# Physical Manifestation of NP-Completeness in Analog Computing Devices

by

Frank Lee

S.B., Massachusetts Institute of Technology (1998)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1999



Author .....
Department of Electrical Engineering and Computer Science
February 3, 1999

Certified by .....
Thomas F. Knight, Jr.
Senior Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Theses

# Physical Manifestation of NP-Completeness in Analog Computing Devices

by
Frank Lee

Submitted to the Department of Electrical Engineering and Computer Science
on February 3, 1999, in partial fulfillment of the Requirements for the degree of
Master of Engineering

## Abstract

In 1986, Vergis, Steiglitz, and Dickinson proposed a mechanical analog computer to solve NP-hard problems where the amount of computing time needed could theoretically be made as small as possible if the machine works as designed. They proceeded to argue using Strong Church's Thesis and the hypothesis of $P \neq NP$ that the machine must malfunction when operated, although no detailed physical analysis was done to determine what would be the exact cause of failure. This thesis analyzes the machine at an implementation-independent block-diagram level and points out two fundamental flaws in the design.

Thesis Supervisor: Thomas F. Knight, Jr.
Title: Senior Research Scientist

# Acknowledgments

First and foremost, I thank my advisor Tom Knight for supporting me through this effort, and for providing me with my first research experiences via the MIT Undergraduate Research Opportunities Program. One of these experiences led to this thesis. I've learned a great deal from my conversations with him. He has a way of getting to the heart of the matter very quickly, and places great emphasis on clear, succinct explanation.

I am eternally grateful to my mom and dad for their love and support, and for the efforts they made to provide a peaceful environment where I could get a college education, do research, and contemplate. They were also my first source of funding, providing me with the means to procure the electronic components, tools, books, and software needed to build my first electronics projects.

Special thanks go to my brother Arthur, MIT Class of 1985. He has always been generous in sharing with me his years of wisdom and experience. Throughout my childhood, he helped kindle my interests in mathematics and the physical sciences. He introduced me to algebra, coordinate geometry, trigonometry, and calculus, well before these topics were covered in my schoolwork. He bought me an electronics project kit from Radio Shack when he first saw my interest in taking apart radios. I'll always be thankful for his foresight in working afterschool, weekends, and summers to save up money to buy an Apple ][+ personal computer in 1979. That machine provided me with my first computer experiences while I was still in kindergarten, and my first computer programming experiences when I entered second grade.

I'd like to acknowledge certain people who provided additional support and guidance during various crucial moments of my career at the Institute. Special thanks go to Jim Bales at the Edgerton Center, Jacky Mallett at the Media Lab, George Verghese of EECS, and Anne Hunter at the Course VI Undergraduate Office.

Last but not least, I thank my friends for making things fun throughout my school years. In particular I should mention (in alphabetical order): Mike Jacknis, Peter Leung, James Lin, and Janis Stipins. I'd also like to thank the people on the 6th floor of Ashdown House for making the dorm such a cool place to return to after a long day.

This research was supported by DARPA (the Defense Advanced Research Projects Agency of the United States of America) as part of the Scalable Computing Systems research program, under contract number DABT63-95-C-0130.

*to our curiosity*

# Contents

# Chapter 1

# Introduction & background

## 1.1 The Goal of the Thesis

In 1986, Vergis, Steiglitz, and Dickinson proposed a mechanical analog computer to solve NP-complete problems where the amount of computing time needed could theoretically be made as small as possible if the machine works as designed [Vergis86]. They proceeded to argue using Strong Church's Thesis and the hypothesis of P $\neq$ NP that the machine must malfunction when operated, although no detailed physical analysis was done to determine what would be the exact cause of failure. The goal of my thesis is to analyze the physical operation of the machine and determine using physical arguments why it does not work.

## 1.2 Background

In 1986, Anastasios Vergis, Kenneth Steiglitz, and Bradley Dickinson proposed a class of mechanical devices that can be used to solve 3-SAT, a well-known NP-complete problem. The authors then used widely held principles and beliefs in computer science to argue that these machines could never work, because if they did, it would be possible to solve NP-complete problems within polynomial time, using only polynomial quantities of machine parts and a good machine shop. This would throw into doubt either the belief that P $\neq$ NP, or the widely-held strengthened interpretation of the Church-Turing Thesis — that the P and NP class hierarchy is invariant over all physically-realizable computing machines. This has been referred to in the literature as "Strong Church's Thesis" [Vergis86].

This debunking of the proposed machine is in the same spirit as the instinctive, 5-second dismissal of proposed perpetual motion machines by invoking the laws of thermodynamics. But there is a difference. The principles of thermodynamics have proven themselves over more than a century of intense developments in engineering, physics, chemistry, and biology. The P $\neq$ NP statement, on the other hand, only expresses a belief of computer scientists who

have worked but failed for decades to either show that P = NP or find a polynomial-time algorithm for an NP-complete problem [Garey79]. It has not been proven that P $\neq$ NP. Also the Strong Church's Thesis has not been tested exhaustively enough to be held at the same level of confidence and broad applicability as thermodynamic laws such as conservation of energy. For instance, recent work in quantum computing suggests the P and NP classes for quantum computers may differ from those of classical ones [Shor97]. We will, however, restrict our discussion to classical computers, and hold the Strong Church's Thesis to say that the P and NP class hierarchy is invariant over all *classical* computing machines.

## 1.3 The argument for machine failure

Assume the 3-SAT machine *does work* and solves 3-SAT using polynomial resources (time, energy, force). By Strong Church's Thesis, any digital computer can simulate any analog one by expending a quantity of computing effort that is at most a polynomial function of the resources used by the analog device. A digital computer running a simulation of the 3-SAT machine would thus be able to solve 3-SAT within polynomial time. But this contradicts the hypothesis that P $\neq$ NP. Therefore the assumption that the 3-SAT machine works must be false — *the 3-SAT machine does not work.*

## 1.4 Motivation for the present work

The line of reasoning in the Vergis paper raises issues intersecting physics and computer science. The authors were able to use the major theoretical ideas in computer science of Church's Thesis and P $\neq$ NP to say something about how a physical machine should or should not work, without the need to study the detailed mechanism of the machine to see why. This could inspire, among other things, future attempts at disproving machines through the argument "If that machine really worked, it would mean P = NP, which is dubious, thus it is unlikely that machine really works."

There may be a fundamental physical principle at work. The principles of thermodynamics were founded on the experimental fact that no successful perpetual motion machines have ever been built. Analogously, there may be physical principles, discovered or undiscovered, that correspond to the notions of Church's Thesis and whether or not P is a proper subset of NP.

This thesis takes the first step into this line of inquiry by providing a detailed analysis of the operation of the Vergis-Steiglitz-Dickinson 3-SAT machine (henceforth referred to as the "3-SAT machine") with the goal of isolating the exact failure mechanism.

## 1.5 Vergis' Algorithm to convert a 3-SAT instance to a machine

Start with the instance of 3-SAT to be solved. For example:

*Given boolean variables $X_1, X_2, X_3, X_4$. Does there exist an assignment of truth values to these variables that will cause all the clauses listed below to be true?*

$$\begin{array}{l} X_1 + \overline{X}_2 + X_4 \\ X_2 + \overline{X}_3 + \overline{X}_4 \\ \overline{X}_1 + X_2 + \overline{X}_3 \\ \overline{X}_2 + X_3 + \overline{X}_4 \end{array}$$

The following procedure is applied to the 3-SAT instance to create a machine:

1. Define real variables $x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3, x_4, \overline{x}_4$, i.e., one real variable for each unnegated boolean one, and one real variable for each negated boolean variable. Also define a real variable $x_0$.

2. Convert all the boolean clauses in the 3-SAT instance to inequalities by changing all boolean variables to their corresponding real variables, and by reinterpreting all + signs to mean addition instead of logical-or. Append the symbols $\geq x_0$ to the right of each converted clause to form inequalities.

3. The result is:

$$\begin{array}{l} x_1 + \overline{x}_2 + x_4 \geq x_0 \\ x_2 + \overline{x}_3 + \overline{x}_4 \geq x_0 \\ \overline{x}_1 + x_2 + \overline{x}_3 \geq x_0 \\ \overline{x}_2 + x_3 + \overline{x}_4 \geq x_0 \end{array}$$

   We will call these inequalities *Real 3-SAT clauses.*

4. The relationship enforced between each $x_i$ and $\overline{x}_i$ is

$$x_i^2 - \mathrm{F}\left(\frac{5x + 3\overline{x}_i}{4}\right) \geq 0,$$

   where $F$ is defined as

$$\mathrm{F}(w) \begin{cases} = w^2 & \text{if } w \geq 0 \\ \leq 0 & \text{if } w < 0 \\ \textit{differentiable} & \text{for all } w \end{cases}$$

5. Build a machine out of gears, shafts, squarers, cams (for $F$), differentials, and limit stops that implements these equations as kinematic constraints. An example schematic diagram for such a machine is given below. Initialize all the $x_i$'s and $\overline{x}_i$'s to zero.

Torque the $x_0$ shaft in the positive direction. If the shaft moves, the 3-SAT instance is satisfiable, else it is not.

## 1.6 3-SAT Machine Design Rationale

### 1.6.1 Some terminology

The 3-SAT machine has many moving parts — the exact number of moving parts is proportional to the sum of the number of clauses and the number of variables in the 3-SAT problem from which it was derived. Any motion of the machine will involve the coordinated motion of many of these parts. It is important that we establish a common terminology for discussing these complex motions.

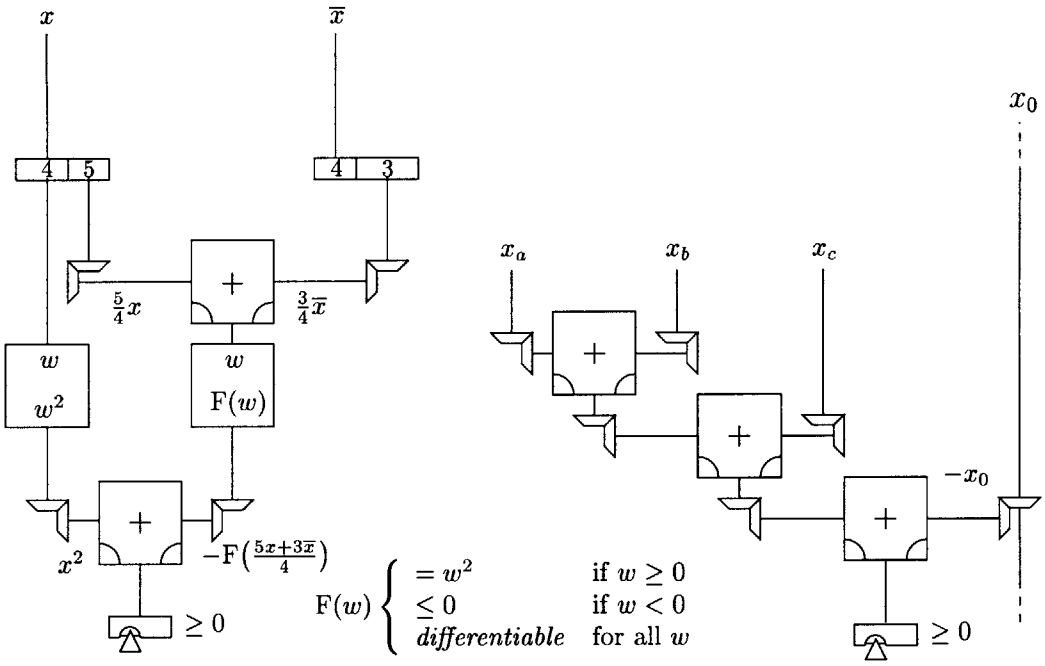
First, we need a way of describing the machine's current position or state. The machine's current state can be captured by listing the angular displacements of all of the machine's shafts. If there are $N$ shafts, then we need a list of $N$ numbers. In general, the positions of $N$ shafts lie in an $N$-dimensional space $\mathrm{R^N}$. We call this the *state space* of the machine.

A machine motion can be described as a line or curve or path through state space. Each coordinate axis of the space corresponds to a unique shaft in the machine. Given a machine motion that is described as a path through state space, the movement of any particular shaft of the machine is the projection of the path onto that shaft's axis in state space.

Often, shaft motions are not independent of each other. Some shafts are so rigidly coupled to others that knowing the displacement of one of them is always enough to indicate the displacements of the others. For example, knowing the angular displacement of just one of the gears in a gear train is enough to allow one to determine the angular displacements of all the gears in the train, if the axles of the gears are all fixed with respect to one another. For any machine made of gears, shafts, differentials, cams, and limit stops all mounted on the same rigid frame, there is a minimum number of shafts whose displacements one must know to be able to figure out the displacements of all shafts in the machine. This number is called the number of *degrees of freedom* of the machine.

The numbers $x_0$, $x_i$, and $\overline{x}_i$, $i \in \{1, 2, 3, \ldots, n\}$ in the equations generated by Vergis' Algorithm each represent shaft displacements. $n$ is the number of variables in the 3-SAT instance that mapped to the machine. While there may be more than $2n + 1$ shafts in any practical implementation of the machine, these shafts taken together are the fewest necessary to represent all the machine's degrees of freedom. This makes it possible to capture the configuration of the machine with a state space of dimension $2n + 1$.

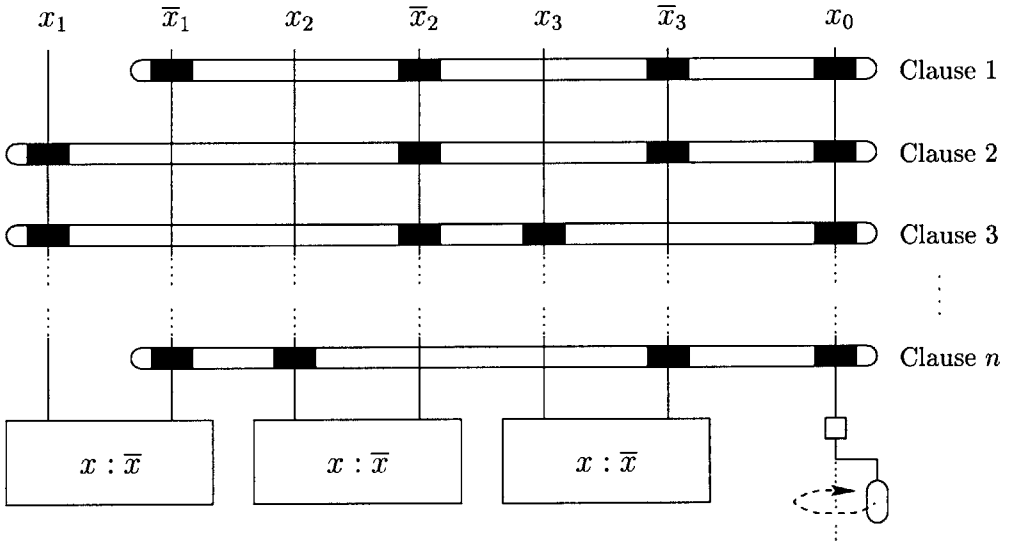
The equations generated by Vergis' Algorithm tell us which machine configurations are allowed and disallowed. Only those points in state space that satisfy the kinematic constraints generated by the algorithm correspond to allowed configurations of the machine. The set of all allowed configurations is called the *feasible space* of the machine.




*The $x : \overline{x}$ mechanism* *The Clause mechanism*

*Directional convention: Viewed from the top or from the left, clockwise rotation is (+).*

---




*Example schematic layout of the 3-SAT machine*

For example, consider a simple machine: the differential gear. It is a device with 3 shafts, which we'll label $x$, $y$, and $z$. The differential gear enforces the relationship

$$z = x + y.$$

Some examples of legal configurations are $(x, y, z) = (0, 0, 0)$, $(1, 0, 1)$, $(-1, 1, 0)$, $(2, 3, 5)$, and $(-3, 1, -2)$. An example of an illegal configuration is $(x, y, z) = (3, 2, 1)$, because $3 + 2 \neq 1$.

### 1.6.2 The 3-SAT instance shapes the machine's feasible space

The goal of any 3-SAT problem is to determine whether a set of Boolean clauses can be satisfied simultaneously. Vergis' Algorithm translates 3-SAT problems into equations defining the feasible space of a machine, where the shape of the feasible space has a particular property if and only if the 3-SAT problem is satisfiable.

The particular property is whether or not the origin of the feasible space (*i.e.*, all $x_i$'s and $\overline{x}_i$'s set to zero) is a local maximum for $x_0$. Equivalently, when starting from the origin, is it possible to move to any point where $x_0 > 0$ such that, along the way, $x_0$ is strictly increasing? A proof in Vergis' paper which we will recount below establishes the following correspondence: If the instance of 3-SAT is satisfiable, then the origin is not a local maximum for $x_0$. Otherwise, the origin is a local maximum for $x_0$.

The shape of any feasible space can be explored mathematically: we can guess points in the state space and check them against the equations to see if they satisfy. With enough guesses, we can map out the shape of the space, learn whether the origin maximizes $x_0$ and thus discover whether the corresponding 3-SAT instance is satisfiable or not. Since 3-SAT is NP-complete, this guess-and-try method will take exponential time.

### 1.6.3 Physical exploration of a mathematical space

The idea of the 3-SAT machine is to embody these equations physically in the workings of a machine. Each part of the machine acts to enforce part of the constraints imposed by the equations. The hope is that physical exploration of the space will be faster than any guess-and-try approach. It doesn't look like the machine has to move very much to compute and report its result. The machine is exploring a *local* property of its starting point. The parts of the machine are reasonably simple. It looks like the machine could work, and yet we know that it can't.

### 1.6.4 Intended functioning of the 3-SAT machine

As described in Vergis' Algorithm, to operate the machine, all shafts are initially set to zero. This corresponds to the machine starting at the origin of its feasible space. Then the $x_0$ shaft is torqued in the positive direction. This $x_0$ shaft will either move or not.

**If the 3-SAT instance that mapped to the machine is satisfiable, then the machine can move.** This is because there is at least one path through feasible space that starts from the origin and progresses monotonically to arbitrarily large positive values of $x_0$. This path can be constructed from knowledge of a satisfying assignment $A$ to the 3-SAT problem, using this procedure:

*For each $i \in \{1, \ldots, n\}$ where $n$ is the number of 3-SAT variables: If $X_i$ = TRUE in $A$ then set $x_i = 3x_0$ and $\overline{x}_i = -x_0$, otherwise ($\overline{X}_i$ = TRUE, so) set $x_i = -x_0$ and $\overline{x}_i = 3x_0$.*

Then for all $x_0 > 0$ this is a valid path. A satisfying assignment ensures each 3-SAT clause has at least one variable that's TRUE, so correspondingly, the left-hand side of each Real 3-SAT clause is guaranteed to have at least one term equal to $3x_0$. Since the other 2 terms can sum to no less than $-2x_0$, it is clear that the left-hand side of each Real 3-SAT clause equals or exceeds $x_0$, satisfying the inequality.

To complete the proof that the path is feasible, we must check the relationships between each $x_i$ and its $\overline{x}_i$. First, notice that enforcing

$$x_i^2 - \mathrm{F}\left(\frac{5x_i + 3\overline{x}_i}{4}\right) \geq 0$$

is equivalent to enforcing

$$\overline{x}_i \leq \begin{cases} -\frac{1}{3}x_i & \text{for } x_i \geq 0 \\ -3x_i & \text{for } x_i < 0 \end{cases}.$$

Each pair $(x_i, \overline{x}_i)$ is either assigned the values $(3x_0, -x_0)$ or the values $(-x_0, 3x_0)$. In either case the relationships hold, hence the path is feasible, and hence the machine can move when the $x_0$ shaft is driven.

**If the machine moves, then the 3-SAT instance is satisfiable.** If the machine moves, then $x_0$ must have moved off the origin onto a positive value. Since all left-hand sides of the Real 3-SAT clauses must equal or exceed $x_0$, each left-hand side must be positive. This means at least one term on each left-hand side must be positive, since the sum of 3 negative numbers cannot possibly make a positive number. Examine the machine to find out which terms are positive and which are negative. For each $x_i$ that is positive set the corresponding 3-SAT $X_i$ = TRUE, and for each $\overline{x}_i$ that is positive set $X_i$ = FALSE. The relationship enforced between $x_i$ and $\overline{x}_i$ ensure that both can't be positive, preventing the absurdity of setting $X_i = \overline{X}_i$.

**Net result: The machine moves if and only if the 3-SAT instance is satisfiable.**

# Chapter 2

# Analysis of the machine

## 2.1 The difficulty in analyzing the 3-SAT machine

### 2.1.1 Digital computing machines are easy to simulate and analyze

The step-by-step time-evolution of conventional digital and analog machines is easy to analyze. Conventional digital machines such as personal computers, workstations, calculators, and embedded controllers are all essentially networks of logic gates (such as NANDs, NORs, and inverters) and memory elements (such as latches, flip-flops, and SRAM/DRAM cells). Each logic gate and memory element has clear and well-defined inputs and outputs. Upon a change in input, any change in output occurs within a fixed finite time. All possible inputs and their corresponding outputs are listed in truth tables and timing diagrams.

In principle, we could figure out exactly what any digital computer will do by writing down the state of all of its memory elements at a given instant. Then we figure out what is at the inputs of every logic gate. We start with those gates whose inputs are connected to the outputs of memory elements. The rest is a straightforward process of table-lookups since the truth-tables of all of these gates are known. Eventually, we figure out what is at the inputs of the memory elements, and from that, we know what the state of these memory elements will become next. The process can be repeated to reveal the following state, *ad infinitum*.

### 2.1.2 Digital computing elements have well-defined inputs and outputs

The logic gates in conventional digital computers have well-defined input and output ports. The op-amps and integrators in electronic analog computers also have well-defined input and output ports. Signals at the input ports of these gates affect and determine what signals

are produced at the output ports. Signals at the output ports, however, don't cause these devices to produce signals at their input ports.

### 2.1.3 Whereas the elements of the 3-SAT machine are bidirectional

The shafts attached to the gear couplings, differentials, and smooth cams in a 3-SAT machine don't share this property. No port on any of these components can be considered strictly an input or strictly an output. It is possible to drive a gear coupling either way: A 2:1 gear coupling, driven one way, can be used to output half the drive speed. Driven the other way, it outputs twice the drive speed. In either case, the gear coupling acts to maintain a 2:1 relationship in shaft speed and rotational displacement. The components in a 3-SAT machine thus act to *enforce constraints* or maintain mathematical relationships among the rotational displacements of the machine's shafts.

### 2.1.4 They enforce constraints, but how?

Since the elements of the 3-SAT machine don't have clearly-defined inputs and outputs, it isn't obvious or likely we can simulate the machine's time-evolution with the same degree of ease and certainty as with digital machines. All we know about these elements is which mathematical constraints they enforce among their shafts. The machine design specifies *what* each element must accomplish, but not *how* it should go about doing it.

To begin to figure out the exact trajectory of a 3-SAT machine, we need to solve its set of kinematic equations to map out the feasible space. Certain types of equations, *e.g.* sets of linear equations, are quite easy to solve. Equations involving nonlinear terms are much harder to solve, often requiring search, iterative, and approximation techniques whose computing time cannot be fixed or even predicted in advance. The 3-SAT machine's kinematic equations fall into the nonlinear category due to the inclusion of the squarers.

But even with a map of the feasible space, we still know nothing about the trajectory of the machine's motion. Knowing how each element goes about enforcing its constraint would enable the formulation of differential equations which can be integrated to find the trajectory. Knowing only what each element enforces can only reveal to us the set of all possible trajectories.

### 2.1.5 Only the shape of the machine's state space is specified

A further difficulty: The 3-SAT machine is specified as a set of equations over the reals which define the feasible state space of the machine. A given feasible space can be implemented countless ways — mechanically, electrically [Main94], electromechanically, and someday maybe even chemically or biologically. Any real-world system that can host a multidimensional state space over the reals can potentially be used to build 3-SAT machines. Yet the machine's dynamics will vary with choice of implementation technology.

## 2.2 Our focus

To render the problem tractable, we will narrow our focus to a particular implementation of the 3-SAT machines — the one suggested in Vergis' paper: a mechanical implementation using gears, shafts, differential gears, smooth cams to make squarers, and limit stops. Even so, the problem is still underspecified. To calculate the exact trajectory of the machine through state space requires knowledge of the masses of the machine parts, coefficients of friction, and details of the construction of the cams and differentials.

A balance must be struck between specifying the machine precisely enough to permit analysis and drowning in a flood of details. This thesis will focus on what we can learn about the motion of the machine given only a block-diagram description that shows only the kinematic constraints enforced by each block. We give up trying to compute trajectories in favor of looking at implementation-independent behaviors. We will assume we know nothing about the masses and moments of inertia of the parts, or the coefficients of friction of the machine parts.

Our goal: Given this simplified, underspecified machine model, what can we say about the motion of the 3-SAT machines?

## 2.3 Two fundamental flaws in the 3-SAT machine's design

### 2.3.1 Squarers can jam

The squarers and the F-function cams are the only nonlinear devices in the machine. Since the F-function cam acts the same as a squarer for all nonnegative inputs, much of what we say about the squarers will apply to them.

The squarer is a two-shaft device which constrains the rotational displacement of one of its shafts to be the square of the displacement of the other.

$$y = x^2$$

It works bidirectionally, meaning it is also possible to drive the $y$-shaft and transmit torque to the $x$-shaft.

When the 3-SAT machine is initialized, all shafts are set at zero. At this point, the squarer is no longer bidirectional, since if the $y$-shaft is turned, the differential amount that the $x$-shaft is supposed to turn is

$$\left.\frac{dx}{dy}\right|_{x,y=0} = \left.\left(\frac{dy}{dx}\right)^{-1}\right|_{x,y=0} = \left.\left(\frac{d}{dx}x^2\right)^{-1}\right|_{x,y=0} = \left.\frac{1}{2x}\right|_{x,y=0} = \infty.$$

This means the mechanical advantage against the turning is infinitely great — the squarer's shafts will not turn at all. This jam cannot be overcome by applying more force. It is very much like trying to push a solid block squarely against a flat frictionless wall. The applied force is completely countered by the normal force of the wall. (The more usual case of finite mechanical advantage is analogous to pushing a solid block against a wall at a non-normal angle — the force applied will have a component that is parallel to the wall, and that component will serve to move the object.)

This is a good candidate for a failure mechanism because it occurs regardless of the properties of the materials used to make the machine. It is an inherent property of any machine that implements the squaring function. Since the 3-SAT machine depends on its components functioning in a bidirectional way to enforce constraints, the fact that its squarers aren't bidirectional when it starts out casts doubt on the machine's functioning.

### 2.3.2 The amazing properties of perfectly rigid shafts

Another point of failure is the machine's dependence on perfectly rigid components. Such components, which don't exist in the physical world, have some amazing properties. If one end of a perfectly rigid shaft is torqued, that torque is conveyed *instantaneously* to the the other end. The entire shaft turns in unison. The shaft motion can be considered a signal — it can convey energy and information to a receiver. Here, the signals propagate at infinite speed. This is a violation of Einstein's theory of special relativity, which sets an upper bound on the speed of signal propagation in our universe. Indeed, any physical material has an upper bound for the propagation of mechanical disturbances. It is the speed of sound in that material.

The 3-SAT machine can be thought of as a group of mechanisms that communicate via shaft motions. In zero time, the shafts pool together information from all mechanisms about the constraints on each variable. This impossibly tight cooperation, if only it could exist physically, may indeed allow the machine to work.

The design of the 3-SAT machine implicitly assumes perfectly rigid parts because each shaft's angular displacement is described as a single number in the kinematic constraints generated by Vergis' Algorithm. A more physically realistic shaft would act like a transmission line for torsional waves. Its angular displacement cannot be captured by a single number. Instead, angular displacement must be described as a function of time and position along the shaft. Each infinitesimal segment of the shaft can have its own angular displacement with some amount of independence. The material constraints on these angular displacement functions can probably be represented as (partial differential) wave equations. In any case, the state space of the machine will be of much higher dimensionality than our simple block-diagram model would suggest.

Without instantaneous communication, it is unclear just how long it takes the 3-SAT machine to work. It likely that when the machine is started, a cacophony of torsional vibrations will result, rippling throughout the machine as the various clause mechanisms and variable management mechanisms assert their presence asynchronously, fighting to enforce

their constraints. As torsional disturbances richochet through the device, partially-formed trial solutions to the 3-SAT instance are being tested and rejected by mechanism after mechanism. We can speculate that during this computational process, an observer might hear loud creaking sounds akin to wrenching metal.

## 2.4 Conclusion

The 3-SAT machine will not work for two reasons. One is its reliance on infinite-bandwidth communications between its parts, something which cannot exist under the known laws of our universe. The other is its dependence on the bidirectionality of its components, coupled with the fact that the squaring devices in its mechanism fundamentally can never be bidirectional.

# Chapter 3

# References

**Garey and Johnson,** *Computers and Intractability: A Guide to the Theory of NP Completeness*, Freeman, San Francisco, 1979. A standard reference and introduction to the field of NP-completeness.

**Michael G. Main** "Analog Solution of NP-hard Problems", Department of Computer Science of University of Colorado at Boulder Technical Report CU-CS-700-94, Jan. 1994. This paper describes an electronic implementation of the 3-SAT machine.

**Peter W. Shor** "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484-1509. While neither prime factorization or computing discrete logarithms have been shown to be NP-complete, they are considered problems of such computational complexity that they form the basis of modern cryptographic systems.

**Vergis, Steiglitz, Dickinson,** "The Complexity of Analog Computation", *Mathematics and Computers in Simulation*, vol. 28, 1986, pp. 91-113. This is the source paper describing the 3-SAT machine.