's Analytical Engine [78] (translation of "Sketch of the Analytical Engine" by L. F. Menabrea) describes, in addition to arithmetic operations, also mechanisms for looping and memory addressing. However, the existing descriptions of Babbage's Analytical Engine appear to lack the ability to execute a full repertory of logical and/or finite state transition operations required for general computation. Babbage's background was very strong in analytic mathematics, but he (and the architects of similar cog-wheel based mechanical computing devices at that date) seemed to have lacked knowledge of sequential logic and its Boolean logical basis, required for controlling the sequence of complex computations. This (and his propensity for changing designs prior to the completion of the machine construction) might have been the real reason for the lack of complete development of a universal mechanical digital computing device in the early 1800's.

- **Subsequent Electromechanical Digital Computing Devices with Cog-wheels.** Other electromechanical digital computing devices (see [44]) developed in the late 1940s and 1950s, that contain cog-wheels, included Howard Aiken's Mark 1 [79] constructed at Harvard University and Konrad Zuse's Z series computer constructed in Germany.

Mechanical Devices for Timing, Sequencing and Logical Control

We will use the term *mechanical automata* here to denote mechanical devices that exhibit autonomous control of their movements. These can require sophisticated mechanical mechanisms for timing, sequencing and logical control.

- **Mechanisms used for Timing Control.** Mechanical clocks, and other mechanical device for measuring time have a very long history, and include a very wide variety of designs, including the flow of liquids (e.g., water clocks), or

sands (e.g., sand clocks), and more conventional pendulum-and-gear based clock mechanisms. A wide variety of mechanical automata and other control devices make use of mechanical timing mechanisms to control the order and duration of events automatically executed (for example, mechanical slot machines dating up to the 1970s made use of such mechanical clock mechanisms to control the sequence of operations used for payout of winnings). As a consequence, there is an interwoven history in the development of mechanical devices for measuring time and the development of devices for the control of mechanical automata.

- **Logical Control of Computations.** A critical step in the history of computing machines was the development in the middle 1800's of Boolean logic by George Boole [80,81]. Boole innovation was to assign values to logical propositions: 1 for true propositions and 0 for false propositions. He introduced the use of Boolean variables which are assigned these values, as well the use of Boolean connectives (and and or) for expressing symbolic Booelan logic formulas. Boole's symbolic logic is the basis for the logical control used in modern computers. Shannon [82] was the first to make use of Boole's symbolic logic to analyze relay circuits (these relays were used to control an analog computer, namely MIts Differential Equalizer).

- **The Jevons' Logic Piano: A Mechanical Logical Inference Machine.** In 1870 William Stanley Jevons (who also significantly contributed to the development of symbolic logic) constructed a mechanical device [83,84] for the inference of logical proposition that used a piano keyboard for inputs. This *mechanical inference machine* is less widely known than it should be, since it may have had impact in the subsequent development of logical control mechanisms for machines.

- **Mechanical Logical Devices used to Play Games.** Mechanical computing devices have also been constructed for executing the logical operations for playing games. For example, in 1975, a group of MIT undergraduates including Danny Hillis and Brian Silverman constructed a computing machine made of Tinkertoys that plays a perfect game of tic-tac-toe.

Mechanical Devices used in Cryptography

- **Mechanical Cipher Devices Using Cogwheels.** Mechanical computing devices that used cogwheels were also developed for a wide variety of other purposes beyond merely arithmetic. A wide variety of mechanical computing devices were developed for the encryption and decryption of secret messages. Some of these (most notably the family of German electromechanical cipher devices known as *Enigma Machines* [85] developed in the early 1920s for commercial use and refined in the late 1920s and 1930s for military use) made use of sets of cogwheels to permute the symbols of text message streams. Similar (but somewhat more advanced) electromechanical cipher devices were used by the USSR up to the 1970s.

- **Electromechanical Computing Devices used in Breaking Cyphers.** In 1934 Marian Rejewski and a team including Alan Turing constructed an electrical/mechanical computing device known as the Bomb, which had an architecture similar to the abstract Turing machine described below, and which was used to decrypt ciphers made by the German Enigma cipher device mentioned above.

Mechanical and Electro-Optical Devices for Integer Factorization

- **Lehmer's number sieve computer.** In 1926 Derrick Lehmer [86] constructed a mechanical device called the number sieve computer for various mathematical problems in number theory including factorization of small integers and solution of Diophantine equations. The device made use of multiple bicycle chains that rotated at distinct periods to discover solutions (such as integer factors) to these number theoretic problems.

- **Shamir's TWINKLE.** Adi Shamir [87,88,89] proposed a design for a optical/electric device known as TWINKLE for factoring integers, with the goal of breaking the RSA public key cryptosystem. This was unique among mechanical computing devices in that it used time durations between optical pulses to encode possible solution values. In particular, LEDs were made to flash at certain intervals of time (where each LED is assigned a distinct period and delay) at a very high clock rate so as to execute a sieve-based integer factoring algorithm.

Mechanical Computation at the Micro Scale: MEMS Computing Devices. Mechanical computers can have advantages over electronic computation at certain scales; they are already having widespread use at the microscale. MEMS (Micro-Electro-Mechanical Systems) are manufactured by lithographic etching methods similar in nature to the processes microelectronics are manufactured, and have a similar microscale. A wide variety of MEMS devices [90] have been constructed for sensors and actuators, including accelerometers used in automobile safety devices and disk readers, and many of these MEMS devices execute mechanical computation do their task. Perhaps the MEMS device most similar in architecture to the mechanical calculators described above is the Recodable Locking Device [91] constructed in 1998 at Sandia Labs, which made use of microscopic gears that acted as a mechanical lock, and which was intended for mechanically locking strategic weapons.

VI. Future Directions

Mechanical Self-Assembly Processes.

Most of the mechanical devices discussed in this chapter have been assumed to be constructed top-down; that is they are designed and then assembled by other mechanisms generally of large scale. However a future direction to consider are bottom-up processes

for assembly and control of devices. Self-assembly is a basic bottom-up process found in many natural processes and in particular in all living systems.

- **Domino Tiling Problems.** The theoretical basis for self-assembly has its roots in Domino Tiling Problems (also known as Wang tilings) as defined by Wang [92] (Also see the comprehensive text of Grunbaum, et al, [93]). The input is a finite set of unit size square tiles, each of whose sides are labeled with symbols over a finite alphabet. Additional restrictions may include the initial placement of a subset of these tiles, and the dimensions of the region where tiles must be placed. Assuming an arbitrarily large supply of each tile, the problem is to place the tiles, without rotation (a criterion that cannot apply to physical tiles), to completely fill the given region so that each pair of abutting tiles have identical symbols on their contacting sides.
- **Turing-universal and NP Complete Self-assemblies.** Domino tiling problems over an infinite domain with only a constant number of tiles were first proved by [94] to be undecidable. Lewis and Papadimitriou [95] showed the problem of tiling a given finite region is NP complete.
- **Theoretical Models of Tiling Self-assembly Processes.** Domino tiling problems do not presume or require a specific process for tiling. Winfree [96] proposed kinetic models for self-assembly processes. The sides of the tiles are assumed to have some methodology for selective affinity, which we call pads. Pads function as programmable binding domains, which hold together the tiles. Each pair of pads have specified binding strengths (a real number on the range [0,1] where 0 denotes no binding and 1 denotes perfect binding). The self-assembly process is initiated by a singleton tile (the seed tile) and proceeds by tiles binding together at their pads to form aggregates known as tiling assemblies. The preferential matching of tile pads facilitates the further assembly into tiling assemblies.
- **Pad binding mechanisms.** These provide a mechanism for the preferential matching of tile sides can be provided by various methods:
    - *magnetic attraction*, e.g., pads with magnetic orientations (these can be constructed by curing ferrite materials (e.g., PDMS polymer/ferrite composites) in the presence of strong magnet fields)the and also pads with patterned strips of magnetic orientations,
    - *capillary force*, using hydrophobic/hydrophilic (capillary) effects at surface boundaries that generate lateral forces,
    - *shape matching* (also known as *shape complementarity* or *conformational affinity*), using the shape of the tile sides to hold them together.
    - (Also see the sections below *discussion of the used of molecular affinity* for pad binding.)
- **Materials for Tiles.** There are a variety of distinct materials for tiles, at a variety of scales: Whitesides (see [97] and http://www-chem.harvard.edu/GeorgeWhitesides.html) has developed and tested multiple technologies for meso-scale self-assembly, using capillary forces, shape complementarity, and magnetic forces). Rothemund [98] gave some of the most complex known meso-scale tiling assemblies using polymer tiles on fluid boundaries with pads that use hydrophobic/hydrophilic forces. A materials

science group at the U. of Wisconsin (http://mrsec.wisc.edu/edetc/selfassembly) has also tested meso-scale self-assembly using magnetic tiles

- **Meso-Scale Tile Assemblies.** Meso-Scale Tiling Assemblies have tiles of size a few millimeters up to a few centimeters. They have been experimentally demonstrated by a number of methods, such as placement of tiles on a liquid surface interface (e.g., at the interface between two liquids of distinct density or on the surface of an air/liquid interface), and using mechanical agitation with shakers to provide a heat source for the assembly kinetics (that is, a temperature setting is made by fixing the rate and intensity of shaker agitation).
- **Applications of Meso-scale Assemblies.** The are a number of applications, including:
    - Simulation of the thermodynamics and kinetics of molecular-scale self-assemblies.
    - For placement of a variety of microelectronics and MEMS parts.

Mechanical Computation at the Molecular Scale: DNA Computing Devices. Due to the difficulty of constructing electrical circuits at the molecular scale, alternative methods for computation, and in particular mechanical methods, may provide unique opportunities for computing at the molecular scale. In particular the bottom-up self-assembly processes described above have unique applications at the molecular scale.

- **Self-assembled DNA nanostructures.** Molecular-scale structures known as *DNA nanostructures* (see surveys by Seeman [99] and Reif [100]) can be made to self-assemble from individual synthetic strands of DNA. When added to a test tube with the appropriate buffer solution, and the test tube is cooled, the strands self-assemble into DNA nanostructures. This self-assembly of DNA nanostrucures can be viewed as a mechanical process, and in fact can be used to do computation. The first known example of a computation by using DNA was by Adleman [101,102] in 1994; he used the self-assembly of DNA strands to solve a small instance of a combinatorial optimization problem known as the Hamiltonian path problem.
- **DNA tiling assemblies.** The Wang tiling [92] paradigm for self-assembly was the basis for scalable and programmable approach proposed by Winfree et al [103] for doing molecular computation using DNA. First a number of distinct DNA nanostructures known as DNA tiles are self-assembled. End portions of the tiles, known as pads, are designed to allow the tiles to bind together a programmable manner similar to Wang tiling, but in this case uses the molecular affinity for pad binding due to hydrogen bonding of complementary DNA bases. This programmable control of the binding together of DNA tiles provides a capability for doing computation at the molecular scale. When the temperature of the test tube containing these tiles is further lowered, the DNA tiles bind together to form complex patterned tiling lattices that correspond to computations.
- **Assembling Patterned DNA tiling assemblies.** Programmed patterning at the molecular scale can be produced by the use of strands of DNA that encode the patterns; this was first done by Yan, et al [104] in the form of bar-cord striped patterns, and more recently Rothemund [105] who self-assembled complex 2D

molecular patterns and shapes. Another method for molecular patterning of DNA tiles is via computation done during the assembly.

- **Computational DNA tiling assemblies.** The first experimental demonstration of computation via the self-assembly of DNA tiles was in 2000 by Mao et al [106], and Yan et al [107], which provided a 1 dimensional computation of a binary-carry computation (known as prefix-sum) associated with binary adders. Rothemund et al [108] in 2004 demonstrated a 2 dimensional computational assemblies of tiles displaying a pattern known as the Sierpinski triangle, which is the modulo 2 version of Pascal's triangle.
  Other autonomous DNA devices. DNA nanostructures can also be made to make sequences of movement, and a demonstration of an autonomous moving DNA robotic device, that moved without outside mediation across a DNA nanostructures was given by Yin et al [109]. The design of an autonomous DNA device that moves under programmed control is described in [110]. Surveys of DNA autonomous devices are given in [111] and [112].

Acknowledgements
We sincerely thank Charles Bennett for his numerous suggests and very important improvements to this survey.

IX. Bibliography

[1] Alan Turing, On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, Second Series, Vol. 42. London, pp. 230-265, (1937). Erratum in Vol. 43, pp. 544-546, (1937).

[2] Harry R. Lewis and Christos H. Papadimitriou, Elements of the Theory of Computation, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1997.

[3] R. Landauer, "Irreversibility and heat generation in the computing process", IBM J. Res. Dev. 5, 183 (1961).

[4] C. H. Bennett, "Logical reversibility of computation," IBM Journal of Research and Development, vol. 17, no. 6, pp. 525-532, 1973.

[5] M. Li, P. Vitanyi, Reversibility and Adiabatic Computation: Trading Time and Space for Energy, (Online preprint quant-ph/9703022), Proc. Royal Society of London, Series A, 452(1996), 769-789.

[6] Pierluigi Crescenzi and Christos H. Papadimitriou, “Reversible simulation of space-bounded computations,” Theoretical Computer Science, Vol. 143 , Issue 1, pp 159 - 165 (May 1995).

[7] Wolfram S. Universality and complexity in cellular automata. Physica D10 (1984) 1-35.

[8] Lenore Blum, Felipe Cucker, Mike Shub, Steve Smale, Complexity and Real Computation: A Manifesto, Int. J. of Bifurcation and Chaos, Vol 6, No. 1, World Scientific, Singapore, pp 3-26, (1996).

[9] R. P. Feynman. Simulating physics with computers. International Journal of Theoretical Physics, 21(6/7): pp. 467-488, (1982).

[10] Benioff, P. Quantum mechanical models of Turing machines that dissipate no energy. Phys. Rev. Lett. 48, 1581, (1982).

[11] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. Proceedings of the Royal Society, London, A400:97-117, (1985).

[12] Jozef Gruska, Quantum Computing. McGraw-Hill, New York, NY (1999).

[13] Michael Nielsen and Isaac Chuang, Quantum Computation and Quantum Information. Cambridge: Cambridge University Press, (2000).

[14] Gregg Jaeger, Quantum Information: An Overview. Berlin: Springer, (2006).

[15] J.H. Reif, Quantum Information Processing: Algorithms, Technologies and Challenges, invited chapter in Nano-scale and Bio-inspired Computing, (edited by M. M. Eshaghian-Wilner), John Wiley Inc, Hoboken, NJ, (Feb. 2009).

[16] J.H. Reif, Complexity of the Mover's Problem and Generalizations. 20th Annual IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico, October 1979, pp. 421-427. Also appearing in Chapter 11 in Planning, Geometry and Complexity of Robot Motion, Jacob Schwartz, ed., Ablex Pub. Norwood, NJ, 1987, pp. 267-281.

[17] John Canny. Some algebraic and geometric computations in PSPACE. In (Richard Cole, editor), Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, pages 460-467, Chicago, IL, May 1988. ACM Press.

[18] Jacob T. Schwartz and M. Sharir. On the piano movers problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. Comm. Pure Appl. Math., 36:345-398, 1983.

[19] [HS84] J.E. Hopcroft, J.T. Schwartz, and M. Sharir, "On the Complexity of Motion Planning for Multiple Independent Objects: PSPACE Hardness of the Warehouseman's Problem," International J. Robotics Research, Vol. 3, No. 4, pp. 76-88, (1984).

[20] Charles H. Bennett, "The thermodynamics of computation--a review." Intl. J. Theoretical Physics 21(12):905-940, 1982. http://www.research.ibm.com/people/b/bennetc/bennettc1982666c3d53.pdf

[21] Charles H. Bennett, "Notes on Landauer's principle, reversible computation, and Maxwell's Demon", Studies in History and Philosophy of Modern Physics vol. 34 pp. 501-510 (2003). eprint physics/0210005: http://xxx.lanl.gov/abs/physics/0210005

[22] Andrew Adamatzky (Ed.), Collision-based computing, Springer-Verlag London, UK, (2001).

[23] Squier R. and Steiglitz K. Programmable parallel arithmetic in cellular automata using a particle model. Complex Systems8 (1994) 311-323.

[24] Edward Fredkin and Tommaso Toffoli, "Conservative logic", Int. J. Theor. Phys., Vol. 21, pp 219-253, (1982).

[25] Adamatzky A.I. On the particle-like waves in the discrete model of excitable medium. Neural Network World1 (1996) pp 3-10.

[26] Adamatzky A.I. Universal dynamical computation in multidimensional excitable lattices. Int. J. Theor. Phys.37 (1998) pp 3069-3108.

[27] Jakubowski M.H., Steiglitz K., and Squier R. State transformations of colliding optical solitons and possible application to computation in bulk media. Physical Review E58 (1998) 6752-6758.

[28] Mariusz H. Jakubowski, Ken Steiglitz, Richard Squier, Computing with solitons: a review and prospectus, Collision-based computing, Springer-Verlag London, UK, pp 277 - 297, (2001).

[29] Richard P. Feynman, "Ratchet and Pawl," The Feynman Lectures on Physics, Vol. 1, Chapter 46, edited by R.P. Feynman, R.B. Leighton, and M. Sands, Addison-Wesley, Reading, Mass, (1963).

[30] [S00] Ehud Shapiro, "A Mechanical Turing Machine: Blueprint for a Biomolecular Computer", Fifth International Meeting on DNA-Based Computers at the Massachusetts Institute of Technology, Proc. DNA Based Computers V: Cambridge, MA, June 14-16, 1999.

[31] John H. Reif and M. Sharir, Motion Planning in the Presence of Moving Obstacles, 26th Annual IEEE Symposium on Foundations of Computer Science, Portland, OR, October 1985, pp. 144-154. Published in Journal of the ACM (JACM), 41:4, July 1994, pp. 764-790.

[32] J. Canny and J.H. Reif, New Lower Bound Techniques for Robot Motion Planning Problems. 28th Annual IEEE Symposium on Foundations of Computer Science, Los Angeles, CA, October 1987, pp. 49-60.

[33] J. Canny, B. Donald, J.H. Reif and P. Xavier. On the Complexity of Kinodynamic Planning. 29th Annual IEEE Symposium on Foundations of Computer Science, White Plains, NY, October 1988, pp. 306-316. Published as Kinodynamic Motion Planning, Journal of the ACM, Vol 40(5), November 1993, pp. 1048-1066.

[34] J.H. Reif and H. Wang, The Complexity of the Two Dimensional Curvature-Constrained Shortest-Path Problem, Third International Workshop on Algorithmic Foundations of Robotics (WAFR98), Pub. by A. K. Peters Ltd, Houston, Texas, pages 49-57, June, 1998.

[35] J.H. Reif, D. Tygar, and A. Yoshida, The Computability and Complexity of Optical Beam Tracing. 31st Annual IEEE Symposium on Foundations of Computer Science, St. Louis, MO, October 1990, pp. 106-114. Published as

The Computability and Complexity of Ray Tracing in Discrete & Computational Geometry, 11: pp 265-287 (1994).

[36] S.R. Tate and J.H. Reif, The Complexity of N-body Simulation, Proceedings of the 20th Annual Colloquium on Automata, Languages and Programming (ICALP'93), Lund, Sweden, July, 1993, p. 162-176.

[37] J.H. Reif and Z. Sun, The Computational Power of Frictional Mechanical Systems, Third International Workshop on Algorithmic Foundations of Robotics, (WAFR98), Pub. by A. K. Peters Ltd, Houston, Texas, pages 223-236, Mar. 5-7 1998. Published as On Frictional Mechanical Systems and Their Computational Power, SIAM Journal of Computing(SICOMP), Vol. 32, No. 6, pp. 1449-1474, (2003).

[38] Christopher Moore, Undecidability and Unpredictability in Dynamical Systems. Physical Review Letters 64 (1990) 2354-2357.

[39] Christopher Moore, Generalized Shifts: Undecidability and Unpredictability in Dynamical Systems. Nonlinearity 4 (1991) 199-230.

[40] Toshinori Munakata, Sudeshna Sinha, and William L. Ditto, Chaos Computing: Implementation of Fundamental Logical Gates by Chaotic Elements, IEEE Transactions on Circuits adn Systems-I Fundamental Theory and Applications Vol. 49, No. 11, pp 1629-1633 (Nov 2002).

[41] Sinha S, Ditto, Computing with distributed chaos. Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics. 60(1):363-77. (1999).

[42] Cargill Gilston Knott, editor. Napier tercentenary memorial volume. London: Published for the Royal Society of Edinburgh by Longmans, Green, 1915.

[43] Douglas R. Hartree, Calculating Instruments and Machines, Cambridge University Press, London, UK (1950).

[44] Engineering Research Associates Staff, High-Speed Computing Devices, McGraw-Hill Book Co., New York City, NY 1950.

[45] George C. Chase, "History of Mechanical Computing Machinery." IEEE Annals of the History of Computing, Volume 2, No. 3, pp. 198-226, July 1980.

[46] Ernst Martin, "The Calculating Machines." The MIT Press, Cambridge, Massachusetts, 1992.

[47] Martin Davis, The Universal Computer: The Road from Leibniz to Turing, Norton Press, Norton, VA, 2000.

[48] Jeremy M. Norman (Editor), The Origins of Cyberspace: From Gutenberg to the Internet: a sourcebook on the history of information technology, Norman Publishing, Novato, CA, 2002.

[49] E. M. Horsburgh, Modern Instruments of Calculation, G. Bell & Sons, London (1914), p. 223.

[50] J.A.V. Turck, "Origin of Modern Calculating Machines." The Western Society of Engineers, Chicago, 1921.

[51] Antonin Svoboda, Computing Mechanisms and Linkages, McGraw-Hill, New York, NY, 1948.

[52] Walter A. Soroka, Analog Methods in Computation and Simulation, McGraw-Hill Co., New York, NY (1954).

[53] T. Freeth, Y. Bitsakis X. Moussas, J. H. Seiradakis, A. Tselikas, H. Mangou, M. Zafeiropoulou, R. Hadland, D. Bate, A. Ramsey, M. Allen, A. Crawley, P. Hockley, T. Malzbender, D. Gelb, W. Ambrisco and M. G. Edmunds, Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism, Nature, Nature 444, 587-591 (30 November 2006).

[54] H. de Morin, Les appareils d'intégration: intégrateurs simples et composés; planimètres; intégromètres; intégraphes et courbes intégrales; analyse harmonique et analyseurs, Gauthier-Villars Publishers, Paris, (1913).

[55] William Thomson (later known as Lord Kelvin), Harmonic Analyzer. Proceedings of the Royal Society of London, Vol. 27, 1878, pp. 371-373.

[56] Henrici, Philosophical Magazine, 38, 110 (1894).

[57] Lord Kelvin, Harmonic analyzer and synthesizer. Proc. Royal Society, 27, 371 (1878).

[58] Dayton Miller, The Henrici Harmonic Analyzer and Devices for Extending and Facilitating Its Use. Journal of the Franklin Institute, Vol. 181, pp. 51-81 and Vol. 182, pp. 285-322 (Sept 1916).

[59] E.G. Fisher, Tide-predicting machine, Engineering News, 66, 69-73 (1911).

[60] Vannevar Bush, "The differential analyzer: A new machine for solving differential equations," Journal of the Franklin Institute, 212: 447, 1931.

[61] J. D. Bernal, "The Structure of Liquids", Proc. Roy. Soc. London, Ser. A 280, 299 (1964).

[62] J. L. Finney, Random Packings and the Structure of Simple Liquids. I. The Geometry of Random Close Packing, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, Vol. 319, No. 1539 (Nov. 10, 1970), pp. 479-493.

[63] L. Bragg and J.F.Nye, , "A dynamical model of a crystal structure,"*Proc. R. Soc.* A,, **190**, pp 474 –481 (1947).

[64] L. Bragg andW.M. Lomer, "A dynamical model of a crystal structure II," *Proc. R. Soc.* A,, 196, 171–181 (1948).

[65] Corcoran, S. G.,Colton, R. J.,Lilleodden, E. T., and Gerberich, W. W., *Phys. Rev.* B, 190, pp 474 (1997).

[66] Andrew Adamatzky, Benjamin De Lacy Costello, and Tetsuya Asai, Reaction-Diffusion Computers, Elsevier, London, UK (2005). [B64]

[67] Adamatzky A.I. Chemical processor for computation of voronoi diagram. Advanced Materials for Optics and Electronics, Vol. 6, No. 4, pp 191-196 (Dec. 1998).

[68] Leonardo da Vinci, Codex Madrid I, 1493.

[69] John Napier, Mirifici logarithmorum canonis descriptio (the description of the. wonderful canon of logarithms), Published by Hart, Edinburgh, UK, 1614.

[70] William Oughtred, Circles of Proportion and the Horizontal Instrument, Translated and Published by William Forster, London, 1632.

[71] Etienne Pascal, Lettre dédicatoire à Monseigneur le Chancelier sur le sujet de la machine nouvellement inventée par le sieur B. P pour faire toutes sortes d'opérations d'arithmétique par un mouvement réglé sans plume ni jetons, suivie d'un avis nécessaire à ceux qui auront curiosité de voir ladite machine et de s'en servir. (1645).

[72] Charles Babbage, On Machinery for Calculating and Printing Mathematical Tables, The Edinburgh Philosophical Journal, Edited by Robert Jameson and David Brewster, Edinburgh, Archibald Constable and Company, Vol. VII, pp. 274-281, August 1, 1822.

[73] Charles Babbage, Observations on the application of machinery to the computation of mathematical tables, The Philosophical Magazine and Journal, Vol. LX, pp. 311-314, London: Richard Taylor, 1825.

[74] Charles Babbage, On a Method of expressing by Signs the Action of Machinery, Philosophical Transactions of the Royal Society of London 116, Part III, pp. 250-265. (1826).

[75] Percy Ludgate, On a proposed analytical engine, Scientific Proceedings of the Royal Dublin Society, 12, 77-91 (1909-10).

[76] Michael Lindgren, Glory and Failure: Difference Engines of Johann Muller, Charles Babbage and Georg and Edvard Scheutz, MIT Press, Cambridge, MA, 1990.

[77] Doron Swade, Charles Babbage and His Calculating Engines, Michigan State University Press, East Lansing, MI, 1991.

[78] Ada Lovelace, translation of "Sketch of the Analytical Engine" by L. F. Menabrea with Ada's notes and extensive commentary. Ada Lovelace, 'Sketch of the analytical engine invented by Charles Babbage', Esq. Scientific Memoirs 3 (1843), pp. 666-731.

[79] [CW99] I. Bernard Cohen, Gregory W. Welch, Makin' Numbers: Howard Aiken and the Computer, MIT Press, Cambridge, MA, (1999).

[80] George Boole, Mathematical Analysis of Logic, pamphlet, 1847.

[81] George Boole, An Investigation of the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities, Published by Macmillan and Co, London (1854).

[82] Claude Shannon "A Symbolic Analysis of Relay and Switching Circuits," Transactions of the American Institute of Electrical Engineers, vol. 57 (1938), pp. 713-719.

[83] William Stanley Jevons, "On the Mechanical Performance of Logical Inference." Philosophical Transactions of the Royal Society, Vol. 160, Part II, pp. 497-518, 1870.

[84] Stanley W. Jevons; The Principles of Science; A Treatise on Logic and Scientific Method . London and New York, Macmillan and Co. 1873.

[85] David Hamer, Geoff Sullivan, Frode Weierud, Enigma Variations: an Extended Family of Machines; Cryptologia 22(3), pp. 211-229, July 1998.

[86] Derrick H. Lehmer, The mechanical combination of linear forms, The American Mathematical Monthly, vol. 35, 114–121, 1928

[87] Adi Shamir, Method and apparatus for factoring large numbers with optoelectronic devices, patent 475920, filed 12/30/1999 and awarded

08/05/2003.

[88] Adi Shamir, Factoring large numbers with the TWINKLE device, Cryptographic Hardware and Embedded Systems (CHES) 1999, LNCS 1717, 2-12, Springer-Verlag, Heidelberg, Germany, 1999.

[89] Arjen K. Lenstra, Adi Shamir, Analysis and optimization of the TWINKLE factoring Device, proc. Eurocrypt 2000, LNCS 1807, 35--52, Springer-Verlag, 2000.

[90] Marc J. Madou, Fundamentals of Microfabrication: The Science of Miniaturization, , Second Edition, CRC Publishers, Boca Raton, FL (2002).

[91] David Plummer, Larry J. Dalton, Frank Peter, The recodable locking device, Communications of the ACM, Volume 42 , Issue 7 (July 1999) pp 83 – 87.

[92] H. Wang, Dominoes and the AEA Case of the Decision Problem. pp. 23-55 in Mathematical Theory of Automata, J. Fox, ed. Brooklyn, N.Y.: Polytechnic Press, (1963).

[93] Grunbaum, S., Branko, and G.C. Shepard, Tilings and Patterns, Chapter 11, H Freeman and Co, San Francisco, CA (1987).

[94] Berger, R. The Undecidability of the Domino Problem, Memoirs of the American Mathematical Society, 66, pp. 1-72 (1966).

[95] Lewis, H.R., and C.H. Papadimitriou, Elements of the Theory of Computation, Prentice-Hall, Upper Saddle River, NJ, pages 296-300 and 345-348, (1981).

[96] Erik Winfree and Xiaoping Yang and Nadrian C. Seeman, Universal computation via self-assembly of DNA: Some theory and experiments, DNA Based Computers II, Volume 44 of DIMACS, American Mathematical Society, Providence, RI pp. 191-213 (1996).

[97] Xia, Y. and Whitesides, G. M., Soft Lithography, Annu. Rev. Mater. Sci. 1998,28, 153-184.

[98] Rothemund, P.W.K., Using lateral capillary forces to compute by self-assembly, Proc. Nat. Acad. Sci. (USA) 97, 984-989 (2000).

[99] Nadrian C. Seeman, Nanotechnology and the Double Helix; Scientific American, 290 (6), 64-75 (June 2004).

[100] John H. Reif and Thomas H. LaBean, Nanostructures and Autonomous Devices Assembled from DNA, invited chapter in Nano-scale and Bio-

inspired Computing, (edited by M. M. Eshaghian-Wilner), John Wiley Inc, Hoboken, NJ, (Feb. 2009).

[101] Leonard M. Adleman, Molecular computation of solutions to combinatorial problems, Science, v.266 n.11, p.1021-1024, (Nov. 1994).

[102] Leonard Adleman, Computing with DNA, Scientific American, 279(2), p 34-41, (August 1998).

[103] Winfree, E., Liu, F., Wenzler, L.A., and Seeman, N.C. (1998). "Design and Self-Assembly of Two-Dimensional DNA Crystals, Nature 394, 539-544.

[104] Hao Yan, Thomas H. LaBean, Liping Feng, and John H. Reif, Directed Nucleation Assembly of Barcode Patterned DNA Lattices, PNAS, Volume 100, No. 14, pp. 8103-8108, July 8, (2003).

[105] Paul W. K. Rothemund, Folding DNA to create nanoscale shapes and patterns, Nature 440, 297-302 (16 March 2006).

[106] C. Mao, LaBean, T.H. Reif, J.H., Seeman, Logical Computation Using Algorithmic Self-Assembly of DNA Triple-Crossover Molecules, Nature, vol. 407, pp. 493–495. (Sept. 28 2000).

[107] Hao Yan, Liping Feng, Thomas H. LaBean, and John Reif, DNA Nanotubes, Parallel Molecular Computations of Pairwise Exclusive-Or (XOR) Using DNA "String Tile" Self-Assembly in Journal of American Chemistry Society(JACS), Vol. 125, No. 47, pp. 14246-14247, 2003.

[108] Paul W.K. Rothemund, Nick Papadakis, Erik Winfree, Algorithmic Self-Assembly of DNA Sierpinski Triangles, PLoS Biology 2 (12): electronic pub. e424 doi:10.1371/journal.pbio.0020424, (Dec., 2004).

[109] Peng Yin, Hao Yan, Xiaoju G. Daniel, Andrew J. Turberfield, John H. Reif, A Unidirectional DNA Walker Moving Autonomously Along a Linear Track, Angewandte Chemie [International Edition], Volume 43, Number 37, pp 4906-4911, (Sept. 20, 2004).

[110] [RS07] John H. Reif and Sudheer Sahu, Autonomous Programmable DNA Nanorobotic Devices Using DNAzymes, John H. Reif and Sudheer Sahu, Autonomous Programmable DNA Nanorobotic Devices Using DNAzymes, 13th International Meeting on DNA Computing (DNA 13), edited by Max Garzon and Hao Yan), Springer-Verlag Lecture Notes for Computer Science (LNCS), Springer, Berlin Heidelberg, (2008). To Appear in Special Journal Issue on Self-Assembly, Theoretical Computer Science (TCS), (2008).

[111] John H. Reif and Thomas H. LaBean, Autonomous Programmable Biomolecular Devices Using Self-Assembled DNA Nanostructures, Communications of the ACM (CACM), Special Section entitled "New Computing Paradigms (edited by Toshinori Munakata), 2007. Extended version: http://www.cs.duke.edu/~reif/paper/AutonomousDNA/AutonomousDNA.pdf

[112] [BY07] Jonathan Bath and Andrew J. Turberfield, DNA nanomachines, Nature Nanotechnology 2, pp 275 - 284, (2007).